**LAPORAN TUGAS AKHIR**

**PENGEMBANGAN SISTEM LAPORAN SMK3 FASYANKES BERBASIS WEB**

***DEVELOPMENT OF WEB BASED … REPORT SYSTEM***

***LOGO***

**FARIS TAKHASSUNA ZAMZANI**

**17/415514/SV/13379**

**PROGRAM STUDI D3 KOMPUTER DAN SISTEM INFORMASI**

**DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA**

**SEKOLAH VOKASI**

**UNIVERSITAS GADJAH MADA**

**YOGYAKARTA**

**2020**

**LAPORAN TUGAS AKHIR**

**PENGEMBANGAN SISTEM LAPORAN SMK3 FASYANKES BERBASIS WEB**

***DEVELOPMENT OF WEB BASED … REPORT SYSTEM***

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Ahli Madya

Komputer dan Sistem Informasi

***LOGO***

**FARIS TAKHASSUNA ZAMZANI**

**17/415514/SV/13379**

**PROGRAM STUDI D3 KOMPUTER DAN SISTEM INFORMASI**

**DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA**

**SEKOLAH VOKASI**

**UNIVERSITAS GADJAH MADA**

**YOGYAKARTA**

**2020**

**HALAMAN PENGESAHAN**

Iii

**PERNYATAAN BEBAS PLAGIASI**

### 

### **BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Fasyankes khususnya puskesmas merupakan salah satu tempat kerja yang memiliki risiko terhadap keselamatan dan kesehatan kerja baik pada SDM, pasien, pendamping pasien, pengunjung, maupun masyarakat di sekitar lingkungan Fasyankes. Potensi bahaya keselamatan dan kesehatan kerja di Fasyankes meliputi bahaya fisik, kimia, biologi, ergonomi, psikososial, dan bahaya kecelakaan kerja. Selain itu, adanya penggunaan berbagai alat kesehatan dan teknologi di Fasyankes serta kondisi sarana dan prasarana yang tidak memenuhi standar keselamatan akan menimbulkan risiko kecelakaan kerja dari yang ringan hingga fatal. Pelaporan tentang Sistem Manajemen Keselamatan dan Kesehatan Kerja (SMK3) sangat dibutuhkan untuk menjaga kesejahteraan SDM, pasien dan pengunjung. Namun, pada saat ini sistem pelaporan masih bersifat manual dan terkesan lambat dalam mengumpulkan laporan pada Dinas Kesehatan.

Hal lain yang perlu diperhatikan oleh SDM di Puskesmas adalah dengan masuknya virus corona (COVID-19) di Indonesia membuat kebijakan adanya pelaporan tentang keselamatan dan kesehatan kerja (K3) di fasilitas kesehatan perlu ditingkatkan. Selain itu berdasarkan studi pendahuluan dan observasi di beberapa puskesmas dihasilkan bahwa kegiatan pelaporan Sistem Manajemen K3 di Puskesmas masih belum berjalan secara optimal karena bingung dalam membuat laporan SMK3. Anggota Upaya Kesehatan Kerja (UKK) di Puskesmas juga memberikan informasi bahwa kegiatan pelaporan SMK3 merupakan salah satu bentuk penilaian dari sistem akreditasi puskesmas. Dengan adanya pelaporan SMK3 berbasis web ini diharapkan dapat membantu pelaporan Fasyankes khususnya Puskesmas ke Dinas Kesehatan di Kabupaten.

* 1. **Rumusan Masalah**

Rumusan masalah didapatkan dari latar belakang yang telah dijabarkan sebelumnya, tentang bagaimana mengembangkan sebuah sistem laporan berbasis web guna mempermudah proses pelaporan.

* 1. **Batasan Masalah**

Untuk menghindari penambahan fitur, maka perlu pembatasan terhadap masalah yang ada. Batasan masalah yang ada adalah sebagai berikut :

1. Aplikasi hanya dapat melakukan operasi CRUD (*Create, Read, Update, Delete)* konvensional atau pengolahan data secara biasa dan tidak terlalu kompleks.
2. Tampilan UI khususnya pada bagian pelaporan tidak dapat diubah melalui aplikasi itu sendiri.
3. Hanya pengguna dengan status *Admin* yang dapat membuat pengguna baru.
   1. **Tujuan Penelitian**

Tujuan penelitian adalah untuk mendesain, mengembangkan dan mengimplementasi aplikasi berbasis web yang memudahkan pelaporan antara puskesmas ke dinas kesehatan.

* 1. **Manfaat Penelitian**

Manfaat dari penelitian antara lain adalah sebagai berikut:

1. Mempermudah pembuatan laporan.
2. Mempermudah pengumpulan laporan dari unit puskesmas.
   1. **Metodologi Penelitian**

Pengembangan aplikasi ini melibatkan 2 bagian metodologi penelitian, yaitu metode pengumpulan data dan pengembangan aplikasi.

1. Metode Pengumpulan Data

Pengembangan aplikasi web didasarkan dari metode studi literatur, dengan dokumen yang dipelajari merupakan proposal yang diajukan dari pihak pemohon. Serta dokumentasi dan panduan yang ada untuk membantu pembuatan aplikasi.

1. Pengembangan Aplikasi

Pengembangan modul nilai mata uang dan biaya ini menggunakan model pengembangan perangkat lunak yaitu model Scrum. Model pengembangan ini merupakan peningkatan dan pembaruan dari beberapa model pengembangan yang berorientasi objek atau iterasi yang sudah umum digunakan. (Schwaber, 1997)

Model pengembangan ini terbagi menjadi 3[[1]](#footnote-0) tahap, antara lain:

a. Perencanaan dan perancangan

Pada tahap ini dilakukan pemetaan kebutuhan fungsional yang didasarkan pada *business logic* dari proposal yang diberikan serta penentuan durasi pengerjaan aplikasi.

b. Sprint pengembangan

Setelah kebutuhan fungsional sudah ditentukan, maka akan masuk pada tahap Sprint yaitu mulai mengembangkan aplikasi dengan menggunakan framework dan bahasa pemrograman yang sudah ditentukan.

c. Penutupan

Tahap ini merupakan tahap di mana akan dilakukan ulasan dan pengecekan modul yang telah dikembangkan, apakah sesuai dengan rencana dan rancangan pada tahap awal pengembangan.

* 1. **Sistematika Penelitian**

Penulisan laporan tugas akhir ini tersusun atas tujuh bab, yang dapat di urai menjadi sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini menjabarkan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan.

**BAB II TINJAUAN PUSTAKA**

Bab ini berisi penjabaran singkat tentang beberapa modul sejenis yang pernah dibuat sebelumnya dan digunakan sebagai referensi dalam penelitian tugas akhir ini.

**BAB III LANDASAN TEORI**

Bab ini berisi tentang teori, konsep, prinsip dan pendapat yang berasal dari buku dan sumber lain yang terkait dengan pengembangan modul yang dapat dipertanggungjawabkan secara ilmiah. Teori-teori yang dimuat dalam bab ini meliputi teori yang sesuai dan terkait dengan modul yang dikembangkan.

**BAB IV ANALISA DAN PERANCANGAN SISTEM**

Pada bab ini dijelaskan tahapan-tahapan analisis dan perancangan sistem. Tahapan analisis terdiri atas deskripsi, *use case* aplikasi, dan alur olah data. Proses perancangan meliputi perancangan tampilan antarmuka pengguna, dan alur olah data.

**BAB V IMPLEMENTASI MODUL**

Bab ini menjelaskan tentang rincian penerapan atau pengimplementasian dari rancangan modul yang telah dibangun. Rincian implementasi berupa potongan *source code* dan penjelasannya.

**BAB VI PENGUJIAN DAN PEMBAHASAN**

Pada bab ini membahas proses dan hasil pengujian modul yang telah dibangun, serta integrasi terhadap produk atau layanan pengiriman uang dari dan ke luar negeri.

**BAB VII PENUTUP**

Bab ini berisi kesimpulan dari pengembangan modul nilai tukar uang dan biaya. Pada bab ini juga terdapat saran untuk penelitian selanjutnya.

**DAFTAR PUSTAKA**

Pada bab ini memuat sumber literatur yang digunakan penulis sebagai acuan dalam mengembangkan modul.

**LAMPIRAN**

### **BAB II**

**TINJAUAN PUSTAKA**

Guna mendapatkan hasil penelitian yang memiliki dasar yang kuat, maka dilakukanlah tinjauan pustaka yang mengacu pada beberapa sumber yang dapat dipertanggung-jawabkan kredibilitasnya. Di bawah tertera sumber pustaka yang terkait pengembangan aplikasi sistem manajemen keselamatan dan kesehatan kerja berbasis web.

Ohchi, M., Furukawa, T., & Tanaka, S.-I.(2008), mengembangkan aplikasi yang digunakan untuk membantu pembelajaran mengenai analisis numerik pada medan elektromagnetik. Aplikasi ini dibuat dengan bahasa pemrograman Java dan berbasis sistem arsitektur klien server. Pemilihan arsitektur ini agar aplikasi dapat dipakai tanpa mempedulikan sistem operasi dan perangkat keras pengguna.

Hasibuan, A., Mustadi, M., Syamsuddin, I. E. Y., & Rosidi, I. M. A.(2015) mendesain dan mengembangkan sistem IoT otomatisasi rumah yang modular. Sistem ini didesain untuk mempermudah personalisasi sistem dari *smart house*. Sistem ini dibagi menjadi 3 fungsi, yaitu *device control, device status* dan *device automation*. Fungsi tersebut saling berkomunikasi dengan REST API.

Adhoni, Z. A., Hamad, H. A., & Adhoni, Z. A.(2014) mengembangkan portal penggunaan Al-Quran yang memanfaatkan teknologi *cloud*. Portal di kembangkan dengan teknologi PHP, MySQL, Apache dan Drupal. Hasil akhir dari penelitian ini merupakan portal yang dapat diakses pengguna melalui internet dimana pengguna dapat melakukan pencarian Surat dan Ayat.

Dari tinjauan pustaka diatas, penulis mengembangkan aplikasi sistem manajemen kesehatan dan keamanan kerja berbasis web menggunakan *framework* React.js sebagai bagian *frontend* dan Node.js dan Express.js sebagai *backend.* Bahasa yang digunakan adalah JavaScript serta menggunakan MongoDB sebagai *database*. Aplikasi ini dikembangkan untuk mempermudah pembuatan dan pengelolaan laporan kesehatan dan keamanan fasilitas pelayanan kesehatan. Pada tampilan antarmuka akan berupa daftar, detail dan tampilan pengolahan data lainnya, sedangkan pada bagian *backend* merupakan *middleware* dan *database*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Penulis** | **Judul** | **Teknologi** | **Deskripsi** |
| Ohchi, M., Furukawa, T., & Tanaka, S.-I. | Development of Education Support System for Numerical ElectromagneticAnalysis Based on Server–Client Model Using Java | Java, UML, Swing | Aplikasi yang mendukung pendidikan analisis numerik elektromagnetik berdasarkan model *client - server* dengan GUI |
| Hasibuan, A., Mustadi, M., Syamsuddin, I. E. Y., & Rosidi, I. M. A.  . | Design and implementation of modular home automation based on wireless network, REST API, and WebSocket. 2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS). | IoT (Internet of Things), WebSocket, SQLite | Desain perangkat modular untuk mengotomatisasikan penggunaan peralatan rumah |
| Adhoni, Z. A., Hamad, H. A., & Adhoni, Z. A. | An API for Quran portal using Drupal technology. The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT). | PHP, MySQL, Apache, | Portal untuk penelusuran dan penggunaan Al-Quran berbasis *cloud* |

### 

### 

### **BAB III**

**LANDASAN TEORI**

* 1. **Konsep dasar API**

API merupakan akronim dari Application Programming Interface atau Antarmuka Pemrograman Aplikasi. Daniel Jacobson, Dan Woods, Greg Brail (2012) mengatakan bahwa API merupakan cara aplikasi dari 2 komputer yang berbeda berkomunikasi melalui sebuah jaringan menggunakan bahasa yang biasa digunakan yang dimengerti kedua pihak.

Dalam konteks aplikasi ini, API menjembatani komunikasi antara dua sisi aplikasi ini yaitu *Client* dengan *Server*. API memungkinkan keduanya untuk berkomunikasi tanpa harus mengetahui penerapannya.

* + 1. **REST API**

Rest API adalah web API yang menggunakan konsep dari arsitektur REST (Representational State Transfer). Karena Rest API adalah turunan dari arsitektur REST, maka tidak ada standar resminya. Didefinisikan oleh Roy Fielding (2000), bahwa API sudah tergolong Rest API selama memenuhi enam syarat yang dimiliki oleh sistem yang berarsitektur REST, yaitu:

1. *Client-Server*

Arsitektur REST terdiri dari klien, server, dan sumber daya yang

dihubungkan dengan requests melalui HTTP.

1. *Stateless*

Tidak ada konten informasi klien yang disimpan di server saat requests.

Informasi tentang sesi requests tetap disimpan oleh klien.

1. *Cache*

Metode caching mampu mengeliminasi kebutuhan untuk terus berinteraksi

antara klien dengan server.

1. *Uniform Interface*

Dengan menerapkan prinsip generalisasi pengembangan aplikasi terhadap

*interface* komponen, keseluruhan sistem arsitektur menjadi sederhana dan

visibilitas interaksi meningkat.

1. *Layered System*

Interaksi antara klien dengan server dapat diberi mediasi dengan layer tambahan. Layer tambahan ini dapat berupa load balancing, dan keamanan lanjutan.

1. *Code on Demand*

Server dapat menambahkan ekstra fungsionalitas kepada klien dengan mengirim kode executable.

* 1. **Siklus pengembangan Sistem**

Jogiyanto (2005) menyebutkan bahwa pengembangan sistem pembuatan sistem baru yang lebih mutakhir dengan tujuan mengganti dan memperbarui sistem yang sudah ada. Termasuk di dalamnya adalah metode pengembangan berjenis agile development. Agile development memiliki beberapa turunan lagi, salah satunya yaitu Scrum. Schwaber, dan Ken (1997) menyebutkan bahwa metode pengembangan Scrum ini adalah manajemen, pemutakhiran, dan pemeliharaan untuk sebuah sistem yang sudah ada maupun yang masih dalam tahap prototipe produksi.

* + 1. **Komponen Tim Pada Scrum**

1. Manajemen

Dipimpin oleh seorang Product Manager, mendefinisikan konten atau kebutuhan awal dan jadwal dari produk yang akan dirilis, lalu mengelola perubahan mereka ketika proyek berjalan dan variabel berubah. Manajemen mengatur backlog, resiko, dan konten atau produk yang akan dirilis.

1. Tim pengembang

Tim pengembang relatif sedikit, dengan terdiri dari pengembang, dokumenter, dan staf kontrol kualitas. Para pengembang dapat dibagi lagi menjadi beberapa tim yang terdiri dari tiga hingga enam orang. Setiap tim ditugaskan untuk mengembangkan suatu modul spesifik, termasuk backlog yang terkait dengan modul tersebut. Tim pengembang dikelompokkan berdasarkan keahlian dan spesifikasi masing-masing.

* + 1. **Karakteristik Scrum**

Proyek scrum memiliki karakteristik sebagai berikut:

1. Keluaran yang fleksibel – isi dari keluaran yang dihasilkan ditentukan oleh environment.
2. Jadwal yang fleksibel – keluaran yang dihasilkan terkadang dapat dibutuhkan lebih cepat atau lebih lama dari yang sudah direncanakan.
3. Tim yang kecil – tiap tim terdiri tidak lebih dari 6 anggota. Sangat dimungkinkan ada beberapa tim dalam suatu proyek.
4. Ulasan yang sering – progress tim diulas sesering mungkin sesuai kompleksitas proyek (biasanya satu hingga empat minggu dalam satu siklus).
5. Kolaborasi – intra dan inter-kolaborasi sangat diharapkan selama pengerjaan proyek.
6. Berorientasi Objek – setiap tim akan diberi tugas dengan objek terkait, dengan spesifikasi dan kebutuhan yang jelas.
   * 1. **Keuntungan Metodologi Scrum**

Metodologi scrum dirancang sedemikian rupa untuk benar-benar fleksibel selama proses pengembangan produk. Metode ini menyediakan mekanisme kontrol untuk perencanaan rilis produk, lalu mengelola variabel selama proses pengembangan berjalan. Hal ini memungkinkan organisasi atau perusahaan untuk mengubah proyek dan hasil keluaran, pada waktu kapanpun, demi memastikan rilis produk yang paling dikehendaki. Metodologi scrum membebaskan pengembang untuk merancang dan menerapkan solusi terbaik yang mereka punyai terhadap proyek yang sedang dikerjakan, ini juga dikarenakan terjadinya pembelajaran di lingkup pengembang itu sendiri. Teknologi yang berorientasi objek adalah basis dari metodologi scrum. Objek, fitur, atau kebutuhan bisnis, menawarkan lingkungan pengembangan yang terpisah dan mudah untuk dikelola.

* 1. **Perangkat Lunak Modul**

Dalam pengembangan aplikasi ini, digunakan beberapa *framework* dan layanan pendukung.

* + 1. **JavaScript**

JavaScript merupakan bahasa pemrograman yang ringan, ter interpretasikan, atau *just-in-time compiled* dengan fungsi sebagai *first-class citizen*. Walaupun lebih dikenal sebagai bahasa *scripting* untuk halaman Web, banyak lingkungan non-browser yang menggunakannya seperti Node Js, Apache CouchDB dan Adobe Acrobat. JavaScript merupakan bahasa yang berbasis prototipe, berbagai paradigma, *single-threaded*, dan dinamik yang mendukung pemrograman berbasis objek, imperatif, dan deklaratif.

Walaupun memiliki nama yang hampir sama dengan bahasa pemrograman Java, keduanya merupakan hal yang berbeda.

* + 1. **React.js**

React.js merupakan suatu *library* untuk JavaScript yang digunakan untuk mengembangan tampilan atau *view* pada model MVC. React.js dikembangkan oleh Facebook pada akhir tahun 2011, namun kemudian menjadi *open source* pada tahun 2013. React.js memiliki keunggulan di antara lainnya adalah:

1. Deklaratif

Deklaratif dalam konteks ini adalah bagaimana pengguna React tidak harus mendefinisikan seluruh bagian dari sebuah antarmuka, namun hanya menggambarkan hari akhir atau yang diinginkan. Hal ini membantu tampilan antarmuka menjadi lebih interaktif.

1. Berbasis komponen

Komponen merupakan unit dasar dan unit terkecil suatu aplikasi berbasis React, dengan komponen pengembang harus berpikir antarmuka dari sisi bagian - bagian tiap tampilan. Dengan mengkombinasi dan menggabungkan berbagai macam komponen ke dalam satu antarmuka, kode dapat digunakan ulang sebanyak mungkin dan menjadi efisien dalam penulisannya.

Penulis menggunakan react-admin, sebuah *library* yang menggunakan React dan material-ui sebagai *framework* antarmuka.

* + 1. **Node.js**

Node.js merupakan JavaScript *runtime* yang tidak sinkron dan didorong oleh *event*, Node didesain untuk membuat aplikasi yang berskala. Kemampuan untuk tidak sinkron membuat Node.js lebih cepat dan efisien dalam melakukan eksekusi kode. Node memungkinkan penggunaan JavaScript sebagai bahasa pemrograman aplikasi karena pengembang dapat menggunakan JavaScript diluar *web browser*.

Node.js dirilis pada tahun 2009 danPada bagian awal,nya hanya mendukung sistem operasi Linux dan MacOS, namun Microsoft mengimplementasi versi Windows pada tahun 2011.

* + 1. **Express.js**

Express.js merupakan *web framework* untuk Node yang ringan dan tidak beropini. Express memberikan berbagai macam fungsionalitas yang penting dalam membuat aplikasi web yang tidak didukung oleh Node langsung.

Express dirilis pada tahun 2010 sebagai *software* yang gratis dan *open source.* Express didirikan oleh TJ Holowaychuk, namun kemudian hak untuk mengelola diperoleh StrongLoop di tahun 2014. StrongLoop sendiri di akuisisi oleh IBM pada tahun 2015.

* + 1. **MongoDB**

MongoDB merupakan program basis data berbasis dokumen yang dapat dipakai di berbagai *platform* yang terklasifikasi sebagai NoSQL. NoSQL berarti basis data non-relasional. Struktur data dari basis data non-relasional tidak tetap. Ada jenis penyimpanan kunci (nilai kunci), jenis penyimpanan dokumen dan sebagainya. MongoDB adalah database berorientasi dokumen. Bebas skema, yang berisi Basis Data, Koleksi, dan Dokumen. Satu Basis Data dapat memiliki beberapa Koleksi. Setiap Koleksi adalah kumpulan Dokumen. Pada model SQL koleksi dapat disamakan dengan baris, dan dokumen dengan kolom. Dokumen sendiri terdiri dari *key* dan *value*.

MongoDB mulai dikembangkan pada tahun 2007 oleh perusahaan perangkat lunak 10gen, yang kemudian berganti nama menjadi MongoDB Inc.

* 1. **Alat Implementasi**

Untuk membantu implementasi dari hasil perancangan, analisis dan desain sistem pelaporan berbasis web ini, penulis menggunakan beberapa alat sebagai berikut:

* + 1. **Sublime Text 3**

Sublime Text 3 merupakan editor teks yang terdapat di berbagai *platform* dengan model shareware dan menggunakan antarmuka pemrograman aplikasi Python. Sublime secara langsung mendukung banyak bahasa pemrograman dan bahasa markup, dan fungsi dapat ditambahkan oleh pengguna dengan *plugin*, biasanya dibuat oleh komunitas dan dikelola di bawah lisensi perangkat lunak bebas.

Sublime Text 3 dipilih sebagai editor penulis karena kecepatannya dalam penggunaan, ringannya dalam penggunaan resource serta fungsi khusus lain seperti *Goto Anything* dan *Multiple Selection*. Banyaknya fitur yang diberikan Sublime dan *plugin* yang ada meningkatkan produktivitas penulis dan mempercepat implementasi sistem.

* + 1. **Github**

GitHub, Inc. adalah perusahaan global yang berbasis di Amerika Serikat yang menyediakan hosting untuk pengembangan perangkat lunak dan kontrol versi menggunakan Git. Github didirikan pada tahun 2008, dan kemudian di akuisisi oleh Microsoft pada tahun 2018.

* + 1. **Postman**

Postman merupakan sebuah aplikasi yang berfungsi sebagai klien REST untuk menguji REST API yang sudah dikembangkan. Postman memiliki keunggulan utama dibandingkan dengan aplikasi sejenis lainnya karena memiliki fitur workspace yang memungkinkan para pengembang untuk berkolaborasi dan menguji API secara real-time.

### **BAB IV**

**ANALISIS PERANCANGAN**

* 1. **Analisis Aplikasi**

Setiap aplikasi yang dibuat merupakan salah satu tahapan dari solusi suatu permasalahan, dan untuk memaksimalkan efektifitas aplikasi maka diperlukan tahapan analisis. Tahapan ini berfungsi untuk mendalami masalah dan memetakan masalah yang ada. Dalam tahap ini analisis dilakukan terhadap masalah produk atau layanan, kebutuhan fungsional, dan non fungsional modul.

* + 1. **Alur Data dan Cara Kerja REST API**

Dalam penggunaan dan penampilan data, aplikasi berkomunikasi menggunakan API. API yang ada menjembatani antara *Client* dan *Server*. Konfigurasi ini dikenal dengan *Client-Server Architecture*. *Client* akan melakukan *request* pada *endpoint* yang disediakan oleh *server* untuk melakukan fungsi tertentu, yang kemudian oleh *server* mengembalikan respon sesuai dengan *request*. *Endpoint* merupakan suatu alamat yang dibuka oleh *server* yang dapat digunakan oleh *client*, aksi yang dilakukan meliputi GET, POST, PUT dan UPDATE.

Dalam aplikasi ini, klien akan melakukan *request* melalui protokol HTTP yang kemudian akan diproses oleh server. Server kemudian akan mengembalikan data berdasarkan *query* oleh klien atau melakukan aksi lain berdasarkan *request* dari klien. Data yang dikirim yang berupa JSON akan digunakan oleh klien untuk menampilkan hasil *request*.

* 1. **Analisis Masalah**

Dalam operasi sehari harinya, Fasilitas Pelayanan Kesehatan (Fasyankes) memiliki potensi risiko terhadap keselamatan dan kesehatan warga Fasyankes. Salah satu cara untuk mengurangi kemungkinan dan dampak hal tersebut adalah adanya Sistem Manajemen Keselamatan dan Kesehatan Kerja (SMK3). Dengan sistem pelaporan ini, setiap Fasyankes selalu sadar dengan risiko yang ada dan cara pencegahan dan penangannya. Namun, dalam praktek hal ini menghambat produktivitas SDM Fasyankes karena pengoperasian yang masih manual dan terkesan lambat.

Untuk memecahkan masalah tersebut, maka di usulkan penggunaan teknologi yang ada untuk memudahkan pembuatan dan pengumpulan laporan. Usulan yang diajukan dari proposal yang diterima penulis adalah untuk membuat suatu aplikasi SMK3 berbasis web. Aplikasi ini diharapkan untuk dapat melakukan operasi pembuatan, pembacaan, pembaruan, dan penghapusan. Dengan perubahan sistem pelaporan dari operasi manual ke web, produktivitas dan efektivitas SDM Fasyankes dapat meningkat serta mempermudah pengurangan dan pencegahan risiko yang dihadapi Fasyankes setiap harinya

* 1. **Analisis Kebutuhan Fungsional**

Berdasarkan hasil analisa sebelumnya, maka aplikasi ini memiliki kebutuhan sebagai berikut:

1. Aplikasi dapat menampilkan data yang disimpan.
2. Aplikasi dapat membuat data baru.
3. Aplikasi dapat memperbaharui data yang disimpan.
4. Aplikasi dapat menghapus data yang disimpan.
5. Aplikasi dapat di akses dari berbagai lokasi sekaligus.
6. Aplikasi dapat peringatan terhadap pengguna yang ditunjuk sebagai pembuat laporan dalam bentuk SMS dan *Email*
   1. **Analisis Kebutuhan Non Fungsional**

Setelah mengetahui kebutuh fungsional aplikasi, kebutuhan non fungsional dapat dianalisa. Kebutuhan fungsional yang ada merupakan kebutuhan aplikasi yang mendukung berjalannya aplikasi dengan maksimal dan efektif. Kebutuhan yang ada adalah sebagai berikut:

1. Aplikasi memerlukan jaringan internet.
2. Aplikasi hanya dapat diakses setelah mengkonfirmasi identitas dengan melakukan aksi *login*.
3. Fungsionalitas aplikasi hanya dapat digunakan di klien.
   1. **Perancangan Aplikasi**

Untuk mempermudah implementasi aplikasi, maka dilakukan tahap perancangan aplikasi. Perancangan difokuskan pada basis data, proses, serta tampilan antarmuka aplikasi

* + 1. **Perancangan Basis Data**

Pada pengimplementasian aplikasi, basis data yang memerlukan beberapa unsur penting yaitu laporan, pengguna, peringatan. Unsur pelengkap dari unsur penting berupa fasyankes. Unsur tersebut akan diubah menjadi tabel atau dalam kasus ini skema dokumen karena penggunaan basis data NoSQL.

1. Laporan

Data laporan yang ada dibagi menjadi dua, yaitu laporan per tahun dan laporan per semester. Hal ini sesuai dengan proposal yang diterima penulis dan untuk mengoptimalkan kecepatan aplikasi. Kedua laporan tersebut memiliki struktur sebagai berikut

1. Tabel *year\_report*

Tabel *year\_report* merupakan tabel yang memuat nilai laporan per tahun yang ada. Tabel memiliki atribut *\_id,* *author*, *total*, *year*, *institution*, *report*. Rincian tabel *year\_report* adalah sebagai berikut:

|  |  |  |
| --- | --- | --- |
| **Atribut** | **Tipe Data** | **Keterangan** |
| **\_**id | Integer | *Unique Identifier* yang dibuat otomatis oleh MongoDB |
| author | ObjectId | Mengarah ke *unique identifier* tabel *user* |
| total | Number | Jumlah SDM di Fasyankes |
| year | Date | Tanggal waktu dibuatnya laporan |
| institution | ObjectId | Mengarah ke *unique identifier* tabel *institution* |
| report | Object | Berisi laporan SMK3 per tahun |

1. Tabel *semester\_report*

Tabel *semester\_report* merupakan tabel yang memuat nilai laporan per semester yang ada. Tabel memiliki atribut *\_id,* *author*, *month*, *year*, *institution*, *report*. Rincian tabel *semester\_report* adalah sebagai berikut:

|  |  |  |
| --- | --- | --- |
| **Atribut** | **Tipe Data** | **Keterangan** |
| **\_**id | ObjectId | *Unique Identifier* yang dibuat otomatis oleh MongoDB |
| author | ObjectId | Mengarah ke *unique identifier* tabel *user* |
| month | String | Berisi bulan pembuatan laporan |
| year | Date | Tanggal waktu dibuatnya laporan |
| institution | ObjectId | Mengarah ke *unique identifier* tabel *institution* |
| report | Object | Berisi laporan SMK3 per tahun |

1. Pengguna

Pengguna memuat semua data informasi akun yang dapat melakukan otentikasi dan diizinkan untuk melakukan operasi pada aplikasi. Pengguna diwakili pada tabel *users* di basis data aplikasi ini.

Tabel *users* merupakan tabel yang memuat nilai laporan per semester yang ada. Tabel memiliki atribut *\_id,* *username*, *first\_name*, *full\_name*, *email*, *phonenumber*, *privilege*, *job\_title*, *nip*, *password*. Rincian tabel *users* adalah sebagai berikut:

|  |  |  |
| --- | --- | --- |
| **Atribut** | **Tipe Data** | **Keterangan** |
| **\_**id | ObjectId | *Unique Identifier* yang dibuat otomatis oleh MongoDB |
| username | String | Nama akun pengguna yang digunakan ketika otentikasi |
| first\_name | String | Nama depan asli pengguna |
| full\_name | String | Nama lengkap pengguna |
| email | String | Email pengguna yang aktif |
| phonenumber | Number | Nomor handphone pengguna |
| privilege | String | Level izin yang dimiliki oleh akun pengguna |
| job\_title | String | Jabatan pengguna |
| nip | String | NIP pengguna apabila ada |
| password | String | Kata sandi pengguna untuk melakukan otentikasi |

1. Peringatan

Peringatan memuat informasi peringatan yang dibuat. Dokumen ini berfungsi untuk menyimpan status peringatan serta semua peringatan yang telah dibuat. Peringatan diwakili pada tabel *notification* di basis data aplikasi ini.

Tabel *notification* merupakan tabel yang memuat nilai laporan per semester yang ada. Tabel memiliki atribut *\_id,* *complete\_status*, *remindee*, *remind\_date*, *report\_type*. Rincian tabel *notification* adalah sebagai berikut:

|  |  |  |
| --- | --- | --- |
| **Atribut** | **Tipe Data** | **Keterangan** |
| **\_**id | Integer | *Unique Identifier* yang dibuat otomatis oleh MongoDB |
| complete\_status | Boolean | Status penyelesaian peringatan |
| remindee | ObjectId | Mengarah ke *unique identifier* tabel *user* pembuat *notification* |
| reminde\_date | Date | Tanggal waktu peringatan laporan |
| report\_type | String | Jenis laporan yang diberitahukan ke *users* |

1. Fasyankes

Fasyankes memuat informasi fasyankes laporan yang dibuat. Dokumen ini berfungsi untuk menyimpan informasi dan data fasyankes. Fasyankes diwakili pada tabel *institution* di basis data aplikasi ini.

Tabel *institution* merupakan tabel yang memuat nilai laporan per semester yang ada. Tabel memiliki atribut *\_id,* *name*, *address*, *area*, *city*, *province*. Rincian tabel *institution* adalah sebagai berikut:

|  |  |  |
| --- | --- | --- |
| **Atribut** | **Tipe Data** | **Keterangan** |
| **\_**id | Integer | *Unique Identifier* yang dibuat otomatis oleh MongoDB |
| name | String | Nama Fasyankes |
| address | String | Alamat Fasyankes |
| area | Number | Luas Fasyankes |
| city | String | Kota/Kabupaten Fasyankes terletak |
| province | String | Provinsi Fasyankes terletak |

* + 1. **Perancangan Proses**

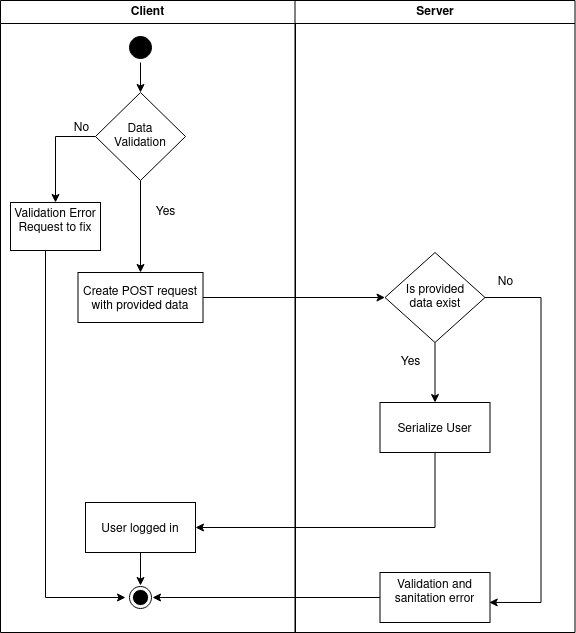
Untuk membuat aplikasi yang fungsinya sesuai dengan hasil analisa dan perancangan sebelumnya, penulis melakukan perancangan proses. Pada perancangan proses dijelaskan mengenai alur data menggunakan diagram alir (*flowchart*). Diagram alir ini aka diwakili dengan model UML.

Diagram alir berguna untuk memastikan bahwa aliran data berjalan sesuai dengan rancangan. Dari awal atau *request* klien hingga *response* dari server yang kemudian digunakan oleh klien. Diagram alir digunakan sebagai acuan pada perancangan proses *backend* dan algoritma lain.

Dikarenakan beberapa operasi terdapat banyak kesamaan, maka penulis hanya menjelaskan 3 operasi. Operasi yang dijelaskan adalah *login*, *CRUD*,dan *notification*.

1. *Login*

Proses API untuk melakukan login memerlukan masukan berupa username dan password. Saat endpoint API dipanggil, maka server akan menerima request dengan tipe POST. Server kemudian mengecek apakah username ada di basis data, yang apabila ada akan mengecek password yang diberikan pengguna dengan password yang ada di basis data. Apabila semuanya terdapat di basis data maka pengguna terotentikasi ke dalam aplikasi.

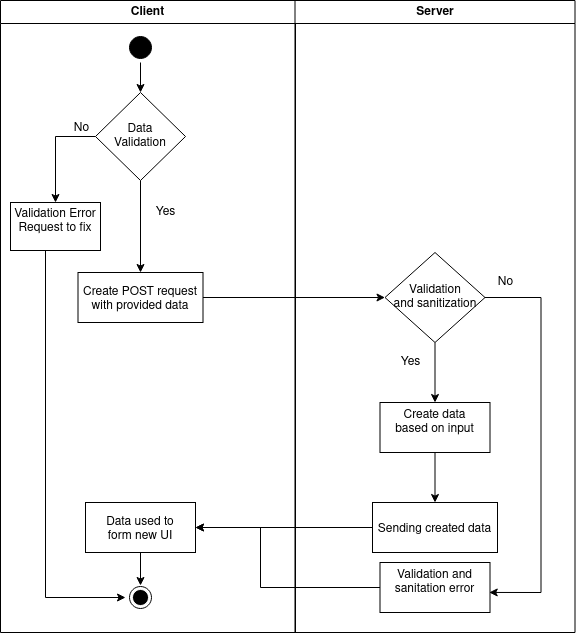


1. *CRUD*

Dikarenakan banyaknya operasi yang sama, maka penulis mengelompokkan beberapa ke diagram alir ini. Operasi CRUD yang digabung adalah operasi CRUD pada *user*, *year\_report*, *semester\_report*, *notification* dan *institution*.

* *Create*

Proses API untuk melakukan operasi *create* memerlukan masukan dari user. Data akan divalidasi oleh klien sebelum memanggil API. Saat endpoint API dipanggil, maka server akan menerima request dengan tipe POST. Server kemudian mengecek apakah data valid dan membersihkan data. Apabila data masukan sesuai dengan skema yang ditentukan maka data akan dibuat dan disimpan ke basis data. Data yang dibuat akan dikirim sebagai JSON ke klien untuk membentuk antarmuka baru.

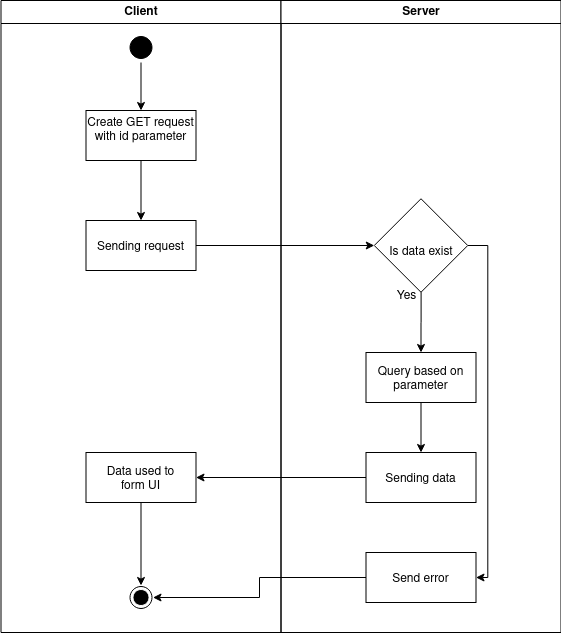


* *Read*

Proses API untuk melakukan operasi *read* dibagi menjadi dua yaitu memanggil satu data (*Get One*) atau semua data (*Get All*).

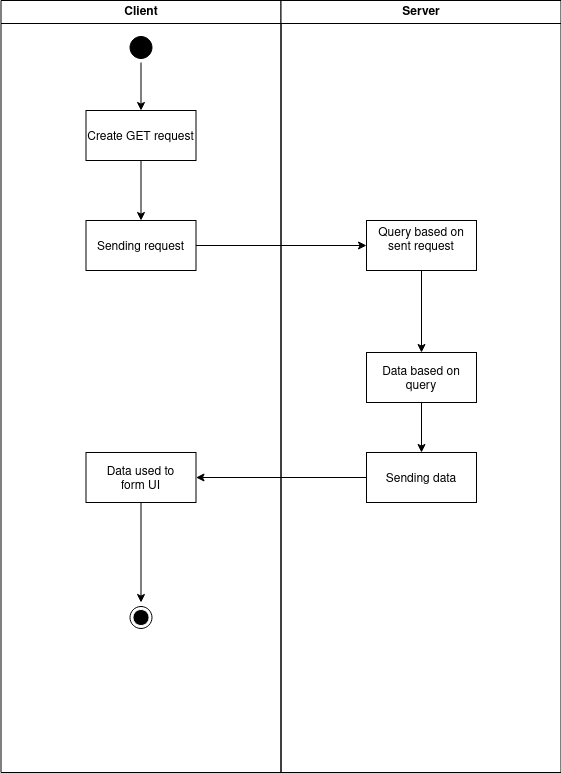
1. *Get One*

Proses ini memerlukan parameter. Saat endpoint API dipanggil, maka server akan menerima request dengan tipe GET. Server kemudian mengecek apakah data dengan parameter diberikan ada. Apabila data ditemukan maka server akan mengirim data sebagai JSON ke klien untuk membentuk antarmuka baru.



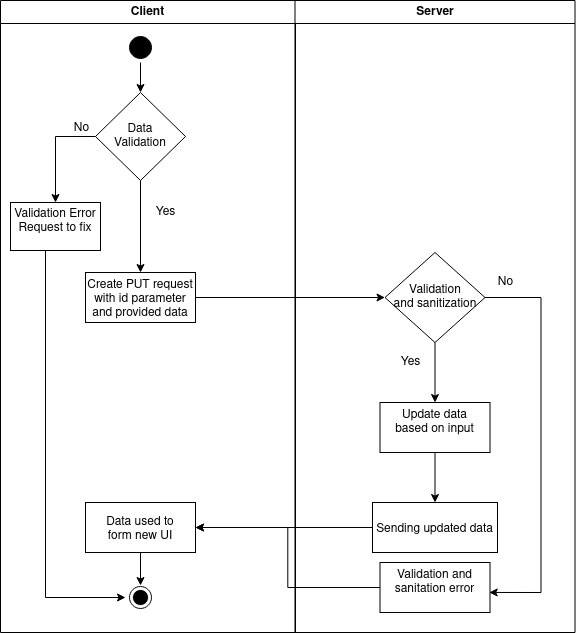
1. *Get All*

Saat endpoint API dipanggil, maka server akan menerima request dengan tipe GET. Data oleh server akan dikirim sebagai JSON ke klien untuk membentuk antarmuka baru.



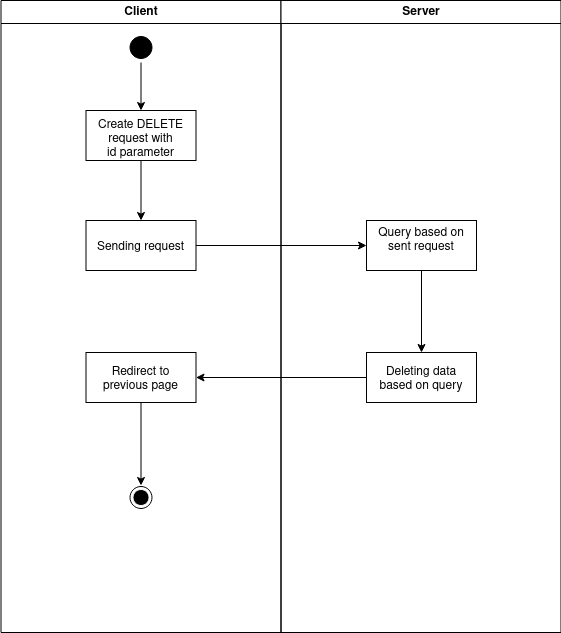
* *Update*

Proses API untuk melakukan operasi *update* memerlukan parameter. Data akan divalidasi oleh klien sebelum memanggil API. Saat endpoint API dipanggil, maka server akan menerima request dengan tipe PUT. Server kemudian mengecek apakah data berdasarkan parameter ada pada basis data. Server kemudian mengecek apakah data valid dan membersihkan data. Apabila data masukan sesuai dengan skema yang ditentukan maka data akan diubah dan disimpan ke basis data. Data yang dibuat akan dikirim sebagai JSON ke klien untuk membentuk antarmuka baru.



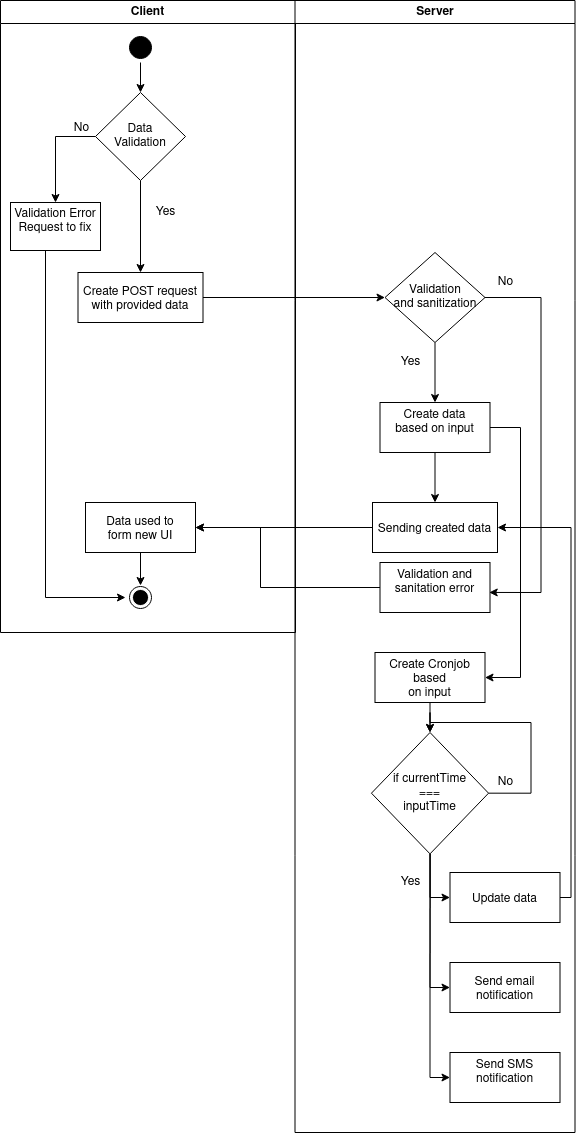
* *Delete*

Proses API untuk melakukan operasi *delete* memerlukan parameter. Saat endpoint API dipanggil, maka server akan menerima request dengan tipe DELETE. Server kemudian mengecek apakah data berdasarkan parameter ada pada basis data. Apabila data ada, maka data akan dihapus dari basis data, klien kemudian akan mengarahkan pengguna ke halaman sebelumnya.



1. *Notification*

Proses API *create* untuk *notification* memiliki beberapa perbedaan dibanding proses yang ada pada CRUD sebelumnya. Perbedaan terdapat setelah data disimpan ke basis data. Server tidak hanya mengirim data yang dibuat, namun juga melakukan pembuatan Cronjob berdasarkan data yang diberikan. Cronjob akan terus mengecek waktu yang apabila sesuai dengan waktu yang diberikan oleh pengguna maka akan melakukan beberapa aksi. Aksi yang dilakukan adalah pertama mengirim *email* ke akun yang ada pada pembuatan *notification*, kemudian server akan mengirim SMS ke handphone pengguna sesuai dengan nomor handphone yang diberikan, terakhir mengubah data pada *notification* yang sesuai dengan kondisi.

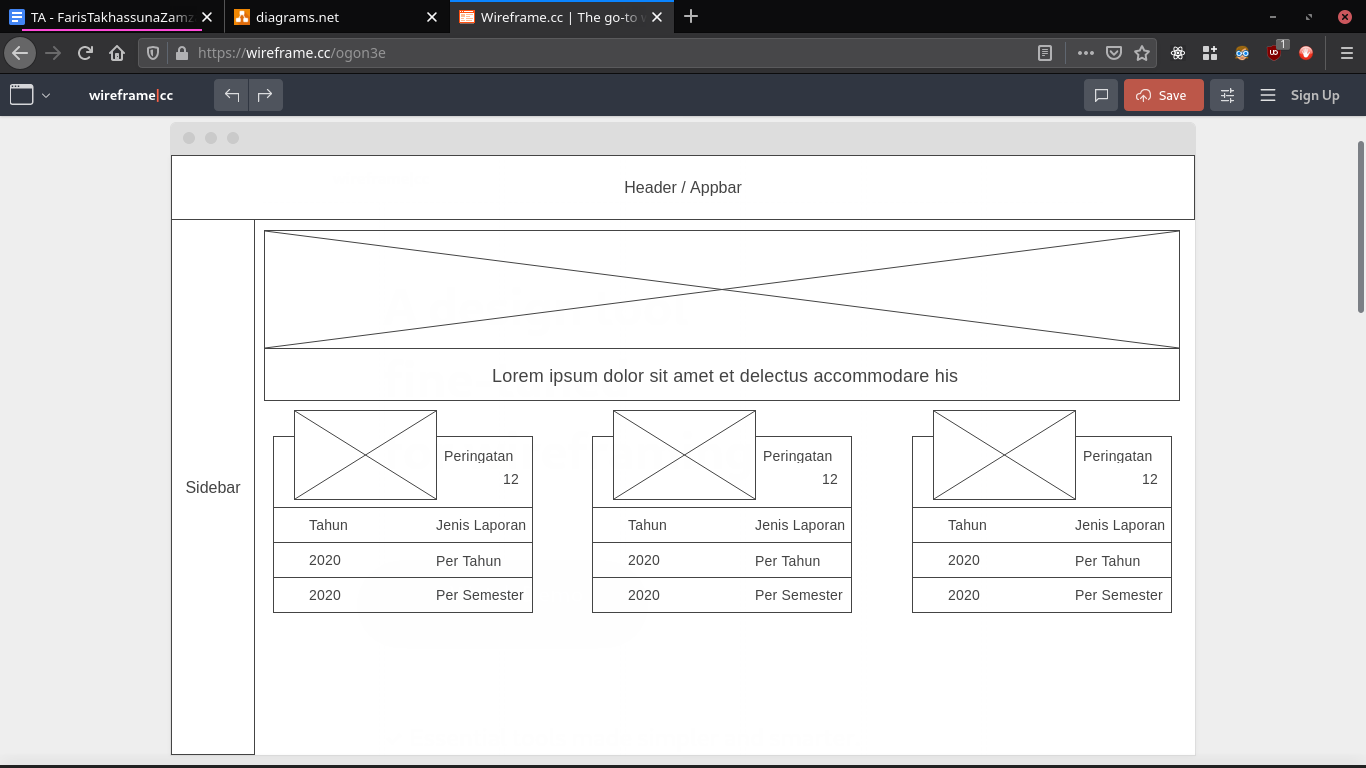


* + 1. **Perancangan Tampilan Antarmuka**

Antarmuka sebagai tampilan yang digunakan oleh pengguna untuk berinteraksi dengan aplikasi memerlukan perhatian khusus. Untuk mempermudah implementasi serta keefektifan antarmuka diperlukan perancangan tampilan antarmuka.

Berikut merupakan rancangan tampilan *dashboard*, dan halaman *CRUD* yang diwakili oleh halaman *users*.

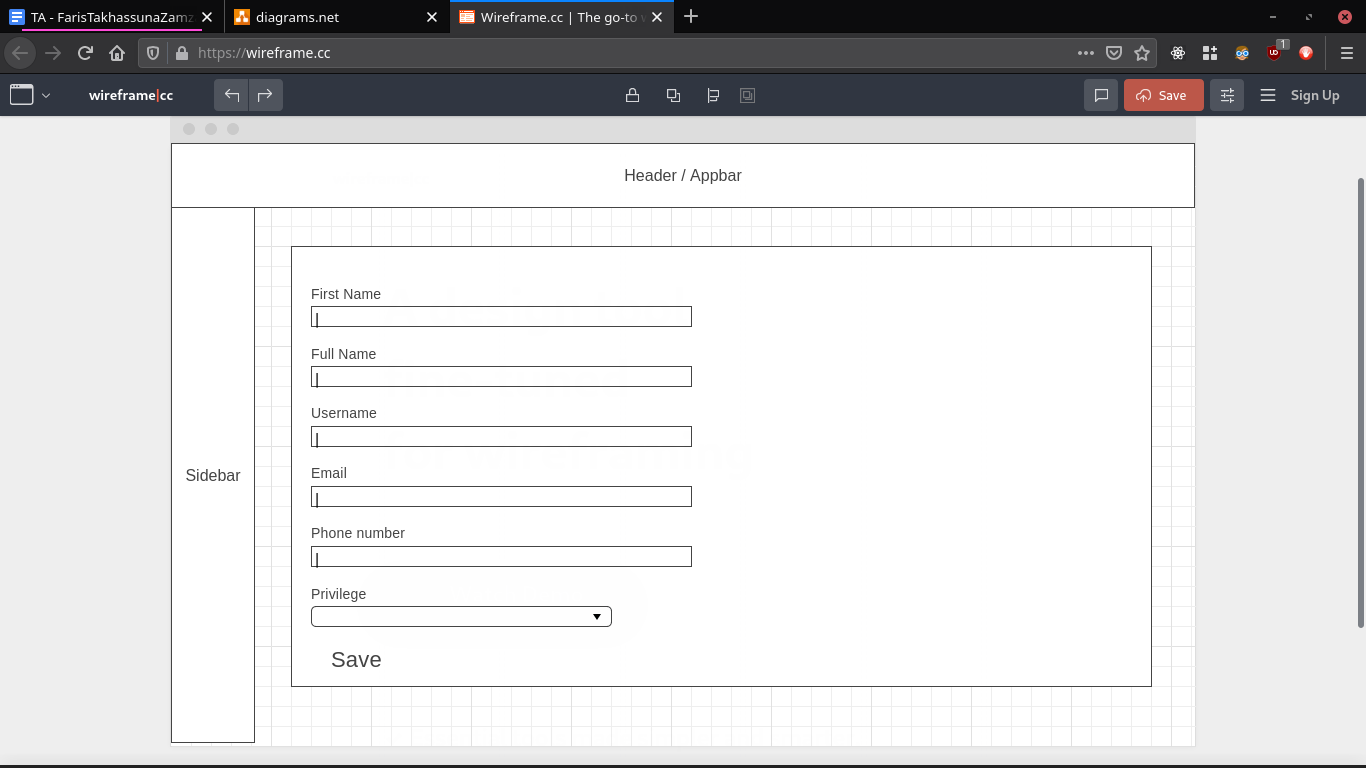
1. *Dashboard*



1. *CRUD*

Dikarenakan kesamaan yang banyak antar halaman, maka penulis mengelompokkan beberapa halaman menjadi satu rancangan. Halaman yang digabung adalah Pengguna, Peringatan, Fasyankes, Laporan per Tahun, Laporan per Semester.

* Halaman pembuatan data baru



* Halaman pengubahan data



* Halaman tabel semua data



* Halaman detail satu data



### **BAB V**

**IMPLEMENTASI**

Sebagai wujud realisasi dari perancangan aplikasi diatas, maka dalam bab ini dijelaskan implementasi atau penerapan dari penjelasan tersebut. Bab ini akan menjelaskan bagaimana pembuatan kode program sehingga menjadi sebuah aplikasi berbasis web yang mampu digunakan sebagai sistem manajemen.

* 1. **Implementasi Perangkat Lunak Pembangun**

Sebagai alat untuk mengembangkan aplikasi ini, berikut merupakan daftar perangkat lunak yang digunakan selama pengerjaan aplikasi yang meliputi:

1. Manjaro Linux XFCE 20.0.3 Lysia sebagai sistem operasi
2. Sublime Text 3 sebagai *text editor*
3. Bahasa JavaScript digunakan pada *frontend* and *backend*
4. React.js digunakan sebagai *frontend* framework bersama dengan react-admin
5. Mozilla Firefox Developer 79.0b7 sebagai *web browser*
6. Node.js sebagai *server environment backend*
7. Postman untuk menguji API
8. Twillio sebagai *SMS Service*
9. Mailtrap untuk menguji *email*
   1. **Implementasi Perangkat Keras Pembangun**

Sebagai alat untuk mengembangkan aplikasi ini, berikut merupakan daftar perangkat keras yang digunakan selama pengerjaan aplikasi yang meliputi:

1. Intel Core i3-6006U
2. HDD 500GB
3. RAM 4GB
4. Layar Monitor 15’’
5. *Keyboard*
6. *Mouse*
   1. **Implementasi Basis Data**

Untuk menyimpan data yang disimpan, digunakanlah basis data yang pada aplikasi ini menggunakan basis data MongoDB. MongoDB merupakan basis data dengan model NoSQL sehingga tidak memerlukan skema yang ditentukanPada bagian awal, perancangan aplikasi. Walaupun begitu, untuk mempermudah pengimplementasian aplikasi maka penulis menggunakan *library* mongoose.js untuk menghubungkan *backend* dengan *database* dan mongoose membutuhkan pembuatan skema dalam bentuk model.

Model dalam mongoose merupakan pembungkus untuk skema. Skema menentukan struktur, validasi dan nilai standar sedangkan model memberikan antarmuka untuk memanipulasi data.

Berikut merupakan implementasi basis data berdasarkan rancangan basis data:

1. *year\_report*

|  |
| --- |
| **const** BoolQuestionSchema = **new** Schema({  information: {type: Boolean, required: true, default: false}, },{ \_id : false })  **const** TextQuestionSchema = **new** Schema({  information: {type: String, required: true, default:null}, },{ \_id : false })  **const** DuoBoolQuestionSchema = **new** Schema({  a: BoolQuestionSchema,  b: BoolQuestionSchema, },{ \_id : false })  **const** TrioBoolQuestionSchema = **new** Schema({  a: BoolQuestionSchema,  b: BoolQuestionSchema,  c: BoolQuestionSchema, },{ \_id : false })  **const** OneBoolTwoTextQuestionSchema = **new** Schema({  a: BoolQuestionSchema,  b: TextQuestionSchema,  c: TextQuestionSchema, },{ \_id : false })  **const** OneBoolOneTextQuestionSchema = **new** Schema({  a: BoolQuestionSchema,  b: TextQuestionSchema, },{ \_id : false })  **const** reportGroupSchema = **new** Schema({  question1: TrioBoolQuestionSchema,  question2: TrioBoolQuestionSchema,  question3: {  a: BoolQuestionSchema,  b: BoolQuestionSchema,  c: BoolQuestionSchema,  d: BoolQuestionSchema,  },  question4: TrioBoolQuestionSchema,  question5: TrioBoolQuestionSchema,  question6: DuoBoolQuestionSchema,  question7: TrioBoolQuestionSchema,  question8: TrioBoolQuestionSchema,  question9: BoolQuestionSchema,  question10: {  a: BoolQuestionSchema,  b: OneBoolTwoTextQuestionSchema,  c: DuoBoolQuestionSchema,  },  question11: OneBoolOneTextQuestionSchema,  },{ \_id : false })  **const** ReportYearSchema = **new** Schema({    author: {type: Schema.Types.ObjectId, ref:'User', required: true},  institution:{type: Schema.Types.ObjectId, ref:'Institution', required: true},  total:{type: Number, required: true},  year: {type: Date},  report: reportGroupSchema, }) |

Potongan kode diatas merupakan pembuatan skema ReportYearSchema, yang terdiri dari *key* author, institution, total, year dan report. *Key* author mereferensikan ObjectId skema User, hal serupa ada pada *key* institution yang mereferensikan ObjectId Institution. *Key* total memiliki tipe data Number sedangkan report memiliki tipe data Date. *Key* report mereferensikan skema reportGroupSchema, yang terdiri dari berbagai subskema yang diatur untuk tidak mengadakan ObjectId secara otomatis. Subskema yang ada pada dasarnya akan mereferensikan dua skema dasar yaitu BoolQuestionSchema dan TextQuestionSchema. Kedua skema tersebut juga diatur untuk tidak mengadakan ObjectId secara otomatis. BoolQuestionSchema memiliki tipe data Boolean, diharuskan untuk ada dan memiliki nilai false secara otomatis apabila tidak diisi. TextQuestionSchema memiliki tipe data String, diharuskan untuk ada dan memiliki nilai null secara otomatis apabila tidak diisi.

1. *semester\_report*

|  |
| --- |
| **const** questionSchema = **new** Schema({  total: {type: Number, required: true},  detail: {type: String, max: 300}, },{ \_id : false })  **const** reportGroupSchema = **new** Schema({  question1: questionSchema,  question2: questionSchema,  question3: questionSchema,  question4: questionSchema,  question5: questionSchema,  question6: questionSchema,  question7: questionSchema,  question8: questionSchema, },{ \_id : false })  **const** ReportSemesterSchema = **new** Schema({    author: {type: Schema.Types.ObjectId, ref:'User', required: true},  month: {type: String, required:true, enum:["Januari","Februari","Maret","April","Mei","Juni","Juli","Agustus","September","Oktober","November","Desember"], default: 'Januari'},  year: {type: Date, default: Date.now},  institution:{type: Schema.Types.ObjectId, ref:'Institution', required: true},  report: reportGroupSchema , }) |

Potongan kode diatas merupakan pembuatan skema ReportSemesterSchema, yang terdiri dari *key* author, month, year, institution dan report. *Key* author mereferensikan ObjectId skema User, hal serupa ada pada *key* institution yang mereferensikan ObjectId Institution. *Key* month memiliki tipe data String dengan pilihan sedangkan year memiliki tipe data Date. *Key* report mereferensikan skema reportGroupSchema, yang terdiri dari berbagai subskema yang diatur untuk tidak mengadakan ObjectId secara otomatis. Subskema yang ada pada dasarnya akan mereferensikan skema dasar yaitu questionSchema. questionSchema diatur untuk tidak mengadakan ObjectId secara otomatis. questionSchema memiliki *key* total dengan tipe data Number dan diharuskan untuk ada sedangkan *key* detail memiliki tipe data String, dan maksimum huruf sejumlah 300.

1. *user*

|  |
| --- |
| **const** UserSchema = **new** Schema({    username:{type: String, required: true, unique:true},  first\_name:{type: String, required: true},  full\_name:{type: String, required: true},  email:{type: String, required: true},  phonenumber:{type: Number, required: true},  privilege:{type: String, required:true, enum:['Admin','User'], default: 'User'},  job\_title:{type: String, required:true},  nip: {type:String, default: null},  password:{type: String, required: true, min:8} }) |

Potongan kode diatas merupakan pembuatan skema UserSchema, yang terdiri dari *key* username, first\_name, full\_name, email, phonenumber, privilege, job\_title, nip dan password. *Key* username bertipe data String, diharuskan untuk diisi dan harus unik. *Key* first\_name, full\_name, email, job\_title, nip, dan password bertipe data String dan diharuskan untuk diisi dan khusus untuk *key* password minimum panjang kata adalah 8 huruf. *Key* phonenumber bertipe data Number dan juga harus diisi sedangkan privilege berupa String dengan pilihan yang secara otomatis diisi data “User”.

1. *notification*

|  |
| --- |
| **const** NotificationSchema = **new** Schema({    complete\_status: {type: Boolean, default:false},  remindee: {type: Schema.Types.ObjectId, ref:'User', required: true},  remind\_date: {type: Date, required: true},  report\_type: {type: String, required:true, enum:['yearly','semesterly']} }) |

Potongan kode diatas merupakan pembuatan skema NotificationSchema, yang terdiri dari *key* complete\_status, remindee, remind\_date dan report\_type. *Key* complete\_status bertipe data Boolean dan otomatis terisi false dan remindee bertipe data ObjectId yang mereferensikan skema User dan diharuskan untuk diisi. *Key* reminde\_date bertipe data Date dan diharuskan untuk diisi. *Key* report\_type bertipe data String dan juga harus diisi dengan pilihan.

1. *institution*

|  |
| --- |
| **const** InstitutionSchema = **new** Schema({    name: {type: String, required: true, max: 100},  address: {type: String, required: true, max: 300},  area: {type: Number, required: true},  city: {type: String, required: true, max: 25},  province: {type: String, required: true, max: 25}, }) |

Potongan kode diatas merupakan pembuatan skema InstitutionSchema, yang terdiri dari *key* name, address, area, city dan province. *Key* name, address, city dan province bertipe data String dan diharuskan untuk diisi. *Key* area bertipe data number dan juga harus diisi.

* 1. **Implementasi REST API**

Dikarenakan arsitektur aplikasi berupa sistem terdistribusi *client - server* maka digunakan API untuk menyampaikan informasi antar *client* dengan *server*. Untuk membuka *server* maka diperlukan *endpoint* yang terbatas pada beberapa hal yang mengacu pada perancangan pada tahap sebelumnya.

Berikut merupakan implementasi REST API:

1. *Create*

|  |
| --- |
| exports.create = [  body('username', 'Username Error').trim().isLength({min:1}),  body('first\_name', 'Test').trim().isLength({min:1}),  body('full\_name').trim().isLength({min:1}),  body('email', 'Email Error').trim().isEmail(),  body('phonenumber','Phone Number Error').trim().isNumeric(),  body('privilege',' Privilege Error'),  body('jobtitle','Job Title Error').trim().escape().isLength({min:1}),  body('nip','NIP Error').trim().escape(),  body('password','Password Error').trim().isLength({min:1}),    **async** (req, res, next) => {  **const** error = validationResult(req);  **if**(!error.isEmpty()){  console.log(error)  **throw** **new** Error(error[0].msg)  }**else**{   **const** hashedPassword = **await** bcrypt.hash(req.body.password,10)   **const** user = **new** User({  username: req.body.username,  first\_name: req.body.first\_name,  full\_name: req.body.full\_name,  email: req.body.email,  phonenumber: req.body.phonenumber,  privilege: req.body.privilege,  jobtitle: req.body.jobtitle,  nip: req.body.nip,  password : hashedPassword  })  user.save(**function** (err, results) {  **if**(err){**return** next(err);}  res.send(results);  })   }  } ] |

Potongan kode diatas merupakan pembuatan endpoint REST API create, yang berfungsi untuk membuat data. Fungsi merupakan *array*, dimana pada tahap awal melakukan sanitasi dan validasi. Kemudian dilanjutkan dengan pengecekan *error* dan khusus pada data yang digunakan sebagai contoh yaitu User melakukan *hashing* pada masukan password. Pada data lain, setelah melakukan pengecekan *error* akan menuju tahapan pembuatan data dengan masukan yang telah di sanitasi dan validasi. Data tersebut kemudian akan disimpan dan hasil data dikirim ke klien untuk membentuk tampilan baru.

1. *Read*

|  |
| --- |
| **const** handleFilter = (filter) => {  **const** filterJson = JSON.parse(filter)  **return** filterJson;  }  **const** handleRange = (range) => {  **const** rangeJson = JSON.parse(range)  **const** [start, end] = rangeJson;  **const** limitation = end-start+1;  **return** [start,limitation]; }  **const** handleSort = (sort) => {  **const** sortJson = JSON.parse(sort)  **const** [resource, order] = sortJson;  **const** orderLowerCase = order.toLowerCase()  **return** sortOn={  [resource]: [orderLowerCase]  } } |

Potongan kode diatas merupakan beberapa fungsi pembantu, yang berfungsi untuk mengecek *query* pada *request* yang diterima. Fungsi handleFilter melakukan *parsing* ke bentuk *JSON* dan kemudian mengirim kembali hasilnya. Fungsi handleRange juga melakukan *parsing* ke bentuk *JSON*, setelah *parsing* kemudian melakukan *array destructuring* untuk menghasilkan konstan start dan end. Konstan start dan end digunakan untuk menghitung limitation. handleFilter kemudian mengembalikan start dan limitation dalam bentuk *array*. Fungsi handleSort elakukan *parsing* ke bentuk *JSON*, setelah *parsing* kemudian melakukan *array destructuring* untuk menghasilkan konstan resource dan order. Konstan order kemudian diubah menjadi huruf kecil dan disimpan pada konstan orderToLowerCase. Hasil yang dikembalikan adalah dalam bentuk *object* dengan *key* resource dan value orderToLowerCase.

|  |
| --- |
| exports.show\_all = **async** (req, res, next) => {  **if**(Object.keys(req.query).length === 0){  Notification.find({}).exec(  (err, results) =>{  **if**(err){**return** next(err);}  res.json(results);  }  )   }**else**{  **let** filter={};  **let** start, limitation;  **let** sort;  **if**(req.query.range != undefined){  **const** range = **await** handleRange(req.query.range);  [start, limitation] = range;  }  **if**(req.query.filter != undefined){  filter = **await** handleFilter(req.query.filter);  }  **if**(req.query.sort != undefined) {  sort = **await** handleSort(req.query.sort);  }  Notification.find(filter).sort(sort).skip(start).limit(limitation).exec(  (err, results) =>{  res.json(results)  })  } } |

Potongan kode diatas merupakan pembuatan endpoint REST API show\_all, yang berfungsi untuk memanggil data dari *database* dan mengirimnya ke klien sesuai dengan *request* yang diterima. Pada bagian awal, fungsi akan mengecek apakan terdapat *query* pada *request*. Apabila tidak terdapat *query* maka akan melakukan pencarian pada *database* dengan normal. Apabila terdapat *query* maka akan melakukan serangkaian cek, yaitu apakan terdapat *query* range, atau *query* filter, atau *query* sort. Apabila cek berhasil maka akan memanggil fungsi bantuan dan apabila tidak akan lanjut ke bagian selanjutnya. Setelah mengisi variabel dengan data dari fungsi pembantu maka akan melakukan pencarian pada *database*. Pencarian dilakukan berdasarkan variabel yang diisi dan hasilnya akan dikirim ke klien untuk membentuk tampilan baru

|  |
| --- |
| exports.show\_one = (req, res, next) => {  Notification.findById(req.params.id).exec(  (err, results) =>{  **if**(err){**return** next(err);}  res.json(results);  }  ) } |

Potongan kode diatas merupakan pembuatan endpoint REST API show\_one, yang berfungsi untuk memanggil data tunggal dari *database* berdasarkan parameter dan mengirimnya ke klien sesuai dengan *request* yang diterima. Pada bagian awal, fungsi akan melakukan pencarian pada *database* berdasarkan parameter yang diterima dan hasilnya akan dikirim ke klien untuk membentuk tampilan baru

|  |
| --- |
| exports.show\_ten = **async** (req, res, next) => {  **let** filter={};  **if**(req.query.filter != undefined){  filter = **await** handleFilter(req.query.filter);  }   **async**.parallel({  data: (callback) => {  Notification.find(filter).sort('year +1').limit(10).exec(callback)  },  count: (callback) => {  Notification.countDocuments(filter).exec(callback)  }  },(err, results) => {  **if**(err){**return** next(err);}  res.json(results);  }) } |

Potongan kode diatas merupakan pembuatan endpoint REST API show\_ten, yang berfungsi untuk memanggil data sejumlah sepuluh dari *database* dan mengirimnya ke klien sesuai dengan *request* yang diterima. Pada bagian awal, fungsi akan melakukan pengecekan apakah terdapat *query* filter, yang apabila terdeteksi ada akan mengisi variabel filter dari hasil fungsi pembantu. Tahap selanjutnya adalah menggunakan *module* async untuk membantu *asynchronization*. Fungsi yang dipanggil pertama kali adalah pencarian data yang dibatasi sejumlah sepuluh dan diurutkan berdasarkan tahun terbaru khusus pada data Notification. Fungsi yang dipanggil selanjutnya adalah menghitung jumlah data yang ada pada suatu *collection* di *database*. Hasil kedua fungsi tersebut akan berada pada variabel results pada bagian selanjutnya. Hasil dari pencarian akan dikirim ke klien untuk membentuk tampilan.

1. *Update*

|  |
| --- |
| exports.update = [  body('complete\_status'),  body('remindee'),  body('remind\_date').toDate().optional({ checkFalsy: true }).isISO8601(),  body('report\_type'),  (req, res, next) => {  **const** error = validationResult(req);  **if**(!error.isEmpty()){  **throw** **new** Error("Error : ");  }**else**{  **const** notification = **new** Notification({  \_id: req.params.id,  complete\_status: req.body.complete\_status,  remindee: req.body.remindee,  remind\_date: req.body.remind\_date,  report\_type: req.body.report\_type,  })  Notification.findByIdAndUpdate(req.params.id, notification, (err, results) =>{  **if**(err){**return** (next(err));}  res.send(results);  })  }  } ] |

Potongan kode diatas merupakan pembuatan endpoint REST API update, yang berfungsi untuk mengubah data dari *database* sesuai masukan dan parameter dari kliendan mengirim hasil pengubahan ke klien sesuai dengan *request* yang diterima. Fungsi merupakan *array*, dimana pada tahap awal melakukan sanitasi dan validasi. Kemudian dilanjutkan dengan pengecekan *error* . Setelah melakukan pengecekan *error* akan menuju tahapan pembuatan data dengan masukan yang telah di sanitasi dan validasi serta parameter. Data tersebut sebelum akan disimpan akan melakukan pencarian pada data di *database* sesuai dengan parameter yang diberikan. Setelah data telah ditemukan data disimpan dan hasil data dikirim ke klien untuk membentuk tampilan baru.

1. *Delete*

|  |
| --- |
| exports.delete = (req, res, next) => {  Notification.findByIdAndRemove(req.params.id).exec((err,results) =>{  **if**(err){**return** next(err);}  res.json(results);  }) } |

Potongan kode diatas merupakan pembuatan endpoint REST API delete, yang berfungsi untuk menghapus data tunggal dari *database* berdasarkan parameter dan mengirim hasil ke klien sesuai dengan *request* yang diterima. Pada bagian awal, fungsi akan melakukan pencarian pada *database* berdasarkan parameter yang diterima dan melakukan penghapusan pada data di *database*. Hasilnya akan dikirim ke klien untuk membentuk tampilan baru.

1. *Login*

|  |
| --- |
| exports.login = **function** (req, res, next) {  passport.authenticate('local', {session: false}, (err, user, info) => {  **if** (err || !user) {  **return** res.status(400).json({  message: 'Something is not right',  user : user  });  }  req.login(user, {session: false}, (err) => {  **if** (err) {  res.send(err);  }   **const** token = jwt.sign(user.toJSON(), `${process.env.JWT\_SECRET}`);  **return** res.json(token);  });  })(req, res, next); } |

Potongan kode diatas merupakan pembuatan endpoint REST API login, yang berfungsi untuk autentikasi pengguna berdasarkan data dari *database*. Untuk mempermudah autentikasi pengguna, digunakanlah *module* passport, passport-local, passport-jwt dan jsonwebtoken. Pada bagian awal, fungsi akan memanggil fungsi authenticate dari passport untuk melakukan pengecekan pada *database* berdasarkan pada masukan yang berupa username dan password. Tahapan selanjutnya adalah pengecekan *error* dan ada tidaknya *user* berdasarkan masukan yang diterima, apabila data tidak ditemukan atau terjadi *error* maka akan mengirim informasi *error*. Apabila data ditemukan akan melakukan fungsi login, dan hasilnya berupa variabel user diubah menjadi json dan dibungkus menjadi jsonwebtoken bersama dengan *environment variable* JWT\_SECRET menjadi konstan token. Konstan token kemudian dikirim ke klien untuk otorisasi dan penggunaan lainnya pada klien.

1. *Notification*

|  |
| --- |
| exports.create = [  body('complete\_status'),  body('remindee'),  body('remind\_date').toDate().optional({ checkFalsy: true }).isISO8601(),  body('report\_type'),    (req, res, next) => {  **const** error = validationResult(req);   **if**(!error.isEmpty()){  debug(error)  **throw** **new** Error();  }**else**{  **const** notification = **new** Notification({  complete\_status: req.body.complete\_status,  remindee: req.body.remindee,  remind\_date: req.body.remind\_date,  report\_type: req.body.report\_type,  })  Notification.create(notification, **async** (err, results) =>{  **if**(err){**return** next(err);}  Notification.findById(results.\_id).populate('remindee').exec(**async** (err, data) => {  **if**(err){**return** next(err);}  **await** cronjob.startSchedule(data);  })  res.send(results);  })  }  } ] |

Potongan kode diatas merupakan pembuatan endpoint REST API create Notification, yang berfungsi untuk membuat data dan melakukan *scheduling* menggunakan cron. Fungsi merupakan *array*, dimana pada tahap awal melakukan sanitasi dan validasi. Kemudian dilanjutkan dengan pengecekan *error* dan khusus pada data yang digunakan sebagai contoh yaitu User melakukan *hashing* pada masukan password. Pada data lain, setelah melakukan pengecekan *error* akan menuju tahapan pembuatan data dengan masukan yang telah di sanitasi dan validasi. Data tersebut kemudian akan disimpan dan hasil data dikirim ke klien untuk membentuk tampilan baru. Sebelum melakukan pengiriman data ke klien, dilakukan pencarian berdasarkan \_id dari data yang baru saja disimpan, hasil dari pencarian akan digunakan sebagai parameter fungsi startSchedule dari cronjob.

|  |
| --- |
| exports.startSchedule = ( results ) => {  **let** date = results.remind\_date;  **const** job = **new** CronJob(date,**async** **function**() {  **await** email\_controller.sendEmail(results);  **await** sms.sendSmsReminder();  **await** notification\_controller.setCompletionToTrue(results.\_id);  }, null, true, 'Asia/Jakarta');  job.start(); } |

Potongan kode diatas merupakan fungsi startSchedule pada module cronJob, yang berfungsi untuk membuat cronjob dengan bantuan *module* cron. Hasil dari pencarian dari fungsi sebelumnya ditaruh pada variable results. *Property* remind\_date kemudian dimasukkan pada variabel date. Langkah selanjutnya merupakan pembuatan CronJob dengan nama job. CronJob diatur untuk menjalankan fungsi di dalamnya pada tanggal yang ada di variable date. Fungsi yang dijalankan adalah pengiriman email dengan parameter results, pengiriman sms, serta pengubahan complete\_status pada data dokumen Notification menjadi true.

|  |
| --- |
| exports.setCompletionToTrue = (req, res, next) => {  Notification.findById(req).exec(**async** (err, results) => {  results.complete\_status = true  **await** results.save()  })  } |

Potongan kode diatas merupakan fungsi setCompletionToTrue, yang berfungsi untuk mengubah complete\_status dari suatu dokumen pada koleksi Notification menjadi true. Setelah pemanggilan, fungsi akan melakukan pencarian berdasarkan dari *request* yang diterima. Setelah data ditemukan, *value* dari *key* complete\_status akan diubah menjadi true. Setelah perubahan, data baru akan disimpan.

|  |
| --- |
| exports.sendSmsReminder = (req, res, next) => {  client.messages  .create({  body: 'Reminder - Laporan SMK3 anda belum diajukan ke sistem, check email atau situs Laporan SMK3.',  from: '+12019928783',  to: '+6281xxxxxxxxx'  }) |

Potongan kode diatas merupakan fungsi sendSmsReminder, yang berfungsi untuk mengirim SMS ke nomor yang dituju. Setelah pemanggilan, fungsi akan membuat pesan dengan isi pada *key* body, nomor kirim pada *key* from, dan nomor tujuan pada *key* to. Setelah pesan dibuat, SMS akan dikirim sesuai dari data yang diatur.

|  |
| --- |
| exports.sendEmail = (results) => {  nodemailer.createTestAccount( **async** (err, account) => {  **if** (err) {  console.error('Failed to create a testing account. ' + err.message);  **return** process.exit(1);  }  **var** transporter = nodemailer.createTransport({  host: "smtp.mailtrap.io",  port: 2525,  auth: {  user: "a780fxxx965xxx",  pass: "ba3b1xxx87axxx"  }  });  **const** data = **await** ejs.renderFile(\_\_dirname + "/emailTemplate.ejs", { results: results })  **let** message = {  from: 'Sender Name <sender@example.com>',  to: `Recipient ${results.remindee.email}`,  subject: 'Remider to file SM3 report',  text: 'Hello to myself!',  html: data  };  transporter.sendMail(message, (err, info) => {  **if** (err) {  console.log('Error occurred. ' + err.message);  **return** process.exit(1);  }  console.log("Email Sent!")  });  }); } |

Potongan kode diatas merupakan fungsi sendEmail, yang berfungsi untuk mengirim email ke alamat email yang dituju. Untuk mempermudah pengiriman email, *module* nodemailer digunakan. Setelah pemanggilan, fungsi akan mencoba membuat akun. Apabila pembuatan akun tidak berhasil, akan mengirim pesan *error* pada *console*. Apabila berhasil, tahapan selanjutnya adalah pembuatan *transporter* menggunakan fungsi dari nodemailer. Fungsi akan membuat konstan data yang berisi *template* email, dan membuat pesan email berdasarkan data yang diterima. Setelah pesan email selesai dibuat, pesan email dikirim menggunakan *transporter* ke tujuan yang diatur.

* 1. **Implementasi Antarmuka Pengguna**

Untuk mempermudah pembuatan tampilan antarmuka, digunakan *framework* react-admin. react-admin merupakan *framework* yang berisi komponen React yang siap pakai yang ditujukan untuk membuat admin *panel*. Pada tahap ini, dijelaskan implementasi antarmuka dari aplikasi yang berupa gambar dan potongan kode.

* + 1. **Implementasi Dashboard**

|  |
| --- |
| **export** **const** Dashboard = () => {  **const** classes = useStyles();  **const** { permissions } = usePermissions();  **const** username = localStorage.getItem('username')  **return** (  <div className={classes.root}>  <Title title="SMK3" />  {permissions === "Admin" ?  <Grid container spacing={3}>  <Grid item xs={12}>  <Paper className={classes.paper}> Welcome {username}</Paper>  </Grid>  <Grid item xs={6}>  <ReportYearCard />  </Grid>  <Grid item xs={6}>  <ReportSemesterCard />  </Grid>  <Grid item xs={6}>  <NotificationCard />  </Grid>  <Grid item xs={3}>  <InstitutionCard />  </Grid>  <Grid item xs={3}>  <UserCard />  </Grid>  </Grid>  : null}  {permissions === 'User' &&   <Grid container spacing={3} style={{marginTop: 0.1}}>  <Grid item xs={12}>  <Paper className={classes.paper}> Welcome {username}</Paper>  </Grid>  <Grid item xs={6}>  <ReportStatusCard />  </Grid>  <Grid item xs={3}>  <ReportYearCardUser />  </Grid>  <Grid item xs={3}>  <ReportSemesterCardUser />  </Grid>  </Grid>  }  </div>  ); } |

Potongan kode diatas kode dari tampilan dashboard, dashboard dibagi menjadi 2, bagian untuk admin dan bagian untuk user. Hal ini dicapai dengan memanggil *hook* usePermission. Terdapat komponen title yang digunakan untuk memberikan judul pada halaman. Untuk mengisi halaman dashboard digunakan komponen Card yang menampilkan data baru. Setiap komponen Card diatur penempatannya menggunakan komponen Grid.

|  |
| --- |
| **const** InstitutionTable = (props) => {  **const** classes = useStyles();  **return** (  <Table className={classes.table} size="small" aria-label="simple table">  <TableHead>  <TableRow>  <TableCell className={classes.bold}>Nama</TableCell>  </TableRow>  </TableHead>  <TableBody>  {props.data.map(item => (  <TableRow key={item.\_id}>  <TableCell component="th" scope="item">  {item.name}  </TableCell>  </TableRow>  ))}  </TableBody>  </Table>  ) }  **export** **const** InstitutionCard = () => {  **const** [data, setData] = useState([]);  useEffect(() => {  **const** resource = "institution"  **const** fetchData = **async** () => {  **const** result = **await** fetch(`http://192.168.100.62:9000/${resource}`)  **const** json = **await** result.json();  setData(json);  }  fetchData();  }, []);  **return** (  <CardWithIcon icon={LocalHospitalIcon} link="#/institution" bgcolor="#f44336" name="Fasyankes" data={<InstitutionTable data={data}/>} length={data.length}/>  ) } |

Potongan kode diatas merupakan salah satu dari komponen Card yaitu InstitutionCard. Komponen memanggil data ke server yang dalam komponen ini data Institution, data yang telah diterima ditampilkan menggunakan komponen CardWithIcon untuk menampilkan data ringkasan. InstitutionTable digunakan untuk menampilkan daftar data yang diterima. Ikon komponen dapat diatur pada icon, untuk pengalihan *URL* ketika di klik di set pada atribut link. Warna latar ikon diubah pada bgcolor dan nama yang ditampilkan pada Card diberikan pada atribut name. Untuk menampilkan tabel daftar data, InstitutionTable menjadi isi atribut data yang juga menyampaikan data yang diperlukan untuk ditampilkan

|  |
| --- |
| **const** CardWithIcon = (props) => {  **const** classes = useStyles();  **return** (  <div className={classes.main}>  <CardIcon Icon={props.icon} bgColor={props.bgcolor} link={props.link}/>  <Card className={classes.card}>  <CardContent className={classes.padding}>  <Typography className={classes.title} color="textSecondary">  {props.name}  </Typography>  <Typography variant="h4" component="h4">  {props.length}  </Typography>  </CardContent>  {props.data}  </Card>  </div>  );  };  export default CardWithIcon; |

Potongan kode diatas merupakan kode komponen CardWithIcon. Komponen berfungsi untuk menampilkan bagian atas pada komponen Card.

|  |
| --- |
| **const** CardIcon = ({ Icon, bgColor, link }) => {  **const** classes = useStyles();  **return** (  <Card className={classes.card} style={{ backgroundColor: bgColor }}>  <a href={link}><Icon className={classes.icon} /></a>  </Card>  ); }; |

Potongan kode diatas merupakan kode komponen CardIcon. Komponen berfungsi untuk mengatur letak ikon pada Card.

* + 1. **Implementasi Halaman CRUD**

|  |
| --- |
| **export** **const** InstitutionList = props => (  <List title="Fasyankes" {...props} bulkActionButtons={false}>  <Datagrid rowClick="show">  <TextField source="name" label="Nama Fasyankes"/>  <TextField source="address" label="Alamat"/>  <TextField source="city" label="Kabupaten/Kota"/>  <TextField source="province" label="Provinsi"/>  <NumberField source="area" label="Luas(m2)"/>  <EditButton />  <DeleteButton />  </Datagrid>  </List> ); |

Potongan kode diatas merupakan kode komponen InstitutionList. Komponen berfungsi untuk menampilkan daftar data yang diterima setelah melakukan *request*. Atribut title pada elemen List digunakan untuk mengubah judul tampilan. Elemen Datagrid berfungsi sebagai *iterator* elemen didalamnya, dimana Datagrid akan memanggil elemen didalamnya sebanyak data yang disampaikan dari elemen List. Atribut rowClick digunakan untuk mengatur perilaku komponen ketika di klik. Elemen TextField berguna untuk menampilkan data berupa tipe data String, NumberField untuk tipe data Number. EditButton dan DeleteButton masing-masing untuk melakukan operasi pengubahan dan penghapusan data.

|  |
| --- |
| **export** **const** InstitutionShow = props => (  <Show title={<PageTitle action="Show"/>} {...props}>  <SimpleShowLayout>  <TextField source="name" />  <TextField source="address" />  <TextField source="city" />  <TextField source="province" />  <NumberField source="area" />  </SimpleShowLayout>  </Show> ); |

Potongan kode diatas merupakan kode komponen InstitutionShow. Komponen berfungsi untuk menampilkan detail data tunggal yang diterima setelah melakukan *request*. Atribut title pada elemen List digunakan untuk mengubah judul tampilan. Elemen SimpleShowLayout berfungsi sebagai *layout* elemen didalamnya, dimana SimpleShowLayout akan memanggil elemen didalamnya secara baris per baris dan dengan data yang disampaikan dari elemen Show.. Elemen TextField berguna untuk menampilkan data berupa tipe data String, NumberField untuk tipe data Number.

|  |
| --- |
| **export** **const** InstitutionEdit = props => (  <Edit title={<PageTitle action="Edit"/>} {...props}>  <SimpleForm>  <TextInput source="name" />  <TextInput source="address" />  <TextInput source="city" />  <TextInput source="province" />  <NumberInput source="area" />  </SimpleForm>  </Edit> ); |

Potongan kode diatas merupakan kode komponen InstitutionEdit. Komponen berfungsi untuk menampilkan tampilan pengubahan data tunggal yang diterima setelah melakukan *request*. Atribut title pada elemen Edit digunakan untuk mengubah judul tampilan. Elemen SimpleForm berfungsi sebagai *layout* dan *rendering* elemen didalamnya, dimana SimpleForm akan memanggil elemen didalamnya secara baris per baris dan dengan data yang disampaikan dari elemen Edit. SimpleForm juga menangani data yang dikirim melalui *form*. Elemen TextInput berguna untuk menampilkan data berupa tipe data String dan mengharapkan data masukan String, NumberField untuk tipe data Number mengharapkan data masukan Number.

|  |
| --- |
| **export** **const** InstitutionCreate = props => (  <Create title={<PageTitle action="Create"/>} {...props}>  <SimpleForm>  <TextInput source="name" />  <TextInput source="address" />  <TextInput source="city" />  <TextInput source="province" />  <NumberInput source="area" />  </SimpleForm>  </Create> ); |

Potongan kode diatas merupakan kode komponen InstitutionCreate. Komponen berfungsi untuk menampilkan tampilan pembuatan data tunggal. Atribut title pada elemen Create digunakan untuk mengubah judul tampilan. Elemen SimpleForm berfungsi sebagai *layout* dan *rendering* elemen didalamnya, dimana SimpleForm akan memanggil elemen didalamnya secara baris per baris dan dengan data yang disampaikan dari elemen Edit. SimpleForm juga menangani data yang dikirim melalui *form*. Elemen TextInput berguna untuk menampilkan data berupa tipe data String dan mengharapkan data masukan String, NumberField untuk tipe data Number mengharapkan data masukan Number.

### **BAB VI**

**PENGUJIAN DAN PEMBAHASAN**

Dengan implementasi aplikasi telah mencapai kondisi yang memuaskan, tahap selanjutnya adalah pengujian dan pembahasan yang akan dijelaskan pada bab ini. Pada bab ini aplikasi akan diuji apakah sudah memenuhi rancangan awal yang dibuat dan performa ketika melakukan operasinya. Pengujian ini difokuskan ke fungsionalitas aplikasi yaitu REST API dan antarmuka.

* 1. **Skenario Pengujian REST API**

|  |  |
| --- | --- |
| **Pengujian** | **Materi** |
| Melakukan proses *login* pada | Mengirim *request* API *login* dengan masukan *username* dan *password* |

|  |  |
| --- | --- |
| **Pengujian** | **Materi** |
| Melakukan proses *get all data* pada *collection reminder* | Mengirim *request* API *get all data* ke *collection reminder* |
| Melakukan proses *get all data* pada *collection year\_report* | Mengirim *request* API *get all data* ke *collection year\_report* |
| Melakukan proses *get all data* pada *collection semester\_report* | Mengirim *request* API *get all data* ke *collection semester\_report* |
| Melakukan proses *get all data* pada *collection user* | Mengirim *request* API *get all data* ke *collection user* |
| Melakukan proses *get all data* pada *collection institution* | Mengirim *request* API *get all data* ke *collection institution* |

|  |  |
| --- | --- |
| **Pengujian** | **Materi** |
| Melakukan proses *get single data* pada *collection reminder* | Mengirim *request* API *get single data* ke *collection reminder* dengan parameter *id reminder* |
| Melakukan proses *get single data* pada *collection year\_report* | Mengirim *request* API *get single data* ke *collection year\_report* dengan parameter *id year\_report* |
| Melakukan proses *get single data* pada *collection semester\_report* | Mengirim *request* API *get single data* ke *collection* semester\_report dengan parameter *id semester\_report* |
| Melakukan proses *get single data* pada *collection user* | Mengirim *request* API *get single data* ke *collection* user dengan parameter *id user* |
| Melakukan proses *get single data* pada *collection institution* | Mengirim *request* API *get single data* ke *collection institution* dengan parameter *id institution* |

|  |  |
| --- | --- |
| **Pengujian** | **Materi** |
| Melakukan proses *create* pada *collection reminder* | Mengirim *request* API *create* ke *collection reminder* |
| Melakukan proses *create* pada *collection year\_report* | Mengirim *request* API *create* ke *collection year\_report* |
| Melakukan proses *create* pada *collection semester\_report* | Mengirim *request* API *create* ke *collection semester\_report* |
| Melakukan proses *create* pada *collection user* | Mengirim *request* API *create* ke *collection user* |
| Melakukan proses *create* pada *collection institution* | Mengirim *request* API *create* ke *collection institution* |

|  |  |
| --- | --- |
| **Pengujian** | **Materi** |
| Melakukan proses *update* pada *collection institution* | Mengirim *request* API *update* ke *collection institution* dengan parameter *id institution* |
| Melakukan proses *delete* pada *collection reminder* | Mengirim *request* API *delete* ke *collection reminder* dengan parameter *id reminder* |
| Melakukan proses *delete* pada *collection year\_report* | Mengirim *request* API *delete* ke *collection year\_report* dengan parameter *id year\_report* |
| Melakukan proses *delete* pada *collection semester\_year* | Mengirim *request* API *delete* ke *collection semester\_report* dengan parameter *id semester\_year* |
| Melakukan proses *delete* pada *collection user* | Mengirim *request* API *delete* ke *collection user* dengan parameter *id user* |
| Melakukan proses *delete* pada *collection institution* | Mengirim *request* API *delete* ke *collection institution* dengan parameter *id institution* |

|  |  |
| --- | --- |
| **Pengujian** | **Materi** |
| Melakukan proses *update* pada *collection reminder* | Mengirim *request* API *update* ke *collection reminder* dengan parameter *id reminder* |
| Melakukan proses *update* pada *collection year\_report* | Mengirim *request* API *update* ke *collection year\_report*  dengan parameter *id year\_report* |
| Melakukan proses *update* pada *collection semester\_report* | Mengirim *request* API *update* ke *collection semester\_report* dengan parameter *id semester\_report* |
| Melakukan proses *update* pada *collection user* | Mengirim *request* API *update* ke *collection user* dengan parameter *id user* |
| Melakukan proses *update* pada *collection notification* | Mengirim *request* API *update* ke *collection notification* dengan parameter *id notification* |

|  |  |
| --- | --- |
| **Pengujian** | **Materi** |
| Melakukan proses *get ten data* pada *collection reminder* | Mengirim *request* API *get ten data* ke *collection reminder* |
| Melakukan proses *get ten data* pada *collection year\_report* | Mengirim *request* API *get ten data* ke *collection year\_report* |
| Melakukan proses *get ten data* pada *collection semester\_report* | Mengirim *request* API *get ten data* ke *collection semester\_report* |
| Melakukan proses *get ten data* pada *collection user* | Mengirim *request* API *get ten data* ke *collection user* |
| Melakukan proses *get ten data* pada *collection institution* | Mengirim *request* API *get ten data* ke *collection institution* |

* 1. **Skenario Pengujian antarmuka**
     1. **Create**

|  |  |  |
| --- | --- | --- |
| **Pengujian** | **Materi** | **Hal yang diharapkan** |
| Halaman pembuatan *notification* | Mengakses URL pembuatan *notification* | Menampilkan halaman pembuatan *notification* beserta formulir yang dibutuhkan |
| Halaman pembuatan *user* | Mengakses URL pembuatan *user* | Menampilkan halaman pembuatan *user* beserta formulir yang dibutuhkan |
| Halaman pembuatan *report\_year* | Mengakses URL pembuatan *report\_year* | Menampilkan halaman pembuatan *report\_year* beserta formulir yang dibutuhkan |
| Halaman pembuatan *report\_semester* | Mengakses URL pembuatan *report\_semester* | Menampilkan halaman pembuatan *report\_semester* beserta formulir yang dibutuhkan |
| Halaman pembuatan *institution* | Mengakses URL pembuatan *institution* | Menampilkan halaman pembuatan *institution*  beserta formulir yang dibutuhkan |

* + 1. **Get One**

|  |  |  |
| --- | --- | --- |
| **Pengujian** | **Materi** | **Hal yang diharapkan** |
| Halaman detail satu data *institution* | Mengakses URL detail satu data *institution* | Menampilkan halaman detail satu data *institution* beserta data yang dibutuhkan |
| Halaman detail satu data *user* | Mengakses URL detail satu data *use* | Menampilkan halaman detail satu data *user* beserta data yang dibutuhkan |
| Halaman detail satu data *report\_year* | Mengakses URL detail satu data *report\_year* | Menampilkan halaman detail satu data *report\_year* beserta data yang dibutuhkan |
| Halaman detail satu data *report\_semester* | Mengakses URL detail satu data *report\_semester* | Menampilkan halaman detail satu data *report\_semester* beserta data yang dibutuhkan |
| Halaman detail satu data *notification* | Mengakses URL detail satu data *notification* | Menampilkan halaman detail satu data *notification* beserta data yang dibutuhkan |

* + 1. **Get All**

|  |  |  |
| --- | --- | --- |
| **Pengujian** | **Materi** | **Hal yang diharapkan** |
| Halaman daftar *user* | Mengakses URL daftar *user* | Menampilkan halaman daftar *user* beserta data yang dibutuhkan |
| Halaman daftar *report\_semester* | Mengakses URL daftar *report\_semester* | Menampilkan halaman daftar *report\_semester* beserta data yang dibutuhkan |
| Halaman daftar *report\_year* | Mengakses URL daftar *report\_year* | Menampilkan halaman daftar *report\_year* beserta data yang dibutuhkan |
| Halaman daftar *notification* | Mengakses URL daftar *notification* | Menampilkan halaman daftar *notification* beserta data yang dibutuhkan |
| Halaman daftar *institution* | Mengakses URL daftar *institution* | Menampilkan halaman daftar *institution* beserta data yang dibutuhkan |
| Halaman *user profile* | Mengakses URL *user profile* | Menampilkan halaman *user profile* beserta data yang dibutuhkan |
| Halaman *login* | Mengakses URL *login* | Menampilkan halaman *login* |

* + 1. **Update**

|  |  |  |
| --- | --- | --- |
| **Pengujian** | **Materi** | **Hal yang diharapkan** |
| Halaman pengubahan *notification* | Mengakses URL pengubahan *notification* | Menampilkan halaman pengubahan *notification* beserta formulir dan data yang dibutuhkan |
| Halaman pengubahan *user* | Mengakses URL pengubahan *user* | Menampilkan halaman pengubahan *user* beserta formulir dan data yang dibutuhkan |
| Halaman pengubahan *report\_year* | Mengakses URL pengubahan *report\_year* | Menampilkan halaman pengubahan *report\_year* beserta formulir dan data yang dibutuhkan |
| Halaman pengubahan *report\_semester* | Mengakses URL pengubahan *report\_semester* | Menampilkan halaman pengubahan *report\_semester* beserta formulir dan data yang dibutuhkan |
| Halaman pengubahan *institution* | Mengakses URL pengubahan *institution* | Menampilkan halaman pengubahan *institution*  beserta formulir dan data yang dibutuhkan |

* 1. **Hasil Pengujian Aplikasi**
     1. **Hasil Uji Skenario REST API**

Pengujian API dilakukan menggunakan aplikasi Postman. Pada aplikasi ini terdapat tempat penulisan alamat *endpoint* API dan pemilihan *method* yang dikirim. *Method* yang dapat dipilih antara lain seperti GET, PUT, POST, DELETE. Ketika Postman melakukan *request* ke *endpoint*, hasil dari *request* dapat dilihat seperti *status code* yang mengindikasikan kesuksesan *request*.

1. Pengujian Login
2. Pengujian proses get all data
   1. *notification*

|  |
| --- |
| [  {  "complete\_status": false,  "\_id": "5ef5879b44ec2aa3a16e7889",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef587f4e908f2a45ac53c84",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef588172f27cba52809b820",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  } ] |

* 1. *institution*

|  |
| --- |
| [  {  "\_id": "5ef21f1705d9c7c48737a63e",  "name": "Puskesmas Jatinom",  "address": "Jatinom",  "city": "Klaten",  "province": "Jawa Tengah",  "area": 255,  "\_\_v": 0  },  {  "\_id": "5ef4c4d58d3f135b99ed2dd1",  "name": "RSI",  "address": "Jl Tegalyoso",  "city": "Klaten",  "province": "Jawa Tengah",  "area": 456,  "\_\_v": 0  },  {  "\_id": "5ef4c500f82a4461921943d5",  "name": "RSIA",  "address": "Jl Tegalyoso",  "city": "Klaten",  "province": "Jawa Tengah",  "area": 456,  "\_\_v": 0  } ] |

* 1. *report\_year*

|  |
| --- |
| [  {  "\_id": "5ef2a872962f3236526d50b5",  "author": "5efee06f33db112491b532b5",  "total": 15,  "year": "1932-11-07T16:30:00.000Z",  "institution": "5ef21f1705d9c7c48737a63e",  "report": {  "question3": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  },  "d": {  "information": false  }  },  "question10": {  "a": {  "information": false  },  "b": {  "a": {  "information": false  },  "b": {  "information": "5 gas prop"  },  "c": {  "information": "5 tali prop"  }  },  "c": {  "a": {  "information": false  },  "b": {  "information": false  }  }  },  "question1": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": true  }  },  "question2": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question4": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question5": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question6": {  "a": {  "information": false  },  "b": {  "information": false  }  },  "question7": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question8": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question9": {  "information": true  },  "question11": {  "a": {  "information": false  },  "b": {  "information": "false"  }  }  },  "\_\_v": 0  },  {  "\_id": "5ef2a897962f3236526d50b6",  "author": "5efee06f33db112491b532b5",  "total": 15243535,  "year": "1932-11-07T16:30:00.000Z",  "institution": "5ef21f1705d9c7c48737a63e",  "report": {  "question3": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  },  "d": {  "information": false  }  },  "question10": {  "a": {  "information": false  },  "b": {  "a": {  "information": false  },  "b": {  "information": "5 gas prop"  },  "c": {  "information": "5 tali prop"  }  },  "c": {  "a": {  "information": false  },  "b": {  "information": false  }  }  },  "question1": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": true  }  },  "question2": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question4": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question5": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question6": {  "a": {  "information": false  },  "b": {  "information": false  }  },  "question7": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question8": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question9": {  "information": true  },  "question11": {  "a": {  "information": false  },  "b": {  "information": "false"  }  }  } ] |

* 1. *report\_semester*

|  |
| --- |
| [  {  "month": "Maret",  "\_id": "5ef194035c644d612e10ca2e",  "author": "5efee06f33db112491b532b5",  "report": {  "question1": {  "total": 12,  "detail": "neat"  },  "question2": {  "total": 12,  "detail": "neat"  },  "question3": {  "total": 12,  "detail": "neat"  },  "question4": {  "total": 12,  "detail": "neat"  },  "question5": {  "total": 12,  "detail": "neat"  },  "question6": {  "total": 12,  "detail": "neat"  },  "question7": {  "total": 12,  "detail": "neat"  },  "question8": {  "total": 12,  "detail": "neat"  }  },  "\_\_v": 0,  "year": "2020-07-04T07:02:57.542Z"  },  {  "month": "Maret",  "\_id": "5ef1fa214f11ee98b84ab627",  "author": "5efee06f33db112491b532b5",  "report": {  "question1": {  "total": 12,  "detail": "neat"  },  "question2": {  "total": 12,  "detail": "sol"  },  "question3": {  "total": 12,  "detail": "neat"  },  "question4": {  "total": 12,  "detail": "neat"  },  "question5": {  "total": 12,  "detail": "neat"  },  "question6": {  "total": 12,  "detail": "neat"  },  "question7": {  "total": 12,  "detail": "neat"  },  "question8": {  "total": 12,  "detail": "neat"  }  },  "\_\_v": 0,  "year": "2020-07-04T07:02:45.321Z"  } ] |

* 1. *user*

|  |
| --- |
| [  {  "privilege": "Admin",  "nip": "",  "\_id": "5efee06f33db112491b532b5",  "username": "Username",  "first\_name": "Faris",  "full\_name": "Faris Takhassuna Zamzani",  "email": "test@testmail.com",  "phonenumber": 81326346957,  "jobtitle": "Programmer",  "password": "$2a$10$EuIxDMoNGG9MLyn96TeQZ.62FAthPnG1KVgrVtrdtDiZlS.bNtCiW",  "\_\_v": 0  },  {  "privilege": "Admin",  "nip": "1991",  "\_id": "5efef0200846724a7c609393",  "username": "Admin",  "first\_name": "Administrator",  "full\_name": "Administrator",  "email": "admin@admin.mail",  "phonenumber": 1000,  "jobtitle": "Admin",  "password": "$2a$10$W0XN4zZxhqNrCoCjhG8utOQ/.KaKhz.qobz9vMt9iSBhPBaKRI0/K",  "\_\_v": 0  },  {  "privilege": "User",  "nip": "",  "\_id": "5f0030198aa1836302e35b9e",  "username": "userscrub",  "first\_name": "user",  "full\_name": "user and scrub",  "email": "user@mail.com",  "phonenumber": 888551,  "jobtitle": "scrub",  "password": "$2a$10$YOkEvboOSurm.rkFEMikI.ypcti9kFrmg73Q9gLZo8fw7whHvsapS",  "\_\_v": 0  } ] |

1. Pengujian proses get single data
   1. *notification*

|  |
| --- |
| {  "complete\_status": false,  "\_id": "5ef5879b44ec2aa3a16e7889",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0 } |

* 1. *institution*

|  |
| --- |
| {  "\_id": "5ef21f1705d9c7c48737a63e",  "name": "Puskesmas Jatinom",  "address": "Jatinom",  "city": "Klaten",  "province": "Jawa Tengah",  "area": 255,  "\_\_v": 0 } |

* 1. *report\_year*

|  |
| --- |
| {  "\_id": "5ef2a872962f3236526d50b5",  "author": "5efee06f33db112491b532b5",  "total": 15,  "year": "1932-11-07T16:30:00.000Z",  "institution": "5ef21f1705d9c7c48737a63e",  "report": {  "question3": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  },  "d": {  "information": false  }  },  "question10": {  "a": {  "information": false  },  "b": {  "a": {  "information": false  },  "b": {  "information": "5 gas prop"  },  "c": {  "information": "5 tali prop"  }  },  "c": {  "a": {  "information": false  },  "b": {  "information": false  }  }  },  "question1": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": true  }  },  "question2": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question4": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question5": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question6": {  "a": {  "information": false  },  "b": {  "information": false  }  },  "question7": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question8": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question9": {  "information": true  },  "question11": {  "a": {  "information": false  },  "b": {  "information": "false"  }  }  },  "\_\_v": 0 } |

* 1. *report\_semester*

|  |
| --- |
| {  "month": "Maret",  "\_id": "5ef194035c644d612e10ca2e",  "author": "5efee06f33db112491b532b5",  "report": {  "question1": {  "total": 12,  "detail": "neat"  },  "question2": {  "total": 12,  "detail": "neat"  },  "question3": {  "total": 12,  "detail": "neat"  },  "question4": {  "total": 12,  "detail": "neat"  },  "question5": {  "total": 12,  "detail": "neat"  },  "question6": {  "total": 12,  "detail": "neat"  },  "question7": {  "total": 12,  "detail": "neat"  },  "question8": {  "total": 12,  "detail": "neat"  }  },  "\_\_v": 0,  "year": "2020-07-04T07:02:57.542Z" } |

* 1. *user*

|  |
| --- |
| {  "privilege": "Admin",  "nip": "",  "\_id": "5efee06f33db112491b532b5",  "username": "Username",  "first\_name": "Faris",  "full\_name": "Faris Takhassuna Zamzani",  "email": "test@testmail.com",  "phonenumber": 81326346957,  "jobtitle": "Programmer",  "password": "$2a$10$EuIxDMoNGG9MLyn96TeQZ.62FAthPnG1KVgrVtrdtDiZlS.bNtCiW",  "\_\_v": 0 } |

1. Pengujian proses get ten data
   1. *notification*

|  |
| --- |
| {  "data": [  {  "complete\_status": false,  "\_id": "5ef5879b44ec2aa3a16e7889",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef587f4e908f2a45ac53c84",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef588172f27cba52809b820",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef58bbdd36da9a5ec2361c8",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef58be3d36da9a5ec2361c9",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef58c84d36406aef52164ed",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef58ccc063352af6f9cc0aa",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef58d1803e00eb01ae63c62",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef58d8919b6c3b0e67f8b96",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  },  {  "complete\_status": false,  "\_id": "5ef58e4453acf3b2aef3eed7",  "remindee": "5ef4c64ef3b10a62f8d8350a",  "remind\_date": "2020-06-25T00:00:00.000Z",  "report\_type": "yearly",  "\_\_v": 0  }  ],  "count": 64 } |

* 1. *report\_year*

|  |
| --- |
| {  "count": 3,  "data": [  {  "\_id": "5ef2a872962f3236526d50b5",  "author": {  "privilege": "Admin",  "nip": "",  "\_id": "5efee06f33db112491b532b5",  "username": "Username",  "first\_name": "Faris",  "full\_name": "Faris Takhassuna Zamzani",  "email": "test@testmail.com",  "phonenumber": 81326346957,  "jobtitle": "Programmer",  "password": "$2a$10$EuIxDMoNGG9MLyn96TeQZ.62FAthPnG1KVgrVtrdtDiZlS.bNtCiW",  "\_\_v": 0  },  "total": 15,  "year": "1932-11-07T16:30:00.000Z",  "institution": {  "\_id": "5ef21f1705d9c7c48737a63e",  "name": "Puskesmas Jatinom",  "address": "Jatinom",  "city": "Klaten",  "province": "Jawa Tengah",  "area": 255,  "\_\_v": 0  },  "report": {  "question3": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  },  "d": {  "information": false  }  },  "question10": {  "a": {  "information": false  },  "b": {  "a": {  "information": false  },  "b": {  "information": "5 gas prop"  },  "c": {  "information": "5 tali prop"  }  },  "c": {  "a": {  "information": false  },  "b": {  "information": false  }  }  },  "question1": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": true  }  },  "question2": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question4": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question5": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question6": {  "a": {  "information": false  },  "b": {  "information": false  }  },  "question7": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question8": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question9": {  "information": true  },  "question11": {  "a": {  "information": false  },  "b": {  "information": "false"  }  }  },  "\_\_v": 0  },  {  "\_id": "5ef2a897962f3236526d50b6",  "author": {  "privilege": "Admin",  "nip": "",  "\_id": "5efee06f33db112491b532b5",  "username": "Username",  "first\_name": "Faris",  "full\_name": "Faris Takhassuna Zamzani",  "email": "test@testmail.com",  "phonenumber": 81326346957,  "jobtitle": "Programmer",  "password": "$2a$10$EuIxDMoNGG9MLyn96TeQZ.62FAthPnG1KVgrVtrdtDiZlS.bNtCiW",  "\_\_v": 0  },  "total": 15243535,  "year": "1932-11-07T16:30:00.000Z",  "institution": {  "\_id": "5ef21f1705d9c7c48737a63e",  "name": "Puskesmas Jatinom",  "address": "Jatinom",  "city": "Klaten",  "province": "Jawa Tengah",  "area": 255,  "\_\_v": 0  },  "report": {  "question3": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  },  "d": {  "information": false  }  },  "question10": {  "a": {  "information": false  },  "b": {  "a": {  "information": false  },  "b": {  "information": "5 gas prop"  },  "c": {  "information": "5 tali prop"  }  },  "c": {  "a": {  "information": false  },  "b": {  "information": false  }  }  },  "question1": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": true  }  },  "question2": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question4": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question5": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question6": {  "a": {  "information": false  },  "b": {  "information": false  }  },  "question7": {  "a": {  "information": false  },  "b": {  "information": true  },  "c": {  "information": false  }  },  "question8": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question9": {  "information": true  },  "question11": {  "a": {  "information": false  },  "b": {  "information": "false"  }  }  },  "\_\_v": 0  },  {  "\_id": "5ef60f3fe83e78e52bc51565",  "author": {  "privilege": "Admin",  "nip": "1991",  "\_id": "5efef0200846724a7c609393",  "username": "Admin",  "first\_name": "Administrator",  "full\_name": "Administrator",  "email": "admin@admin.mail",  "phonenumber": 1000,  "jobtitle": "Admin",  "password": "$2a$10$W0XN4zZxhqNrCoCjhG8utOQ/.KaKhz.qobz9vMt9iSBhPBaKRI0/K",  "\_\_v": 0  },  "total": 123,  "year": "2020-05-27T00:00:00.000Z",  "institution": {  "\_id": "5ef4c4d58d3f135b99ed2dd1",  "name": "RSI",  "address": "Jl Tegalyoso",  "city": "Klaten",  "province": "Jawa Tengah",  "area": 456,  "\_\_v": 0  },  "report": {  "question3": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  },  "d": {  "information": false  }  },  "question10": {  "a": {  "information": false  },  "b": {  "a": {  "information": false  },  "b": {  "information": "Tidak ada"  },  "c": {  "information": "Tidak ada"  }  },  "c": {  "a": {  "information": false  },  "b": {  "information": false  }  }  },  "question1": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question2": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question4": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question5": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question6": {  "a": {  "information": false  },  "b": {  "information": false  }  },  "question7": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question8": {  "a": {  "information": false  },  "b": {  "information": false  },  "c": {  "information": false  }  },  "question9": {  "information": false  },  "question11": {  "a": {  "information": false  },  "b": {  "information": "Tidak ada"  }  }  },  "\_\_v": 0  }  ] } |

* 1. *report\_semester*

|  |
| --- |
| {  "count": 3,  "data": [  {  "month": "Februari",  "\_id": "5f0019cdb897113f1676e9c2",  "author": {  "privilege": "Admin",  "nip": "1991",  "\_id": "5efef0200846724a7c609393",  "username": "Admin",  "first\_name": "Administrator",  "full\_name": "Administrator",  "email": "admin@admin.mail",  "phonenumber": 1000,  "jobtitle": "Admin",  "password": "$2a$10$W0XN4zZxhqNrCoCjhG8utOQ/.KaKhz.qobz9vMt9iSBhPBaKRI0/K",  "\_\_v": 0  },  "year": "2020-07-04T00:00:00.000Z",  "report": {  "question1": {  "total": 1,  "detail": "1"  },  "question2": {  "total": 1,  "detail": "1"  },  "question3": {  "total": 1,  "detail": "1"  },  "question4": {  "total": 1,  "detail": "1"  },  "question5": {  "total": 1,  "detail": "1"  },  "question6": {  "total": 1,  "detail": "1"  },  "question7": {  "total": 1,  "detail": "1"  },  "question8": {  "total": 1,  "detail": "1"  }  },  "\_\_v": 0  },  {  "month": "Maret",  "\_id": "5ef1fa214f11ee98b84ab627",  "author": {  "privilege": "Admin",  "nip": "",  "\_id": "5efee06f33db112491b532b5",  "username": "Username",  "first\_name": "Faris",  "full\_name": "Faris Takhassuna Zamzani",  "email": "test@testmail.com",  "phonenumber": 81326346957,  "jobtitle": "Programmer",  "password": "$2a$10$EuIxDMoNGG9MLyn96TeQZ.62FAthPnG1KVgrVtrdtDiZlS.bNtCiW",  "\_\_v": 0  },  "report": {  "question1": {  "total": 12,  "detail": "neat"  },  "question2": {  "total": 12,  "detail": "sol"  },  "question3": {  "total": 12,  "detail": "neat"  },  "question4": {  "total": 12,  "detail": "neat"  },  "question5": {  "total": 12,  "detail": "neat"  },  "question6": {  "total": 12,  "detail": "neat"  },  "question7": {  "total": 12,  "detail": "neat"  },  "question8": {  "total": 12,  "detail": "neat"  }  },  "\_\_v": 0,  "year": "2020-07-04T07:02:45.321Z"  },  {  "month": "Maret",  "\_id": "5ef194035c644d612e10ca2e",  "author": {  "privilege": "Admin",  "nip": "",  "\_id": "5efee06f33db112491b532b5",  "username": "Username",  "first\_name": "Faris",  "full\_name": "Faris Takhassuna Zamzani",  "email": "test@testmail.com",  "phonenumber": 81326346957,  "jobtitle": "Programmer",  "password": "$2a$10$EuIxDMoNGG9MLyn96TeQZ.62FAthPnG1KVgrVtrdtDiZlS.bNtCiW",  "\_\_v": 0  },  "report": {  "question1": {  "total": 12,  "detail": "neat"  },  "question2": {  "total": 12,  "detail": "neat"  },  "question3": {  "total": 12,  "detail": "neat"  },  "question4": {  "total": 12,  "detail": "neat"  },  "question5": {  "total": 12,  "detail": "neat"  },  "question6": {  "total": 12,  "detail": "neat"  },  "question7": {  "total": 12,  "detail": "neat"  },  "question8": {  "total": 12,  "detail": "neat"  }  },  "\_\_v": 0,  "year": "2020-07-04T07:02:57.542Z"  }  ] } |

1. Pengujian proses create data
   1. *notification*
   2. *institution*
   3. *report\_year*
   4. *report\_semester*
   5. *user*
2. Pengujian proses update data
   1. *notification*
   2. *institution*
   3. *report\_year*
   4. *report\_semester*
   5. *user*
3. Pengujian proses delete single data
   1. *notification*
   2. *institution*
   3. *report\_year*
   4. *report\_semester*
   5. *user*
      1. **Hasil Pengujian antarmuka**

### **BAB VII**

**KESIMPULAN DAN SARAN**

* 1. **Kesimpulan**
  2. **Saran**

### **DAFTAR PUSTAKA**

1. Schwaber, Ken. "Scrum development process." Business object design and implementation. Springer, London, 1997. 117-134. [↑](#footnote-ref-0)