PEMBANGUNAN SISTEM INFORMASI PENGELOLAAN DATA PEGAWAI UNIVERSITAS ANDALAS BERBASIS *WEB* DENGAN FASILITAS *WEB SERVICE*

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Menyelesaikan Studi Strata-1 pada Departemen Sistem Informasi Fakultas Teknologi Informasi Universitas Andalas

Oleh:

Shinta Melany Anwar

1611521023

Pembimbing:

Husnil Kamil, M.T

198201182008121002

DEPARTEMEN SISTEM INFORMASI

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2023

LEMBAR PENGESAHAN

PEMBANGUNAN SISTEM INFORMASI PENGELOLAAN DATA PEGAWAI UNIVERSITAS ANDALAS BERBASIS *WEB* DENGAN FASILITAS *WEB SERVICE*

Oleh:

Shinta Melany Anwar

1611521023

LULUS SIDANG TUGAS AKHIR

13 Juni 2023

Padang, 18 Juli 2023

Telah diperiksa dan disetujui oleh

Pembimbing Tugas Akhir

Pembimbing Tugas Akhir

Husnil Kamil, M.T.

NIP. 198201182008121002

Mengetahui,

Ketua Departemen Sistem Informasi

Husnil Kamil, M.T.

NIP. 198201182008121002

PERNYATAAN

Saya menyatakan bahwa laporan tugas akhir saya yang berjudul "**Pembangunan Sistem Informasi Pengelolaan Data Pegawai Universitas Andalas Berbasis *Web* dengan Fasilitas *Web Service***" ini tidak pernah diajukan sebagai salah satu syarat menyelesaikan mata kuliah tugas akhir di suatu perguruan tinggi dan sepengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain kecuali yang secara tertulis dirujuk dalam naskah ini dan disebutkan dalam daftar pustaka.

Padang, Juni 2023

Penulis,

Shinta Melany Anwar

KATA PENGANTAR

Puji dan syukur penulis ucapkan kehadirat Allah SWT, atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir ini yang berjudul "Pembangunan Sistem Informasi Pengelolaan Data Pegawai Universitas Andalas Berbasis *Web* dengan Fasilitas *Web Service*". Tugas akhir ini disusun sebagai salah satu syarat untuk memenuhi salah satu syarat dalam penyelesaian mata kuliah Tugas Akhir pada Departemen Sistem Informasi Fakultas Teknologi Informasi Universitas Andalas.

Dalam penyusunan tugas akhir ini penulis banyak mendapatkan bantuan dari berbagai pihak. Oleh sebab itu, penulis mengucapkan terima kasih atas bimbingan dan bantuan kepada:

1. Bapak Husnil Kamil, M.T. sebagai Kepala Departemen Sistem Informasi Fakultas Teknologi Informasi Universitas Andalas dan sebagai dosen pembimbing dalam tugas akhir ini.
2. Seluruh pihak yang terlibat dan yang telah membantu dan memberikan dukungan kepada penulis baik secara moril maupun materil dalam pembuatan tugas akhir ini.

Penulis menyadari bahwa dalam penyusunan tugas akhir ini masih terdapat banyak kekurangan. Untuk itu penulis sangat mengharapkan kritik dan saran yang bersifat membangun dari pembaca melalui *email* shintamelanyanwar@gmail.com agar penelitian selanjutnya dapat dillakukan lebih baik lagi.

Padang, Juni 2023

Penulis,

Shinta Melany Anwar

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Sistem informasi kepegawaian Universitas Andalas adalah suatu sistem yang mengelola data seluruh pegawai di lingkungan Universitas Andalas. Saat ini sistem yang berjalan di Universitas Andalas tidak efisien karena data pada sistem kepegawaian yang ada saat ini tidak terintegrasi dengan sistem lainnya. Hal ini mengakibatkan redudansi data. Selain itu setiap database dari sistem memiliki perbedaan update data. Oleh karena itu dibuatlah sistem informasi pengelolaan data pegawai yang terintegrasi sehingga memudahkan pihak yang berkepentingan mempergunakan aplikasi ini untuk memanajemen kepegawaian pada instansi yang membutuhkannya. Web service merupakan teknologi yang dapat digunakan untuk melakukan integrasi data. Web service dibangun menggunakan arsitektur REST dengan menggunakan framework Laravel. Metode penelitian yang digunakan adalah identifikasi masalah, studi literatur, pengumpulan data untuk memperoleh informasi yang dibutuhkan, analisis sistem, perancangan sistem, pengkodean dan pengujian aplikasi menggunakan blackbox testing. Aplikasi ini dibangun menggunakan metode pengembangan waterfall dan menggunakan bahasa pemograman PHP. Penelitian ini menghasilkan sistem informasi pengelolaan data pegawai Universitas Andalas berbasis web yang dilengkapi web service. Hasil dari dibangunnya sistem informasi ini dapat menjadi solusi yang ditawarkan dengan harapan mengatasi segala kendala serta mempermudah pengelolaan data pegawai di Universitas Andalas.

Kata kunci: Pengelolaan data, integrasi, web service, pegawai, Universitas Andalas.

BAB I
PENDAHULUAN

Bab ini menjelaskan latar belakang masalah rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian serta sistematika penulisan pada tugas akhir ini.

1.1 Latar Belakang

Perguruan tinggi sebagai suatu lembaga pendidikan jenjang terakhir dalam hierarki pendidikan formal mempunyai tiga misi yang diemban yang dikenal dengan Tri Dharma Perguruan Tinggi. Dalam undang-undang pasal 39 UU No.20 tahun 2003 tentang sistem pendidikan nasional, yang menyatakan bahwa (1) Tenaga kependidikan bertugas melaksanakan administrasi, pengelolaan, pengembangan, pengawasan, dan pelayanan teknis untuk menunjang proses pendidikan pada satuan pendidikan, dan (2) Pendidik merupakan tenaga profesional yang bertugas merencanakan dan melaksanakan proses pembelajaran, menilai hasil pembelajaran, melakukan pembimbingan dan pelatihan, serta melakukan penelitian dan pengabdian kepada masyarakat, terutama bagi pendidik pada perguruan tinggi. Oleh karena itu, dalam mewujudkan Tri Dharma Perguruan tinggi di Universitas Andalas, maka diharuskan adanya pelaksanaan tugas yang baik dalam pengelolaan administrasi kepegawaian di Universitas Andalas. Pengelolaan administrasi tersebut terdiri dari rangkaian pengelolaan data pribadi pegawai, mutasi, dan riwayat pegawai. Besarnya data kepegawaian yang dikelola menimbulkan berbagai kendala baik itu dari segi efektivitas maupun efisiensi waktu dan sumber daya.

Untuk meningkatkan pelayanan dalam administrasi kepegawaian maka dibutuhkan sistem yang bisa menyediakan layanan integrasi bagi aplikasi lain yang membutuhkan data kepegawaian tersebut, sehingga memudahkan pengelolaan data dan administrasi secara efektif, efisien. Kelebihan dari aplikasi berbasis web ini adalah pengolahan informasi dan data lebih mudah dan tidak memakan waktu lama, pendataan lebih akurat dan mudah sehingga dapat menghindari kesalahan data pegawai, pengelolaan data lebih cepat dan berimbang dengan biaya yang dikeluarkan, dan tidak ada perlu mencari informasi secara manual. (Heldiansyah *et al.*, 2016).

Saat ini di Universitas Andalas sudah memiliki sistem informasi kepegawaian yang digunakan untuk proses penyimpanan dan pengolahan data-data kepegawaian. Sistem informasi kepegawaian digunakan untuk mempercepat proses pencatatan dan pengolahan data dan mampu menyajikan informasi kepegawaian kapan saja, sehingga informasi yang diminta dapat tepat waktu, tepat sasaran, dan akurat. Sistem ini digunakan untuk mendukung operasional bagian kepegawaian dimana dengan sistem ini dapat menghasilkan berbagai laporan tentang kepegawaian dengan cepat. Berdasarkan Keputusan Menteri Dalam Negeri Nomor 17 Tahun 2000 tentang Sistem Informasi Manajemen Kepegawaian Departemen Dalam Negeri dan Pemerintah Daerah menjelaskan bahwa sistem informasi kepegawaian adalah gabungan yang terpadu yang terdiri dari perangkat pengolah, yaitu pengumpul, prosedur, tenaga pengolah, dan perangkat lunak. Perangkat penyimpan, yaitu pusat data dan bank data serta perangkat komunikasi yang saling menghubungkan, ketergantungan, dan penentu yang digunakan untuk menyediakan informasi di bidang kepegawaian. Sistem informasi kepegawaian berisi data seperti data pokok pegawai, biodata pegawai, jabatan, pangkat, hubungan keluarga, dan unit kerja. Dengan tersedianya data kepegawaian pada sistem ini, maka pihak yang berkepentingan dapat mempergunakan data tersebut pada instansi yang membutuhkannya. Seperti portal unand, sistem informasi akademik, SIPPMI, remunerasi, MyFMIPA pada Fakultas MIPA dan SIMFONI pada Fakultas Kedokteran.

Terkait pengelolaan data kepegawaian Sofika Enggari sebelumnya telah melakukan penelitian pada tahun 2017 dengan judul Pengembangan Sistem Informasi Administrasi Kepegawaian pada Fakultas Ilmu Sosial Ilmu Politik Universitas Andalas Padang Berbasis *Web*. Penelitian menunjukkan bahwa sistem informasi dapat meningkatkan proses pengelolaan data pegawai. Ini ditunjukkan oleh kecepatan dan ketepatan waktu dalam memproses data, serta akurasi dan kebenaran informasi. Dengan demikian, pembuatan sistem informasi administrasi memungkinkan pengiriman dan pencarian data dengan cepat, sehingga dapat membantu karyawan mengakses informasi administrasi pada Fakultas Ilmu Politik Fakultas Ilmu Sosial. Permasalahan pada Sistem Informasi Kepegawaian Universitas Andalas saat ini yaitu aplikasi tidak berjalan secara efisien karena data

pada sistem informasi kepegawaian yang ada saat ini tidak terintegrasi dengan sistem lainnya. Hal ini mengakibatkan banyaknya *database* dari tiap sistem yang ada di Universitas Andalas, sehingga tiap-tiap *database* pada masing-masing sistem yang dikembangkan memiliki perbedaan *update* data. Selain itu, pengisian data pegawai untuk sistem yang ada saat ini dilakukan dengan pengumpulan data dari *excel* di tiap fakultas, dan admin pada universitas yang melakukan pengisian data. Hal ini mengakibatkan adanya redudansi data sehingga menyulitkan *user* yang memerlukan data yang akurat dan tepat. Sistem ini lebih baik dari sistem sebelumnya karena konsep integrasi data yang ada digunakan. Sistem sebelumnya menggunakan penginputan data manual ke Microsoft Office Excel, yang memperlambat pengolahan data dan menyebabkan ketidaksamaan data antara pusat dan cabang di berbagai bidang bisnis (Nurfaizah, 2017).

Untuk mengatasi hal tersebut perlu dilakukan pembaruan aplikasi sistem informasi kepegawaian yang dilengkapi dengan integrasi data memanfaatkan API (*Application Programming Interface*) agar data pada sistem informasi kepegawaian menjadi satu-satunya pusat data kepegawaian dan dapat digunakan di aplikasi lain yang dikembangakan secara tersendiri oleh lembaga-lembaga yang ada di Universitas Andalas. Integrasi data adalah menyatukan atau menggabungkan data dari berbagai sumber database ke dalam gudang data untuk membantu manajemen data dan memudahkan pengguna melihat kesatuan data. Ini dapat mempermudah proses analisis untuk pengambilan keputusan, penyebaran data di lingkungan kerja, dan mencegah duplikat data (Siti, 2018). Diharapkan bahwa penggunaan teknologi untuk mengintegrasikan kedua sistem tersebut akan mengurangi kesalahan manusia dan mempersingkat waktu pengelolaan (Sutanto, 2017).

Oleh karena itu maka dirasa perlu dilakukan penelitian untuk membangun ulang sistem informasi pengelolaan data pegawai sebagai solusi yang ditawarkan dengan harapan dapat mengatasi kendala yang telah dijelaskan sebelumnya serta mempermudah pengelolaan sistem informasi kepegawaian di Universitas Andalas. Dalam pembangunan sistem ini menggunakan metode *waterfall* dan menggunakan *framework* Laravel. Dengan menggunakan arsitektur MVC, Laravel memungkinkan eksekusi yang lebih cepat, yang memudahkan proses pembangunan sistem. *Framework* Laravel juga dapat meningkatkan produktivitas pengembangan

website dan membuat proses pengerjaan menjadi lebih terstruktur (Valarezo & Guarda, 2018). Dengan Laravel, pengembang dapat mengaktifkan RESTful Controllers untuk membuat REST APIs tanpa harus menulis baris kode lagi. Sebaliknya, framework CodeIgniter tidak memiliki fitur untuk mengembangkan REST APIs, jadi pengembang harus menulis baris kode secara manual untuk membuat REST APIs yang unik (Eko, 2021). Bahasa pemrograman yang digunakan adalah *hypertext prepossesor* dan menggunakan *database* MySQL. Penelitian ini diberi judul "Pembangunan Sistem Informasi Pengelolaan Data Pegawai Universitas Andalas berbasis *web* dengan Fasilitas *Web Service*".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan tersebut, maka rumusan masalahnya adalah bagaimana membangun sistem informasi pengelolaan data kepegawaian berbasis *web* dilengkapi fasilitas *web service* yang dapat memenuhi kebutuhan informasi tentang pegawai di Universitas Andalas.

1.3 Batasan Masalah

Agar penelitian tidak menyimpang terlalu jauh dan ruang lingkup sistem yang akan dibangun tidak terlalu luas, maka dari masalah yang ada ditentukan batasan-batasan masalah sebagai berikut.

1. Sistem informasi yang dibangun hanya mencakup pengelolaan data pegawai Universitas Andalas.
2. Sistem informasi pengelolaan data pegawai yang dibangun hanya sampai proses pengujian sistem.
3. Sistem yang dibangun menggunakan *framework* Laravel dan *database* MySQL.
4. Pengujian sistem menggunakan metode *black box testing.*

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah membangun aplikasi pengelolaan data pegawai Universitas Andalas berbasis *web* dengan fasilitas *web service* untuk mengatasi segala kendala serta mempermudah pengelolaan data kepegawaian di Universitas Andalas.

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah sebagai solusi yang ditawarkan dalam pengembangan sistem informasi terhadap masalah yang ditemukan yaitu kemudahan pengelolaan data pegawai pada sistem informasi kepegawaian Universitas Andalas yang berbasis *web* agar lebih optimal.

1.6 Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibagi menjadi enam bab, yaitu:

BAB I PENDAHULUAN

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

BAB II TINJAUAN PUSTAKA

Bab ini berisi teori-teori dan informasi pendukung yang relevan berkaitan dengan penelitian ini.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan tentang objek penelitian dan metode pengumpulan data yang digunakan.

BAB IV ANALISIS DAN PERANCANGAN

Bab ini menjelaskan analisis dan perancangan sistem yang dilakukan menggunakan *ucase diagram, use case scenario*, dan *sequence diagram*, perancangan *database*, arsitektur aplikasi, *class diagram*, dan *user interface*.

BAB V IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini berisi tentang pengimplementasian aplikasi ke dalam Bahasa pemrograman berdasarkan analisis dan perancangan, serta pengujian terhadap hasil implementasi sistem**.**

BAB VI KESIMPULAN DAN SARAN

Bab ini menjelaskan menjelaskan kesimpulan dan saran dari seluruh pembahasan.

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan teori-teori yang digunakan untuk mendukung penelitian tugas akhir. Pada bab ini beberapa hal yang menjadi tinjauan pustaka mengenai sistem pengelolaan pegawai serta perangkat lunak pendukung yang digunakan untuk membangun sistem informasi pengelolaan data kepegawaian Universitas Andalas.

2.1 Sistem Informasi

Sistem adalah kumpulan komponen yang saling berhubungan yang bekerja sama untuk mencapai suatu tujuan. Subsistem yang lebih kecil membentuk sistem yang lebih besar dan saling bergantung satu sama lain (Romney *et al.*, 2015). Sistem juga dapat didefinisikan sebagai sekumpulan komponen yang saling berhubungan dan mempengaruhi satu sama lain dalam melakukan tugas bersama untuk mencapai tujuan tertentu (Suprihadi dan Rini, 2013). Informasi yaitu kumpulan fakta dari suatu peristiwa atau kejadian yang tidak ada artinya, digunakan untuk menggambarkan data yang diolah menjadi lebih bermanfaat dan bermakna bagi orang yang menerimanya (Miyanto, 2015).

Sistem informasi adalah kombinasi yang teratur antara orang-orang, perangkat keras, perangkat lunak, jaringan komunikasi, dan sumber data yang mengumpulkan, mengubah, dan mendistribusikan informasi dalam suatu organisasi. Sistem informasi juga merupakan kumpulan komponen yang saling berhubungan satu sama lain yang berguna untuk mengintegrasikan, mengolah, menyimpan, dan menyebarkan informasi (Cahyanti dan Purnama, 2012).

2.2 Sistem Informasi Kepegawaian

Sistem informasi manajemen kepegawaian didefinisikan sebagai sistem informasi terpadu, yang meliputi pendataan pegawai, pengolahan data, prosedur, tata kerja, sumber daya manusia dan teknologi informasi untuk menghasilkan informasi yang cepat, lengkap dan akurat untuk membantu proses administrasi kepegawaian (Amalia *et al.*, 2012).

Berdasarkan Keputusan Menteri Dalam Negeri Nomor 17 Tahun 2000 tentang Sistem Informasi Manajemen Kepegawaian Depdagri dan Pemda

menyebutkan bahwa Sistem Manajemen Kepegawaian adalah kumpulan yang terpadu dari perangkat lunak dan perangkat pengolahan, yang mencakup pengumpulan, prosedur, tenaga pengolah, dan perangkat lunak. Perangkat penyimpanan, yang mencakup bank data dan pusat data, serta perangkat komunikasi, yang saling berkaitan, bergantung, dan menentukan untuk menyediakan informasi di bidang kepegawaian Tujuan dari program pengelolaan data pegawai ini adalah untuk mendukung sistem manajemen pegawai yang rasional dan pengembangan sumber daya manusia pemerintah; menciptakan data kepegawaian yang mutakhir dan terintegrasi; memberikan informasi yang akurat tentang keperluan perencanaan, pengembangan, kesejahteraan, dan pengendalian pegawai; dan membantu proses pekerjaan yang lebih lancar di bidang kepegawaian, terutama dalam pembuatan laporan kerja.

Bagian kepegawaian memanfaatkan sistem ini untuk menghasilkan berbagai laporan dengan cepat. Selain itu, sistem informasi manajemen kepegawaian ini dapat digunakan oleh eksekutif untuk membuat keputusan tentang data kepegawaian, seperti riwayat karier pegawai yang dipromosikan untuk jabatan tertentu.

Sistem informasi kepegawaian digunakan untuk mempercepat proses pencatatan dan pengolahan data serta mampu menyajikan informasi kepegawaian setiap saat, sehingga informasi yang diminta dapat tepat waktu, tepat sasaran, dan akurat. Selain itu, peran penting operator dalam menyortir dan menginput data, serta menggunakan perangkat dengan baik membutuhkan ketelitian.

2.3 Perangkat Lunak Pendukung

Bagian ini menjelaskan tentang perangkat lunak pendukung yang digunakan dalam pengembangan sistem informasi.

2.3.1 *Database*

Basis data, juga disebut database, adalah kumpulan data yang saling terhubung, disimpan secara bersama-sama pada media, diorganisasikan menurut skema atau struktur tertentu, dan dapat dimanipulasi dengan software untuk tujuan tertentu. Basis data juga dapat didefinisikan sebagai kumpulan data yang disusun dalam bentuk beberapa tabel yang berdiri sendiri dan berhubungan satu sama lain. (Pamungkas, 2017). Dengan demikian, database dapat didefinisikan sebagai

kumpulan beberapa file yang saling berhubungan yang membentuk data baru dengan nilai informasi yang bermanfaat.

Dalam penelitian ini penulis akan menggunakan *database* MySQL. MySQL adalah perangkat lunak *database* dan merupakan jenis data relasional, artinya MySQL menyimpan datanya dalam tabel yang saling berhubungan. Kelebihan penyimpanan data dalam *database* adalah mudah untuk menyimpan dan menampilkan data karena sudah dalam bentuk tabel (Winarno, 2014).

2.3.2 *Framework*

Framework adalah kerangka kerja yang memudahkan *programmer* untuk membuat sebuah aplikasi sehingga *programmer* akan lebih mudah melakukan perubahan (*customize*) terhadap aplikasinya dan dapat memakainya kembali untuk aplikasi lain yang sejenis (Rosa dan Shalahuddin, 2015). *Framework* adalah kumpulan fungsi atau perintah dasar yang saling berhubungan dan membentuk aturan tertentu. Oleh karena itu, aplikasi web harus mengikuti aturan framework tersebut. Dengan menggunakan *framework* tidak perlu memikirkan kode perintah dan fungsi dasar aplikasi *web* (Wardana, 2010).

Pada penelitian ini digunakan *framework* Laravel. Laravel adalah salah satu *framework* yang menggunakan prinsip MVC (*model, view*, dan *controller*) yang digunakan untuk membangun dan mengembangkan aplikasi berbasis *web* yang ditulis menggunakan bahasa pemrograman PHP. Laravel awalnya dirancang karena keinginan terhadap *framework* PHP yang lebih baik dari *codeigniter*. Kelebihan menggunakan laravel adalah berupa *syntax* yang digunakan simpel dan mudah dibaca. Laravel dirancang untuk mempermudah perancangan *web*. Dengan menggunakan laravel kodingan akan jauh lebih sedikit. Laravel cukup fleksibel untuk bekerja dengan berbagai macam sistem tidak peduli seberapa unik sistem tersebut. Dibandingkan dengan *framework* lain, Laravel menawarkan fitur canggih seperti artisan (*Command Line Tools), blade template, routing, eloquent relational mapping (ORM), migration, seeder,* dan banyak lagi (McCool, 2012).

2.3.3 PHP (*Preprocessor Hypertext Preprocessor*)

PHP merupakan bahasa pemrograman yang mudah dipelajari dan diimplementasikan. PHP adalah bahasa pemrograman *scripting* yang ditempatkan di *server* yang umumnya digunakan untuk membuat aplikasi *web* dinamis. Tujuan

web dinamis adalah untuk menampilkan isi *database* pada halaman *web* untuk memenuhi permintaan saat ini. PHP juga dapat digunakan dalam baris perintah, yang berarti bahwa skrip dapat dijalankan tanpa menggunakan server web atau browser (Kadir, 2008). PHP adalah bahasa pemrograman berbasis web yang digunakan untuk membuat aplikasi berbasis web. Ini termasuk bahasa pemrograman yang dapat berjalan di sisi server, atau bahasa sisi server. Dengan demikian, program yang dibuat dengan kode PHP tidak dapat berjalan kecuali dijalankan di server web, tanpa server web yang terus berjalan, program tersebut tidak akan dapat dijalankan (Nugroho, 2013).

Selain menjadi salah satu dari sembilan bahasa pemograman web paling populer saat ini, PHP telah berkembang sedemikian rupa sehingga banyak framework yang menggunakannya, salah satunya Laravel. PHP memiliki beberapa kelebihan, yaitu (Andi, 2007):

1. Mudah dibuat dan dijalankan.
2. Mampu berjalan pada berbagai sistem operasi dan web server.
3. PHP bersifat *open source*.
4. Dapat diletakkan dalam tag HTML.

2.3.4 API (*Application Programming Interface*)

API adalah singkatan dari *Application Programming Interface* yaitu sebuah *software* yang memungkinkan para *developer* untk mengintegrasikan dan mengizinkan dua aplikasi terhubung satu sama lain secara bersamaan. Salah satu cara API digunakan adalah sebagai penghubung antara suatu aplikasi dan aplikasi lainnya, yang memungkinkan programmer menggunakan sistem fungsional. Sistem operasi mengelola proses ini. Kelebihan API ini adalah memungkinkan aplikasi lain berinteraksi dan berhubungan satu sama lain (Ramadhani, 2015).

API dapat digunakan untuk mengintegrasikan aplikasi dengan aplikasi lain dengan tujuan memungkinkan aplikasi yang sudah dibangun untuk berbagi data satu sama lain. API juga berfungsi sebagai alat untuk berkomunikasi dengan pengembang yang menggunakan berbagai bahasa pemograman. Pengembang tidak perlu menyediakan data sendiri, mereka cukup mengambil data dan informasi dari *platform* melalui API, dan kemudian API dapat membuat *website* dengan berbagai fitur yang mudah digunakan (Zulfian, 2022).

API adalah kumpulan instruksi yang disimpan dalam library dan menunjukkan bagaimana program dapat berinteraksi dengan program lain. Dalam contoh ini, suatu rumah dapat digambarkan sebagai *software* yang akan dibuat. Dengan menyewa kontraktor yang memiliki kemampuan untuk menangani berbagai bagian, pemilik rumah dapat memberi mereka tugas yang harus diselesaikan oleh kontraktor tanpa mengetahui bagaimana cara kontraktor menyelesaikan tugas tersebut. Dalam hal ini, kontraktor berfungsi sebagai API dari *software* tersebut, yang dapat mengerjakan bagian tertentu dari *software* tersebut tanpa harus diketahui bagaimana prosedur dalam melakukan pekerjaan tersebut (Reddy, 2011).

2.3.5 *Web Service*

Web service adalah suatu sistem yang memungkinkan sistem berinteraksi satu sama lain dan berinteraksi dalam jaringan. Teknologi *web service* memungkinkan menjembatani semua data tanpa mempermasalahkan teknologi yang digunakan oleh masing-masing sumber (Siswoutomo, 2004). *Web Service* adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interaksi antar sistem pada suatu jaringan. *Web service* berfungsi sebagai fasilitas yang menyediakan layanan (data atau informasi) kepada sistem lain, memungkinkan sistem lain untuk berinteraksi dengannya melalui layanan yang disediakan oleh *web service*. *Web service* adalah sistem perangkat lunak yang dirancang untuk mendukung interaksi antar sistem di jaringan. Data informasi disimpan oleh *web service* dalam format JSON atau XML, sehingga sistem yang berbeda dapat mengaksesnya, terlepas dari berbagai platform, sistem operasi, dan bahasa pemrograman (Pardede *et al.*, 2013). Untuk melakukan pertukaran data, *web service* menggunakan format XML, yang memungkinkan sistem dari berbagai platform, sistem operasi, dan bahasa pemrograman untuk diakses.

Dalam membangun sistem pengelolaan data pegawai ini *web service* yang digunakan adalah *rest web service*. Data dan fungsi dianggap sebagai sumber daya dalam arsitektur REST dan dapat diakses melalui Uniform Resource Identifier (URI), yang biasanya berupa tautan web. (Kurniawan, 2014). REST API, juga dikenal sebagai RESTful API, adalah implementasi dari API (*Application Programming Interface*). REST (*Representational State Transfer*) adalah arsitektur

metode komunikasi yang menggunakan protokol HTTP REST untuk melakukan proses transaksi data. Tujuan dari REST API adalah untuk membuat sistem lebih cepat, cepat, dan mudah untuk dikembangkan, terutama dalam hal komunikasi dan transaksi data. API RESTful terdiri dari empat komponen penting yaitu URL *Design*, HTTP *Verbs*, HTTP *Response Code*, *Format Response* (Haryadi, 2016).

REST menggunakan metode perintah HTTP yang disebut verb untuk mengirimkan perintah ke server. Ada delapan metode perintah HTTP yaitu *GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE*, dan *CONNECT*. Namun, API REST hanya menggunakan empat metode ini: *GET, POST, PUT*, dan *DELETE*. (Rahman, 2013). *Resource* dapat ditambah dengan *Request method POST, Request method GET untuk mendapatkan daftar anggota dan detail anggota, Request method* PUT *untuk mengubah resource, dan Request method DELETE* untuk menghapus *resource* (Beni, 2018).

2.4 Studi Literatur

Pada sub bab ini dijelaskan beberapa penelitian terdahulu yang terkait dengan sistem informasi pengelolaan data pegawai. Untuk studi literatur dapat dilihat pada tabel 2.1.

Tabel 2.1 Studi Literatur

No	Judul dan Nama Peneliti	Deskripsi	Hasil Penelitian
1	Integrasi Data Kepegawaian Aplikasi Sub Sistem di Universitas Diponegoro Melalui *Web Service* Kodrat Iman Satoto, Rinta Kridalukmana (2014)	Tujuan: Untuk menjembatani kebutuhan data kepegawaian dari yang telah dikelola oleh unit sistem informasi tingkat universitas untuk dapat dipergunakan oleh sistem-sistem lain yang akan dikembangkan di tingkat fakultas dengan mengembangkan *web service.* Sistem ini tidak menggunakan CRUD (*create, read, update, delete).* Sistem ini menggunakan *framework* Codeigniter	Hasil yang diperoleh: - Penggunaan *web service* akan sangat membantu aplikasi mengakses data, sehingga yang sama tidak perlu disimpan berulang kali. - Dengan dukungan *web service*, sub- sub sistem lain dapat memanfaatkan informasi *login* dari sistem kepegawaian untuk masuk ke sub sistem tersebut.

Tabel 2.1 Studi Literatur (lanjutan)

No	Judul dan Nama Peneliti	Deskripsi	Hasil Penelitian
2	Sistem Informasi Pengolahan Data Karyawan Berbasis Web (Studi Kasus di PT Perkebunan Nusantara VIII Tambaksari) Muhammad Faizal dan Sanda Listya Putri (2017).	Tujuan: Membangun sistem informasi yang menyediakan informasi tentang data pegawai di lingkup PT Perkebunan Nusantara (Tambaksari) dan membangun sistem informasi pengolahan data pegawai sehingga dapat diakses oleh pihak yang membutuhkan pada saat yang bersamaan. Menggunakan bahasa pemrograman PHP dan *Database* MySQL. Perancangan sistem dilakukan dengan menggunakan metode *waterfall* yaitu analisis kebutuhan, perancangan sistem, penulisan kode program, pengujian program, dan implementasi program dan pemeliharaan program. Sistem ini tidak menggunakan integrasi data.	Hasil yang diperoleh: - Sistem dapat memudahkan pelayanan informasi bagi semua pihak yang berkepentingan. - Sistem yang terkomputerisasi maka pengolahan data pegawai dan informasi data pegawai dapat berjalan dengan baik dan dapat memenuhi kebutuhan pihak yang membutuhkan informasi.
3	Pengembangan Sistem Informasi Administrasi Kepegawaian Pada Fakultas Ilmu Sosial Ilmu Politik Universitas Andalas Padang Berbasis *Web*. Sofika Enggari, Larissa Navia Rani, Fadli Hernanda (2017)	Tujuan: Membantu kinerja pegawai dalam pengelolaan data pegawai di Fakultas Ilmu Sosial Politik. Dengan penggunaan teknologi, efektivitas berbagai aspek pengelolaan informasi administrasi yang ditunjukkan oleh kecepatan dan ketepatan waktu dalam memproses informasi, serta akurasi dan kebenaran informasi (validasi) dapat ditingkatkan. Metode yang digunakan dalam pengembangan sistem ini adalah *Waterfall*. Bahasa yang digunakan adalah *PHP*, serta menggunakan *MYSQL*. Sistem ini tidak menggunakan integrasi data.	Hasil yang diperoleh: - Sistem ini dapat mengoptimalkan efisienskerja bagian kepegawaian dan memudahkan pelayanan informasi bagi semua pihak yang berkepentingan. - Penelitian ini memudahkan pimpinan untuk melakukan pendataan dan mengakses informasi pegawai.

Tabel 2.1 Studi Literatur (lanjutan)

No	Judul dan Nama Peneliti	Deskripsi	Hasil Penelitian
4	Perancangan Sistem Informasi Pendataan Pegawai pada Dinas Lingkungan Hidup Salatiga Berbasis Web Menggunakan Framework Laravel Kevin Setiawan, Magdalena A. Ineke Pakereng (2021)	Tujuan: Untuk merancang aplikasi sistem informasi pendataan pegawai berbasis *web* menggunakan *framework* Laravel. Sistem informasi kepegawaian merupakan sebuah sistem yang bertujuan untuk penggunaan informasi dalam pengelolaan data pegawai secara lebih efisien dan efektif. Sistem informasi kepegawaian juga membantu mengelola data pegawai. Sistem ini tidak membahas integrasi data.	Hasil yang diperoleh: - Aplikasi ini dapat memudahkan admin untuk memverifikasi data pegawai menambah data pegawai, dan menyetujui pengajuan cuti pegawai secara cepat dan efektif. - Sistem informasi pendataan pegawai ini juga dapat meminimalisir kesalahan dalam proses pendataan pegawai.
5	Aplikasi Pengelolaan Data Pegawai Berbasis REST API untuk Transfer Data Real Time dengan Framework Codeigniter Hendri Aji Pangestu, Dede Kurniadi, Yosep Septiana (2022)	Tujuan: Memenuhi setiap kebutuhan bagi para penggunanya, salah satu manfaat dari adanya teknologi yaitu dapat mengelola data baik data individu ataupun data instansi. Pengelolaan data terutama di bagian pertukaran data dapat dengan mudah di lakukan apabila suatu sistem sudah menerapkan *web service* atau biasa disebut *Representational State Transfer Application Programming Interface*.	Hasil yang diperoleh: - Membantu petugas di instansi pemerintahan dalam proses kelola mutase karena ditinjau secara *real time* dan adanya fitur kelola data pegawai yang dapat memudahkan proses penyimpanan dan pemanggilan data. Perubahan data yang diatur secara real time memungkinkan aplikasi untuk dapat melakukan perubahan data secara cepat sehingga pekerjaan dapat dilakukan dengan lebih efisien.

Berdasarkan penelitian-penelitian yang telah ada dapat disimpulkan bahwa peranan sistem informasi dalam pengelolaan sistem informasi kepegawaian dapat membuat proses pengelolaan menjadi lebih optimal. Namun pada penelitian sebelumnya sistem hanya terbatas pada pengelolaan data pegawai, padahal seharusnya diperlukan integrasi antar aplikasi data pegawai dengan apikasi lainnya yang membutuhkan data pegawai sehingga memudahkan pengelolaan data pegawai yang ditunjukkan oleh kecepatan dan ketepatan waktu dalam memproses informasi, serta akurasi dan kebenaran informasi dapat ditingkatkan. Oleh karena itu, pada penelitian ini penulis akan melakukan pembaruan aplikasi sistem informasi kepegawaian yang dilengkapi dengan *web service* menggunakan *framework waterfall* agar data pada sistem informasi kepegawaian menjadi satu-satunya pusat data kepegawaian dan dapat digunakan di aplikasi lain yang ada di Universitas Andalas.

BAB III
METEDOLOGI PENELITIAN

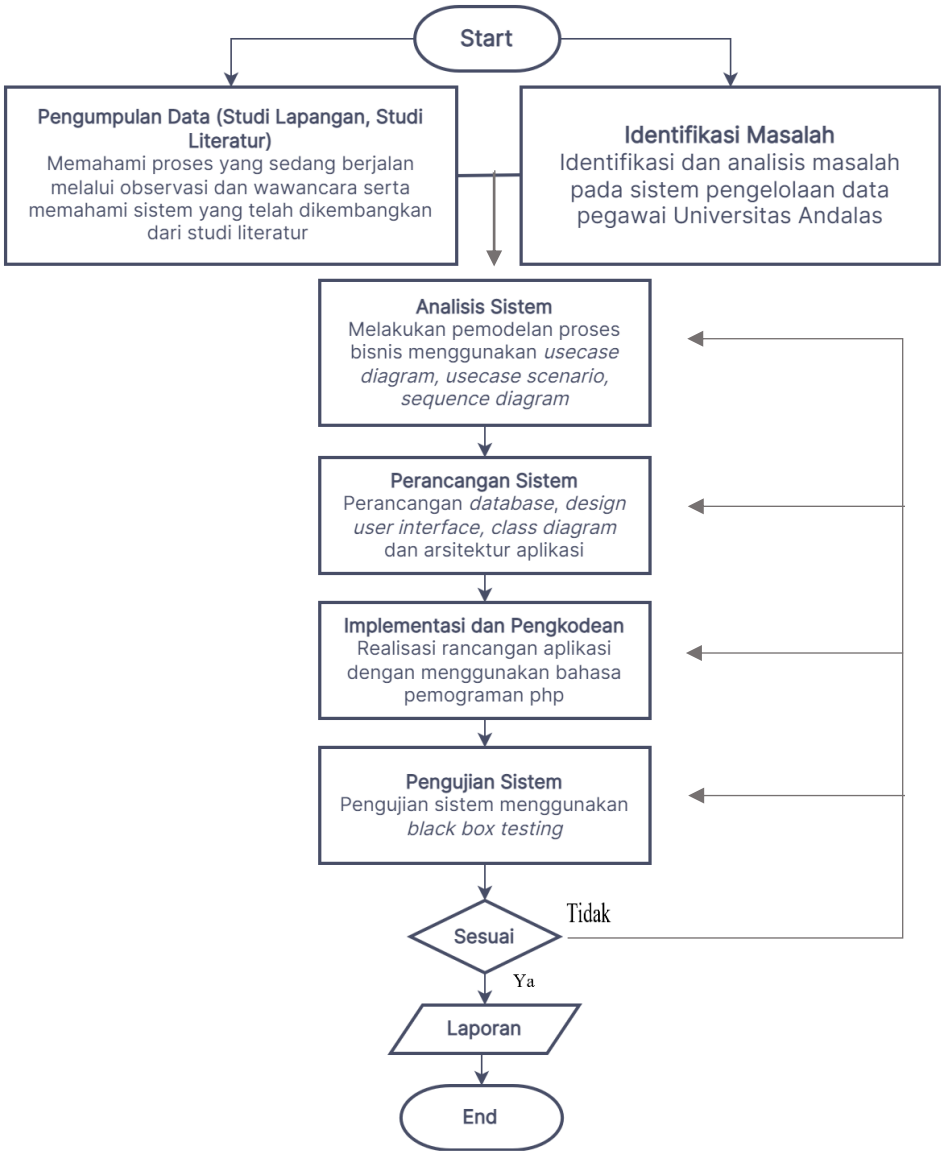
Bab ini menjelaskan tentang objek penelitian, metode penelitian, metode pengumpulan data, metode pengembangan sistem dan metode pengujian sistem yang digunakan dalam pembangunan sistem informasi pengelolaan data kepegawaian Universitas Andalas berbasis *web* dilengkapi fasilitas integrasi.

3.1 Metode Penelitian

Penelitian ini menggunakan studi pendahuluan dan studi literatur. Studi pendahuluan melibatkan pengamatan dan analisis proses pengelolaan data karyawan di Universitas Andalas, dan studi literatur melibatkan mempelajari penelitian sebelumnya tentang topik penelitian. Selanjutnya, pengumpulan data yang berkaitan dengan penelitian yang akan dilakukan.

Tahapan selanjutnya adalah analisis sistem dan pengumpulan kebutuhan sistem, dilakukan dengan menganalisis kebutuhan pengguna dan pemodelan sistem baru yang diusulkan. Hasil dari tahapan ini adalah analisis proses bisnis menggunakan diagram *use case*, skenario *use case* dan diagram *sequence*. Tahap selanjutnya adalah mendesain sistem. Perancangan sistem terdiri dari beberapa tahapan yaitu perancangan *database* (ERD), *class diagram*, desain *user interface*, dan perancangan arsitektur aplikasi.

Langkah berikutnya adalah memasukkan kebutuhan tersebut ke dalam pengkodean dengan menggunakan bahasa pemrograman PHP dengan *framework* Laravel dan *database* MySQL. Hasil dari tahapan ini adalah aplikasi pengelolaan data pegawai yang siap digunakan sesuai dengan rancangan sistem yang diusulkan. Setelah implementasi dan pengkodean selesai, pengujian sistem dilakukan pada tahap analisis. Jika penerapan yang dibuat memenuhi tujuannya, penelitian akan dilanjutkan ke tahap kesimpulan dan rekomendasi. Jika tidak, akan kembali ke tahap perancangan. Tahapan penelitian ini digambarkan pada *flowchart* pada gambar 3.1



Gambar 3.1 *Flowchart* Penelitian

Penjelasan *flow chart* penelitian yang dilakukan pada Gambar 3.1 sebagai berikut:

1. Identifikasi masalah

   Identifikasi masalah berupa penetapan dan analisis masalah yang terjadi pada pengelolaan data pegawai Universitas Andalas. Dalam tahapan ini juga dilakukan penentuan objek penelitian. Objek penelitian yang dikaji pada penelitian ini adalah Sistem Informasi Kepegawaian Universitas Andalas.

2. Studi literatur

Studi literatur menggunakan informasi dari penelitian sebelumnya sebagai data acuan saat melakukan penelitian.

3. Pengumpulan data

Untuk mendukung penelitian ini, wawancara dan pengumpulan dokumen digunakan untuk mengumpulkan data dan informasi tentang pengelolaan data pegawai. Informasi yang dikumpulkan meliputi pengelolaan data pegawai, proses yang sedang berlangsung, kelebihan dan kekurangan sistem saat ini, dan kebutuhan sistem yang lebih baik dengan memanfaatkan teknologi informasi.

4. Analisis sistem

Tahapan analisis sistem dilakukan dengan menganalisis sistem yang sedang berjalan dan kebutuhan sistem. Tahapan ini juga menghasilkan gambaran tentang kelebihan dan kekurangan sistem yang dibangun. Pada tahapan analisis proses bisnis ini dijabarkan kedalam *use case diagram, use case scenario*, dan *sequence diagram.*

5. Perancangan sistem

Pembuatan alur user dalam aplikasi, perancangan basis data (*database*), dan perancangan antarmuka pengguna aplikasi dilakukan dalam tahapan perancangan sistem. Tahapan ini menghasilkan *class diagram*, pembangunan *database* menggunakan ERD, struktur tabel, arsitektur aplikasi *web*, dan perancangan aplikasi.

6. Pengkodean

Dalam proses perancangan sistem, tahap pengkodean adalah tahap yang paling penting. Pada tahap ini, kode program dibuat untuk membangun sistem informasi yang berfungsi untuk mengelola data pegawai. Program komputer yang dibangun membuat aplikasi berbasis *web* dengan bahasa pemrograman PHP dan *framework*Laravel.

7. Pengujian sistem

Proses pengujian sistem mencakup membangun lingkungan sistem, memasang komponen pendukung, dan menguji aplikasi yang telah dibangun. Pemberian gambaran tentang aktor yang dapat menggunakan sistem dan teknis kerja sistem yang dibangun adalah bagian dari lingkungan sistem yang

dibangun. Instalasi aplikasi dan komponen pendukung lainnya dilakukan pada PC yang digunakan untuk mengelola data pegawai. Metode *blackbox testing* digunakan untuk menguji aplikasi. Tahapan ini menghasilkan data yang menunjukkan apakah aplikasi sesuai dengan fungsional yang dibangun.

8. Laporan

   Laporan dibuat setelah analisis, perancangan, dan pengujian sistem yang dibangun. Tahapan ini menghasilkan data dan informasi dari analisis, pengembangan, dan implementasi sistem. Laporan ini dapat digunakan sebagai referensi untuk penelitian selanjutnya.

3.2 Metode Pengumpulan Data

Metode pengumpulan data yang diterapkan pada penelitian ini adalah dengan studi lapangan dan studi literatur. Studi lapangan terdiri dari wawancara dan analisis dokumen.

1. Wawancara

   Wawancara dilakukan untuk melengkapi informasi yang dibutuhkan secara lebih detail dan pasti tentang penelitian yang dilakukan. Wawancara dilakukan dengan ketua program studi sistem informasi Bapak Husnil Kamil, M.T. selaku *user* Sistem Informasi Kepegawaian Universitas Andalas, dan Bapak Irzon, S.Kom, M.Kom selaku staf kepegawaian Universitas Andalas.

2. Analisis dokumen

   Dalam analisis dokumen, dokumen yang terkait dengan sistem yang akan dibangun dipelajari, seperti data pegawai Universitas Andalas.

3. Studi literatur

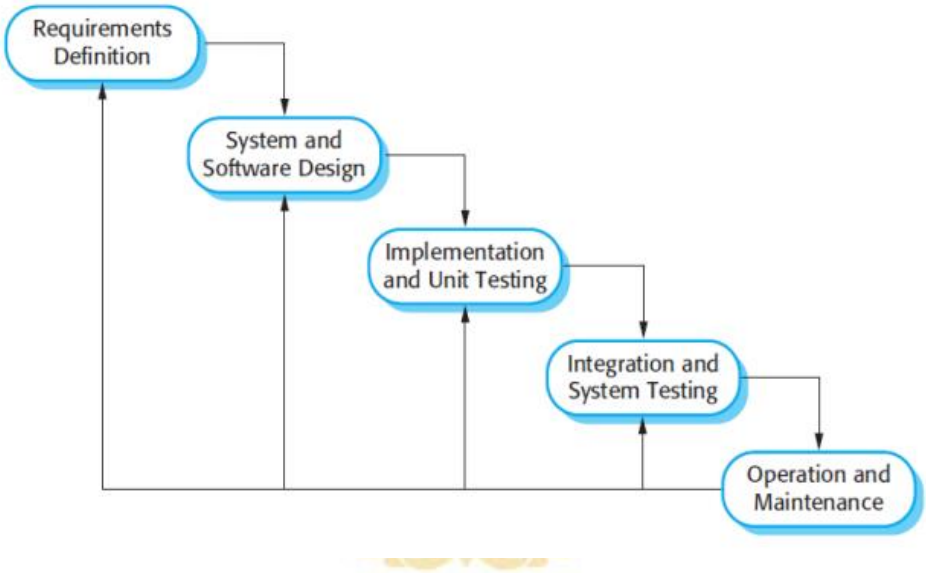
   Jurnal atau buku lain yang relevan dengan aplikasi yang akan dibangun dapat digunakan untuk mencari data melalui studi pustaka.

3.3 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah metode *Waterfall*. Metode *Waterfall* ini adalah salah satu model yang sering digunakan dari konsep *System Development Life Cycle* (SDLC). Metode ini digunakan untuk menyelesaikan masalah kompleks yang muncul selama proyek pengembangan perangkat lunak. Analisis, desain sistem, implementasi, pengujian

operasional, dan pemeliharaan adalah semua bagian dari metode pengembangan *waterfall*. Misi pengembangan yang kompleks dapat dipecahkan menggunakan model ini menjadi beberapa langkah logis (desain, kode, dan pengujian), yang pada akhirnya menghasilkan produk yang siap digunakan. Menggunakan metode *waterfall* karena fokus teknisnya (Simarmata, 2010). Metode waterfall adalah model penelitian yang sistematis dan sekuensial yang dimulai dari tahap pertama dan berakhir pada tahap terakhir. Tahapan tidak dapat diulang dan jika tahapan sebelumnya telah selesai, tahapan berikutnya akan dilakukan. (R. Choirudin and A. Adil, 2019).

Berdasarkan tahapan metode *waterfall* diatas penelitian ini hanya dilakukan dari tahap analisis hingga tahap pengintegrasian dan testing aplikasi yang dibangun.Tahapan metode *waterfall* dapat dilihat pada gambar 3.2.



Gambar 3.2 Tahapan metode *waterfall* (Sommerville, 2011)

Adapun penjelasan dari gambar tersebut adalah sebagai berikut:

1. Mendefenisikan dan Menganalisis Kebutuhan

   Proses ini mendefinisikan apa yang dapat dilakukan sistem, menganalisis hambatan yang ada, dan menentukan tujuan sistem yang akan dibangun. Hasil diskusi dengan pengguna kemudian diatur secara menyeluruh untuk mencapai spesifikasi sistem manajemen kurikulum yang diinginkan. Selain itu, tahapan ini menentukan output yang akan dihasilkan, fitur apa saja yang akan dimiliki aplikasi, dan fungsi yang akan dimilikinya.

2. Perancangan Sistem dan Perangkat Lunak

Pada tahap ini, perangkat lunak dan perangkat keras yang diperlukan untuk sistem dialokasikan untuk membentuk arsitektur sistem yang dibutuhkan secara keseluruhan. Identifikasi dan deskripsi sistem perangkat lunak dasar serta hubungannya adalah bagian dari desain perangkat lunak. Hasil dari tahap perancangan termasuk membangun proses sistem, membuat database aplikasi, dan membuat antarmuka pengguna

3. Implementasi dan pengujian sistem

Pada tahap ini, *system requirements* direalisasikan dan perancangan desain sebelumnya ditulis dalam bahasa pemrograman. Pada tahap ini, hanya implementasi yang dilakukan, dan pengujian sistem tidak dilakukan untuk setiap sistem.

4. Integrasi dan sistem *testing*

Proses ini mengintegrasikan seluruh program dan diuji sebagai sistem yang lengkap dan sesuai dengan kebutuhan sistem.

5. Pengoperasian dan pemeliharaan

Merupakan tahap terakhir dalam membangun sistem. Di sini, sistem dapat digunakan secara nyata dan diperbaiki jika ada kesalahan yang tidak ditemukan pada tahap sebelumnya.

3.4 Metode Pengujian Sistem

Metode yang digunakan dalam pengujian sistem ini dilakukan dengan metode pengujian *black box*. Pengujian *black box* adalah pengujian yang didasarkan pada apakah aplikasi, seperti antarmuka dan kesesuaian fungsionalnya, memenuhi persyaratan pengguna. Pengujian *black box* adalah metode pengujian perangkat lunak yang berfokus pada domain informasi perangkat lunak. Metode ini menguji fungsionalitas sistem dengan tujuan memastikan apakah perangkat lunak beroperasi dengan baik dengan berbagai masukan. (Indriasari, 2012).

Pengujian black box dilakukan untuk memastikan bahwa aplikasi yang diinginkan tidak memiliki kesalahan saat digunakan. Jika hasilnya tidak sesuai dengan kebutuhan pengguna, kesalahan atau ketidaksesuaian tersebut dicatat dalam tabel uji dan digunakan sebagai patokan untuk pengembangan sistem yang sedang dibangun.

BAB IV
ANALISIS DAN PERANCANGAN

Bab ini menjelaskan tentang analisis kebutuhan dan perancangan yang dilakukan untuk membangun sistem informasi pengelolaan data kepegawaian pada Sistem Informasi Kepegawaian Universitas Andalas. Pada tahapan analisis digambarkan menggunakan analisis kebutuhan fungsional, *usecase diagram, usecase scenario, sequence diagram,* dan *class analysis.* Untuk tahap perancangan sistem digambarkan dengan *entity relationship diagram, class diagram,* arsitektur aplikasi dan *user interface.*

4.1 Analisis Sistem

Tahapan analisis sistem menjelaskan tentang kedudukan sistem saat ini, sistem yang diusulkan, serta analisis sistem yang dimodelkan melalui UML (*unified modeling language*). UML yang digunakan dalam analisis sistem ini adalah *use case diagram, sequence diagram,* dan *class diagram.*

4.1.1 Analisis Kebutuhan Fungsional

Kebutuhan fungsional dibuat berdasarkan analisis sistem informasi terkait dan kebutuhan dari objek penelitian setelah dilakukan proses pengumpulan data. Kebutuhan fungsional hanya dua pengguna yaitu admin dan pegawai. Dalam penelitian ini yang bertugas sebagai admin merupakan bagian dari pegawai juga. Berikut ini adalah kebutuhan fungsional pengguna pada sistem informasi pengelolaan data pegawai:

1. Autentikasi

   Semua aktor (admin dan pegawai) dapat melakukan *login* dan *logout*.

2. Kebutuhan fungsional admin terhadap sistem

   Adapun kebutuhan fungsional admin terhadap sistem ini adalah sebagai berikut:

   a. Fungsi untuk mengelola data pribadi pegawai seperti menambah, mengedit, menghapus, dan melihat informasi.
   b. Fungsi untuk mengelola pangkat golongan seperti menambah, mengedit, menghapus, dan melihat informasi.

c. Fungsi untuk mengelola mutasi pegawai seperti menambah, mengedit, menghapus, dan melihat informasi.
d. Fungsi untuk mengelola jabatan fungsional seperti menambah, mengedit, menghapus, dan melihat informasi.
e. Fungsi untuk mengelola jabatan struktural seperti menambah, mengedit, menghapus, dan melihat informasi.
f. Fungsi untuk mengelola data keluarga pegawai seperti menambah, mengedit, menghapus, dan melihat informasi.
g. Fungsi untuk mengelola riwayat pendidikan seperti menambah, mengedit, menghapus, dan melihat informasi.
h. Fungsi untuk mengelola unit kerja seperti menambah, mengedit, menghapus, dan melihat Fungsi untuk mengelola diklat seperti menambah, mengedit, menghapus, dan melihat informasi.
i. Fungsi untuk mengelola *access token* API seperti menambah, mengedit, menghapus, dan melihat informasi.

3. Kebutuhan fungsional pegawai terhadap sistem

Adapun kebutuhan fungsional pegawai terhadap sistem ini adalah sebagai berikut:

a. Fungsi untuk mengelola data pribadi pegawai seperti menambah, mengedit, menghapus, dan melihat informasi.
b. Fungsi untuk mengelola data keluarga pegawai seperti menambah, mengedit, menghapus, dan melihat informasi.
c. Fungsi untuk mengelola riwayat pendidikan seperti menambah, mengedit, menghapus, dan melihat informasi.
d. Fungsi untuk mengelola diklat seperti menambah, mengedit, menghapus, dan melihat informasi.
e. Fungsi untuk mengelola pangkat golongan seperti menambah, mengedit, menghapus, dan melihat informasi.
f. Fungsi untuk mengelola mutasi pegawai seperti menambah, mengedit, menghapus, dan melihat informasi.
g. Fungsi untuk mengelola jabatan fungsional seperti menambah, mengedit, menghapus, dan melihat informasi.

h. Fungsi untuk mengelola jabatan struktural seperti menambah, mengedit, menghapus, dan melihat informasi.

i. Fungsi untuk melihat unit kerja

4. Kebutuhan fungsional sistem terhadap API

a. Fungsi untuk melihat semua data pegawai

b. Fungsi untuk melihat data pegawai berdasarkan NIK/NIP

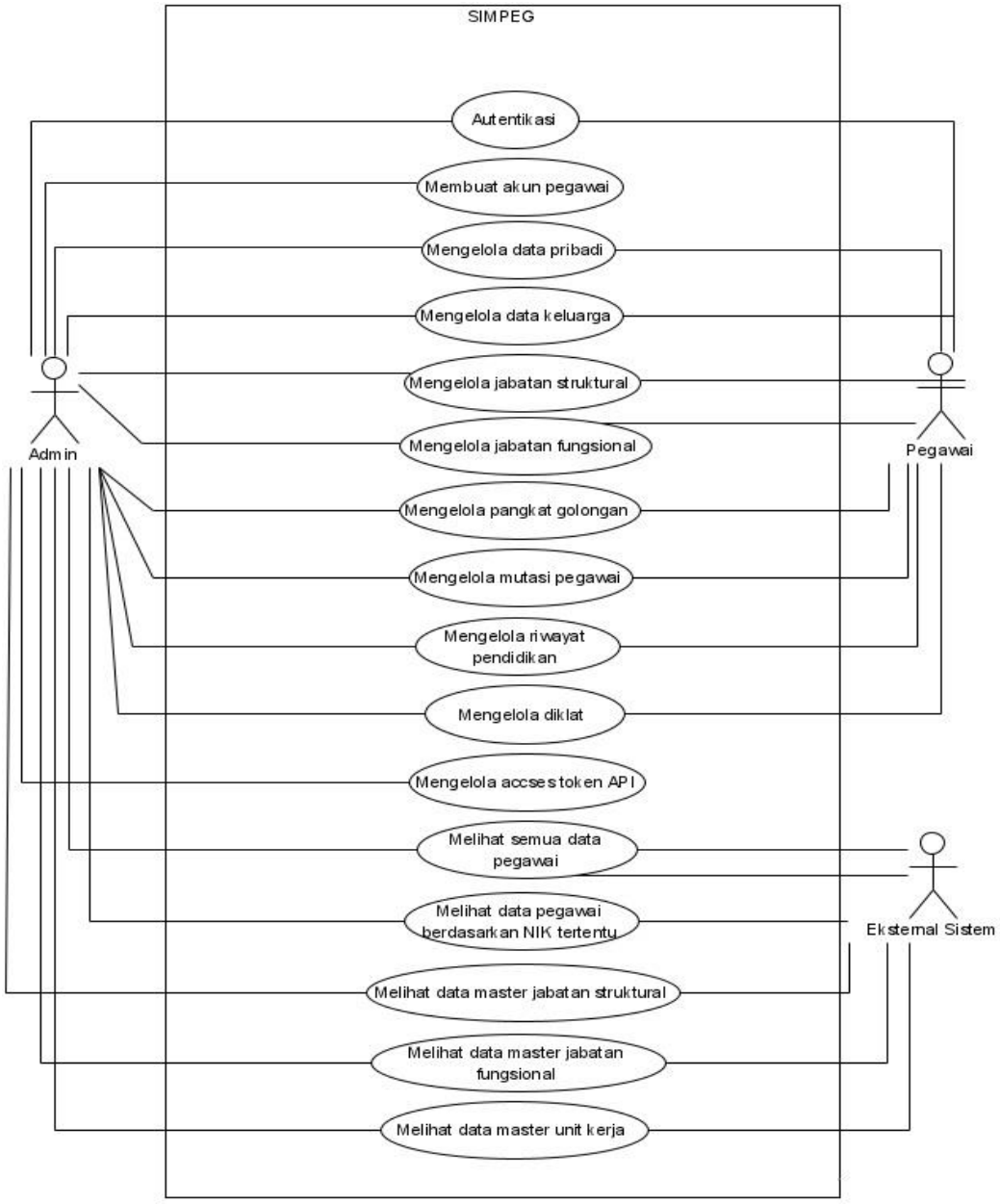
c. Fungsi untuk melihat data master jabatan structural

d. Fungsi untuk melihat data master jabatan fungsional

e. Fungsi untuk melihat data master unit kerja

4.1.2 *Use Case Diagram*

Berdasarkan kebutuhan fungsional yang telah dianalisis, dapat digambarkan menggunakan usecase diagram. Proses yang dilakukan oleh aktor yang ada dalam sistem disebut fungsional-fungsional yang ada. Ada tiga pihak yang terlibat dalam pembuatan sistem informasi pengelolaan data karyawan di Universitas Andalas, yaitu admin, karyawan, dan eksternal sistem. Aplikasi eksternal sistem akan menggunakan data karyawan yang dihubungkan ke web service. Antara aktor dan fungsional memiliki hubungan yang digambarkan dalam bentuk *usecase* diagram dapat dilihat pada gambar 4.1.



Gambar 4.1 *Use Case* Diagram

4.1.3 *Use Case Scenario*

Use case scenario menjelaskan bagaimana aktor berinteraksi dengan sistem. *Use case Scenario* memungkinkan untuk menjelaskan siapa yang berinteraksi dengan sistem dan apa yang harus dilakukan sistem. Berikut beberapa contoh dari *use case scenario* seperti admin mengelola data pegawai, mengelola jabatan pegawai dan mencetak data pegawai. Skenario dari *use case* lainnya dapat dilihat pada lampiran E.

4.1.3.1 *Use Case Scenario* Menambah Data Pribadi Pegawai

Proses mengelola data pribadi pegawai dilakukan oleh admin. Admin tersebut diharuskan *login* terlebih dahulu ke aplikasi. Skenario *use case* kelola data pribadi pegawai dapat dilihat pada tabel 4.4.

Tabel 4.4 *Use Case Scenario* Menambah Data Pribadi Pegawai

Use Case	Menambah data pribadi pegawai
Actor	Admin
Entry Condition	Aktor telah *login*
Flow of Event	1. Aktor memilih menu daftar pegawai 2. Sistem menampilkan halaman daftar pegawai 3. Aktor mengklik tombol tambah pegawai yang berada pada sudut kiri atas tabel 4. Sistem menampilkan form tambah pegawai 5. Aktor mengisi *form* tambah pegawai 6. Aktor mengklik tombol simpan 7. Sistem menampilkan pemberitahuan berhasil 8. Sistem menampilkan halaman daftar pegawai
Exit Condition	Aktor berhasil menambahkan menu daftar pegawai
Alternative course	1. Aktor memilih menu daftar pegawai 2. Sistem menampilkan halaman daftar pegawai 3. Aktor mengklik tombol tambah pegawai 4. Sistem menampilkan form pengisian 5. Aktor mengisi form tambah pegawai 6. Aktor mengklik tombol simpan 7. Sistem memvalidasi dan menyimpan data jika validasi gagal sistem menampilkan form tambah pegawai dan notifikasi gagal

4.1.3.2 Use Case Scenario Menambah Jabatan Struktural

Proses menambah jabatan struktural dilakukan oleh admin. Admin tersebut diharuskan *login* terlebih dahulu ke aplikasi. Skenario *use case* kelola jabatan struktural dapat dilihat pada tabel 4.5.

Tabel 4.5 *Use Case Scenario* Menambah Jabatan Struktural

Use Case	Menambah Jabatan Struktural Pegawai
Actor	Admin
Entry Condition	Aktor telah *login*
Flow of Event	1. Aktor memilih menu daftar pegawai 2. Sistem menampilkan halaman daftar pegawai 3. Aktor mengklik tombol *show user* yang berada pada kolom aksi pada tabel daftar pegawai 4. Sistem menampilkan daftar jabatan struktural pegawai 5. Aktor mengklik tombol tambah jabatan struktural yang berada pada sudut kiri atas tabel 6. Sistem menampilkan form tambah jabatan struktural 7. Aktor mengisi *form* yang telah disediakan 8. Aktor mengklik tombol simpan 9. Sistem menampilkan pemberitahuan berhasil 10. Sistem menampilkan halaman daftar jabatan struktural pegawai
Exit Condition	Aktor berhasil menambahkan daftar jabatan struktural pegawai
Alternative course	1. Aktor memilih menu daftar pegawai 2. Sistem menampilkan halaman daftar pegawai 3. Aktor mengklik tombol *show user* yang berada pada kolom aksi pada tabel daftar pegawai 4. Sistem menampilkan daftar jabatan struktural pegawai 5. Aktor mengklik tombol tambah jabatan struktural yang berada pada sudut kiri atas tabel 6. Sistem menampilkan form tambah jabatan strukturaL

Tabel 4.5 *Use Case Scenario* Menambah Jabatan Struktural (lanjutan)

Alternative course	7. Aktor mengisi *form* yang telah disediakan 8. Aktor mengklik tombol simpan 9. Sistem memvalidasi dan menyimpan data jika validasi gagal sistem menampilkan form tambah jabatan struktural dan notifikasi gagal

4.1.3.3 *Use Case Scenario* Menambah *Access Token*

Proses menambah *access token* dilakukan oleh admin. Admin tersebut diharuskan *login* terlebih dahulu ke aplikasi. Skenario *use case* menambah *access token* dapat dilihat pada tabel 4.6.

Tabel 4.6 *Use Case Scenario* Menambah *Access Token*

Use Case	Menambah *Access Token*
Actor	Admin
Entry Condition	Aktor telah *login*
Flow of Event	1. Aktor memilih menu API 2. Sistem menampilkan halaman API 3. Aktor mengklik tombol tambah *access token* yang berada pada sudut kiri atas tabel 4. Sistem menampilkan form tambah *access token* 5. Aktor mengisi *form* yang telah disediakan 6. Aktor mengklik tombol simpan 7. Sistem menampilkan pemberitahuan berhasil 8. Sistem menampilkan halaman daftar *access token*
Exit Condition	Aktor berhasil menambahkan *access token*
Alternative course	1. Aktor memilih menu API 2. Sistem menampilkan halaman API 3. Aktor mengklik tombol tambah *access token* yang berada pada sudut kiri atas tabel 4. Sistem menampilkan form tambah *access token* 5. Aktor mengisi *form* yang telah disediakan 6. Aktor mengklik tombol simpan

Tabel 4.6 Use Case Scenario *Menambah* Access Token *(lanjutan)*

	7. Sistem memvalidasi dan menyimpan data jika validasi gagal sistem menampilkan form tambah jabatan struktural dan notifikasi gagal.

4.1.3.4 *Use Case Scenario* Melihat Semua Data Pegawai

Proses melihat semua data pegawai dilakukan oleh eksternal sistem. Eksternal sistem telah memiliki *access token* terlebih dahulu. Skenario *use case* melihat semua data pegawai dapat dilihat pada tabel 4.7.

Tabel 4.7 *Use Case Scenario* Melihat Semua Data Pegawai

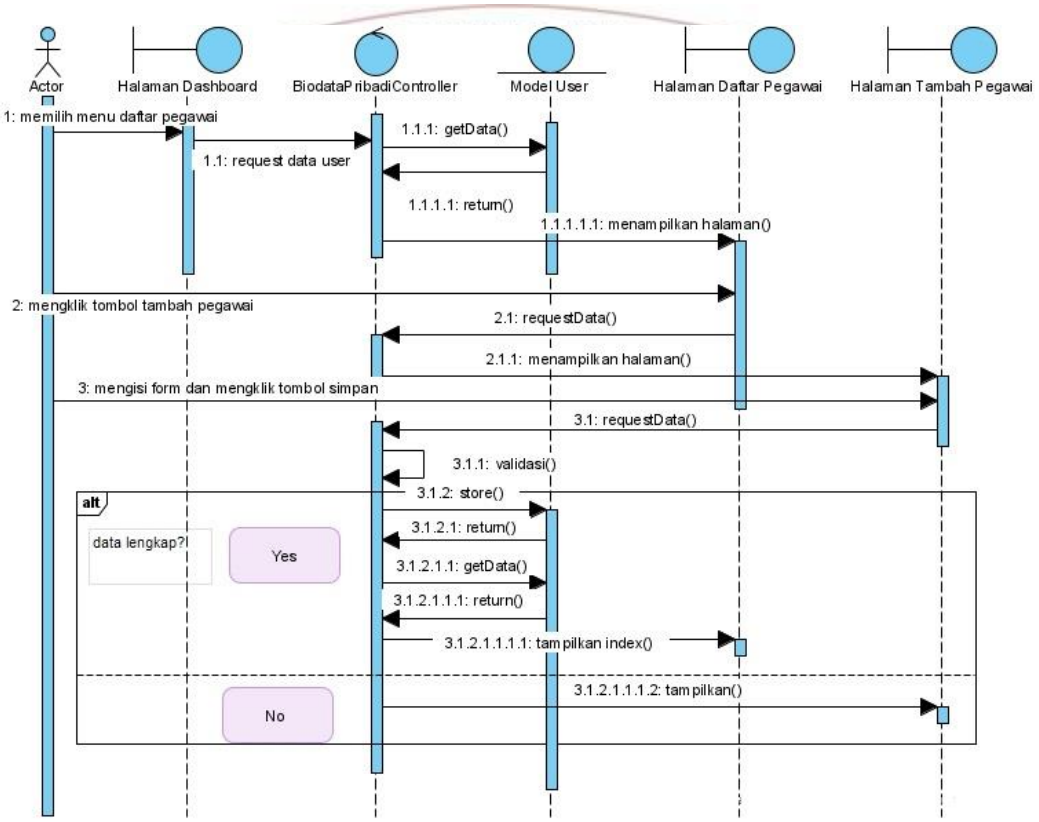
Use Case	Melihat Semua Data Pegawai
Actor	Eksternal sistem
Entry Condition	Eksternal sistem telah mendapatkan *Access Token* API
Flow of Event	1. Ekstenal sistem membuka aplikasi *Postman* 2. Sistem menampilkan *workspace Postman* 3. Eksternal sistem menginputkan *access token* pada *params* bagian *value.* 4. Eksternal sistem kemudian mengklik tombol *send* yang berada pada sudut kanan atas tabel. 5. Sistem menampilkan data seluruh pegawai Universitas Andalas.
Exit Condition	Sistem berhasil menampilkan data seluruh pegawai Universitas Andalas.

4.1.4 *Sequence Diagram*

Sequence diagram yang menjelaskan semua proses yang ada di dalam sistem. Diagram ini berasal dari proses analisis sistem dan kemudian diperluas hingga mencapai proses terkecil untuk setiap kasus. Pada sub bab ini hanya ditampilkan 3 *sequence diagram*, yaitu *sequence diagram* tambah data pegawai, *sequence diagram* tambah jabatan struktural dan *sequence diagram* mengelola *access token*. Untuk *sequence diagram* yang lainnya dipaparkan pada lampiran B.

4.1.4.1 *Sequence Diagram* Tambah Data Pribadi Pegawai

Proses menambah data pribadi pegawai dapat dilakukan oleh admin yang telah *login*. Pada prosesnya admin mengklik menu daftar pegawai. Sistem akan melakukan *request* ke *UserController* untuk menampilkan data pada halaman daftar pegawai. *UserController* menjalankan beberapa query select untuk meminta data yang dibutuhkan dari model *User*. *Model* tersebut akan mengambil data yang diperlukan dari *database*, *UserController* menjalankan beberapa *query* untuk mengirimkan data ke *database* melalui *model User*. *Sequence diagram* untuk kelola data pegawai dapat dilihat pada gambar 4.2.

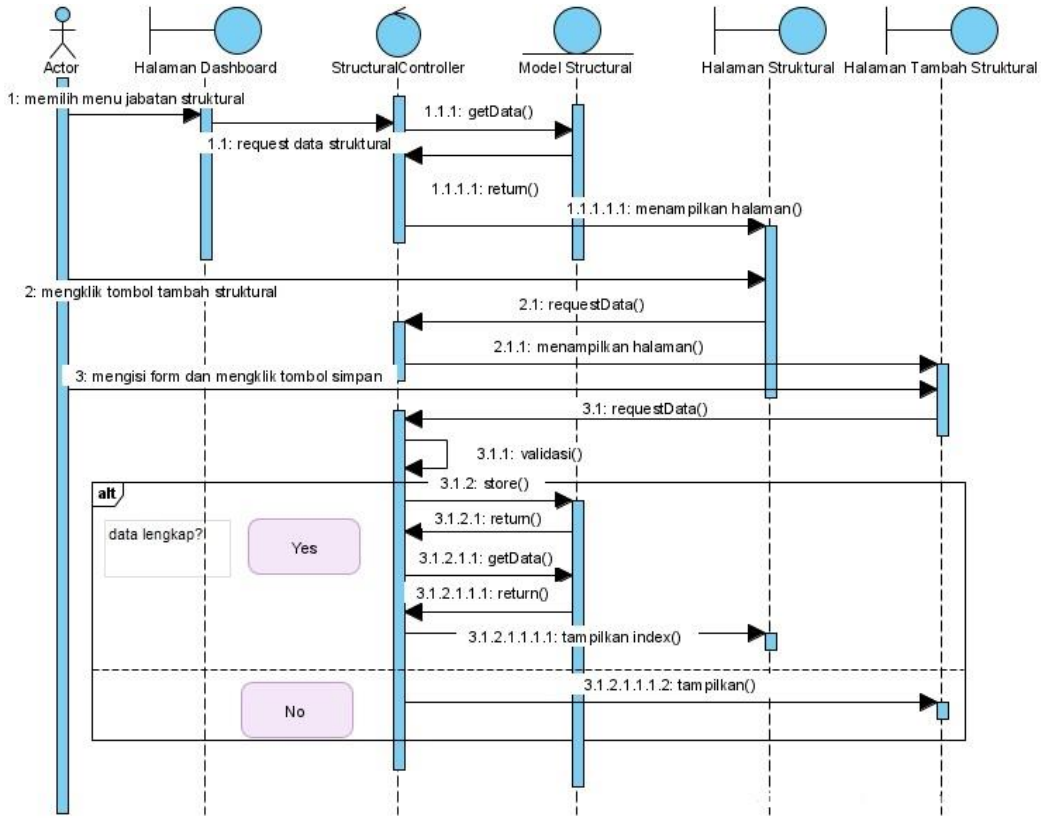


Gambar 3.2 *Sequence diagram* tambah pegawai

4.1.4.2 *Sequence Diagram* Tambah Jabatan Struktural

Proses menambah data jabatan struktural dapat dilakukan oleh *admin.* Sebelum melakukan proses tersebut, *user* harus *login* terlebih dahulu. Pada prosesnya admin mengklik menu jabatan struktural. Sistem akan melakukan *request* ke *StructuralController* untuk menampilkan data pada halaman daftar jabatan struktural. *StructuralController* menjalankan beberapa *query select* untuk

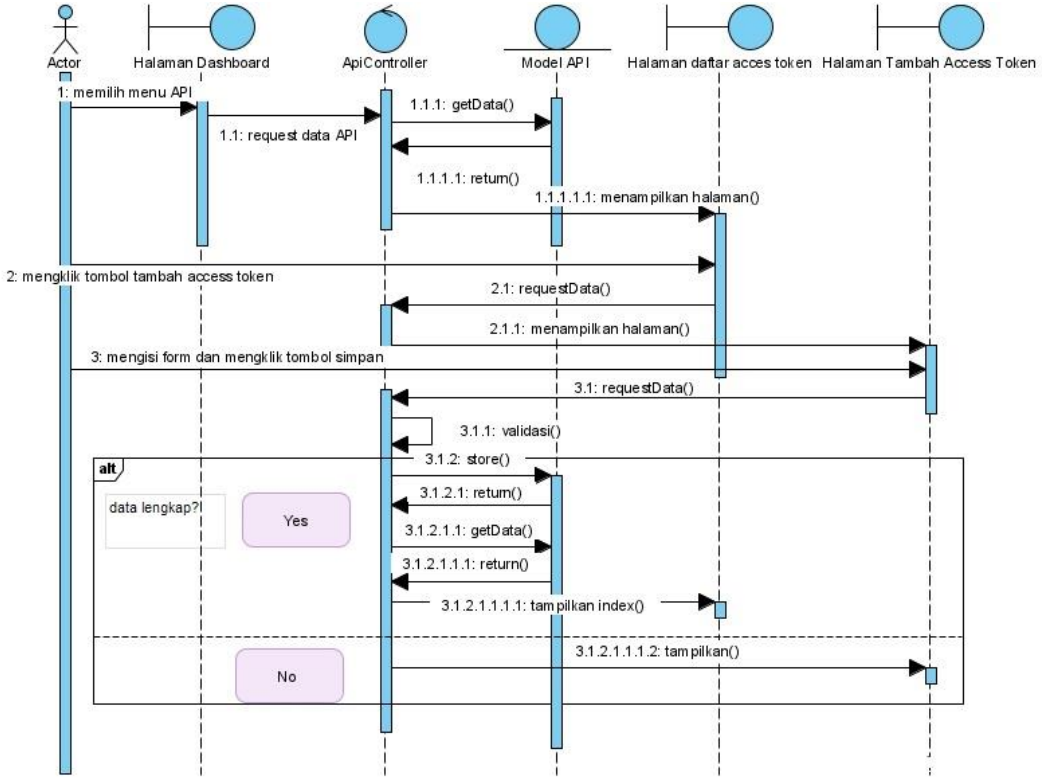
meminta data yang dibutuhkan dari model *Structural*. Model tersebut akan mengambil data yang diperlukan dari *database*, *StructuralController* menjalankan beberapa *query* untuk mengirimkan data ke *database* melalui model *Structural*. *Sequence diagram* untuk kelola jabatan struktural dapat dilihat pada gambar 4.3



Gambar 4.3 Sequence Diagram Tambah Jabatan Struktural

4.1.4.3 *Sequence Diagram* Tambah *Access Token* API *(Application Programming Interface)*

Proses menambah *access token* API dapat dilakukan oleh *admin.* Sebelum melakukan proses tersebut, admin harus *login* terlebih dahulu. Pada prosesnya admin mengklik menu API. Sistem akan melakukan *request* ke *ApiController* untuk menampilkan data pada halaman *access token* API. *ApiController* menjalankan beberapa *query select* untuk meminta data yang dibutuhkan dari model *API*. Model tersebut akan mengambil data yang diperlukan dari *database*, *ApiController* menjalankan beberapa *query* untuk mengirimkan data ke *database* melalui model *API*. *Sequence diagram* untuk kelola API dapat dilihat pada gambar 4.4.



Gambar 4.4 *Sequence Diagram* tambah *access token* API

4.2 Perancangan Sistem

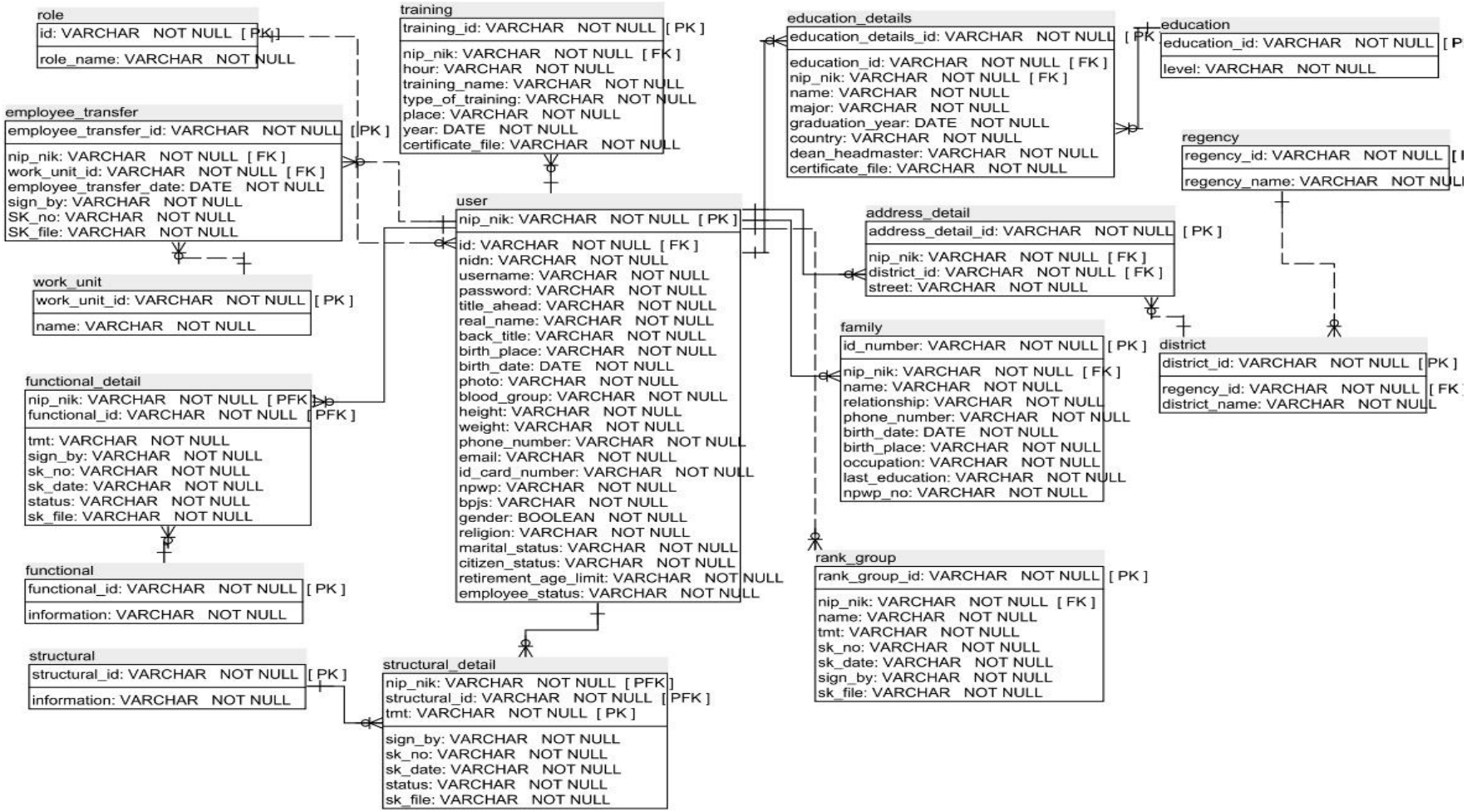
Setelah melakukan tahapan analisis sistem, diperoleh sebuah hasil yang menjadi dasar dan tolak ukur untuk melakukan perancangan. Perancangan yang dimaksud pada penelitian ini ialah perancangan *Entity Relationship Diagram* basis data aplikasi, arsitektur aplikasi, *class diagram* dan perancangan antarmuka.

4.2.1 Perancangan *Database*

Aplikasi pengelolaan data pegawai ini menggunakan *database* sebagai media penyimpanan data. Perancangan *database* dilakukan dengan menentukan entitas serta relasinya satu dengan yang lain. Entitas beserta hubungannya tersebut digambarkan dalam sebuah *entity relationship diagram* (ERD). *Database* berikut terdiri dari 16 tabel. Tabel-tabel tersebut terdiri dari tabel master dan tabel transaksi yang saling memiliki relasi. Untuk lebih jelasnya rancangan relasi entitas dari sistem ini dapat dilihat pada gambar 4.5.

Tabel *users* merupakan tabel yang menyimpan data pengguna. Tabel ini berelasi dengan *roles, structurals, functionals, family, rank_groups, trainings*, *educations* dan *employee_transfers*. Tabel training menyimpan data diklat pegawai,

tabel *employee_transfer* dan *work_unit* untuk menyimpan data mutasi pegawai. Tabel *structural_detail* dan *structural* menyimpan data riwayat jabatan *structural* pegawai. Tabel *functional_detail* dan *functional* untuk menyimpan data riwayat jabatan fungsional pegawai. Kemudian tabel *education* dan *education_details* untuk menyimpan data riwayat pendidikan pegawai. Tabel *rank_group* berguna untuk menyimpan data pangkat golongan pegawai. Sedangkan tabel *family* untuk menyimpan data keluarga pegawai yang bersangkutan.



Gambar 4.5 Perancangan *Database*

4.2.2 Struktur Tabel dan Basis Data

Struktur basis data dan tabel merupakan representasi tabel dan relasinya beserta deskripsi dari seluruh atribut. Pada sub bab ini dijelaskan uraian dari tabel *structurals, employee_transfers, rank_groups, trainings.* Uraian tabel lain dapat dilihat pada bagian lampiran D.

4.2.2.1 Struktur Basis Data dan Tabel *Users*

Tabel *Users* memiliki atribut nip_nik sebagai *primary key* serta atribut role_id sebagai *foreign key* yang menghubungkan tabel ini dengan tabel *user* dan *role*. Tabel ini berguna untuk menyimpan data priadi pegawai. Struktur dari tabel *user* dapat dilihat pada tabel 4.1.

Tabel 4.1 Struktur *Database* Tabel *Users*

Nama atribut	Tipe Data	Keterangan
Nip_nik	Varchar(30)	PK
Role_id	Bigint	FK
nidn	Varchar(30)	
username	Varchar(50)	
password	Varchar(20)	
title_ahead	Varchar(50)	
real_name	Varchar(255)	
back_title	Varchar(50)	
birth_place	Varchar(50)	
birth_date	Date	
photo	Text	
blood_group	Varchar(50)	
height	Varchar(50)	
weight	Varchar(50)	
handicap	Varchar(50)	
phone_number	Varchar(15)	
email	Varchar(50)	
npwp	Varchar(50)	
bpjs	Varchar(50)	
gender	Enum(‘laki-laki’,’perempuan’)	
religion	Varchar(50)	
marital_status	Varchar(50)	
citizen_status	Varchar(50)	
retirement_age_limit	Varchar(50)	
employee_status	Varchar(50)	
remember_token	Varchar(50)	
created_at	timestamp	
updated_at	timestamp	

4.2.2.2 Struktur Basis Data dan Tabel *Structurals*

Tabel *structurals* memiliki atribut *structural_id* sebagai *primary key*, Tabel ini berguna untuk menyimpan data jabatan struktural pegawai. Struktur dari tabel *structurals* dapat dilihat pada tabel 4.2.

Tabel 4.2 Struktur *Database* Tabel Struktural

Nama atribut	Tipe Data	Keterangan
structural_id	varchar(20)	PK
Information	varchar(50)	
created_at	Timestamp	
updated_at	Timestamp	

4.2.2.3 Struktur Basis Data dan Tabel API *(Application Programming Interface)*

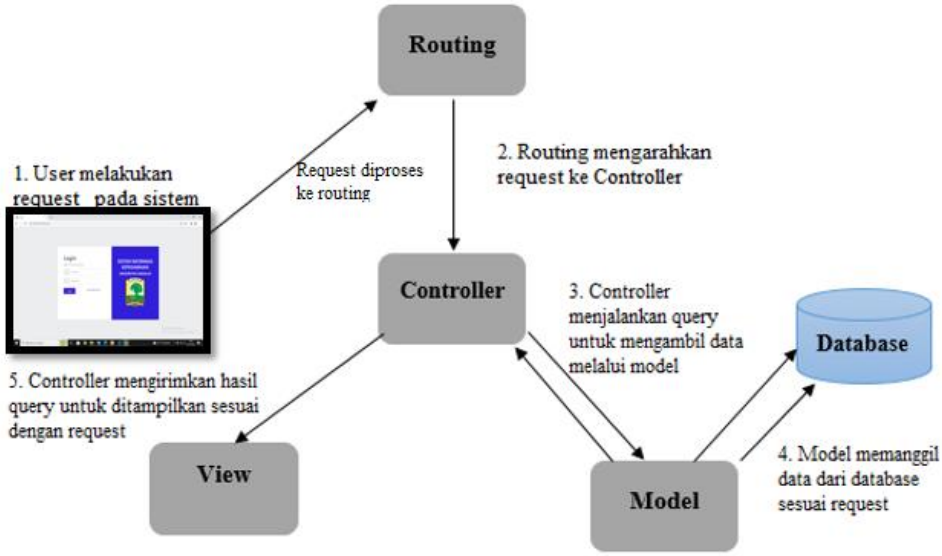
Tabel API memiliki atribut *id* sebagai *primary key*, Tabel ini berguna untuk menyimpan data *access token* API. Struktur dari tabel API dapat dilihat pada tabel 4.3.

Tabel 3.3 Struktur *Database* Tabel API

Nama atribut	Tipe Data	Keterangan
Id	varchar(30)	PK
Name	varchar(30)	
Token	varchar(255)	
created_at	timestamp	
updated_at	timestamp	

4.2.3 Arsitektur Aplikasi

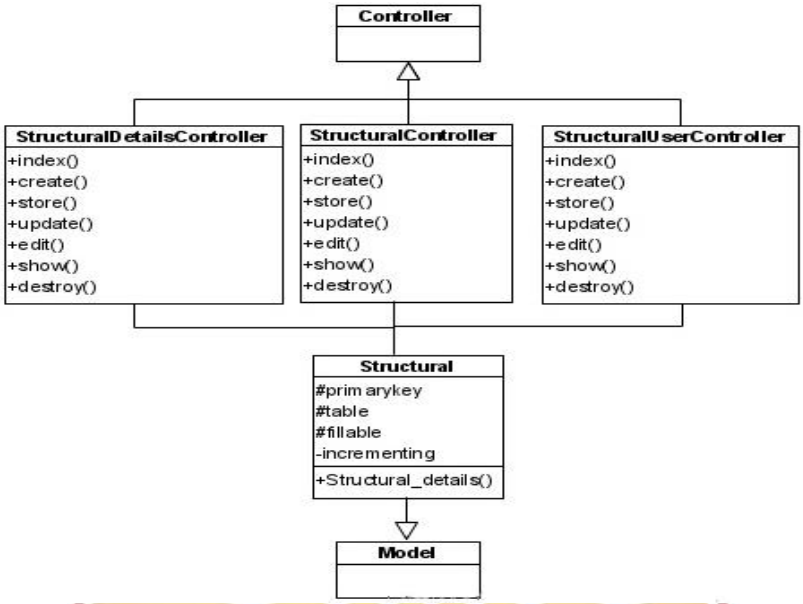
Arsitektur aplikasi yang digunakan dalam perancangan dan pembangunan aplikasi ini yaitu menggunakan arsitektur aplikasi MVC (*Model View Controller*) dengan metode Pemograman Berbasis Objek (PBO) dan menggunakan *framework* Laravel. Arsitektur ini terdiri dari empat komponen yaitu *controller, routing, model,* dan *view*. Ketika pengguna melakukan *request* kepada sistem, *routing* mengarahkan permintaan tersebut ke *controller* yang sesuai. Setelah itu, pengendali menjalankan prosedur atau fungsi tertentu untuk mengirimkan data ke model. Data ini kemudian dikirim ke *database* oleh model, yang kemudian dikirim kembali ke pengendali untuk ditampilkan pada halaman *user*. Rancangan arsitektur sistem informasi pengelolaan data pegawai dapat dilihat pada gambar 4.6.



Gambar 4.6 Arsitektur aplikasi

4.2.4 Class Diagram

Gambaran struktur, deskripsi, dan hubungan antar kelas dalam suatu sistem digambarkan dalam *class diagram*. Tujuan dari perancangan class diagram ini adalah untuk membuat pengkodean program selama proses pembangunan sistem lebih mudah dan lebih terorganisir. Sebuah *class diagram*. akan menunjukkan bagaimana arsitektur sistem yang sedang dirancang (Kendal, 2009). *Class diagram* terdiri dari model dan *controller*, serta *class* yang berisi atribut dan method. Setiap *class* dihubungkan dengan sebuah garis yang disebut asosiasi. *Class diagram* juga menampilkan atribut dan fungsi setiap *class*. Masing-masing *class* memiliki fungsi yang memenuhi kebutuhan sistem.



Gambar 4.7 *Class Diagram* Mengelola Jabatan Struktural Pegawai

Pada bagian ini dijelaskan *class diagram* proses mengelola data jabatan struktural pegawai. *Controller* yang digunakan yaitu *StructuralDetailsController*, *StructuralController* dan *StructuralUserController*. Pada proses ini terdapat kelas model dan *controller*. Dalam proses menambah data jabatan struktural fungsi yang dijalankan terlebih dahulu yaitu *create*(), selanjutnya fungsi store() akan dijalankan saat sistem memproses permintaan. *Query insert* dijalankan, model yang ada berinteraksi dengan *database* dan menyimpan data. *Class diagram* mengelola jabatan struktural pegawai dapat dilihat pada gambar 4.7. Untuk gambar *class diagram* lainnya dapat dilihat pada lampiran D.

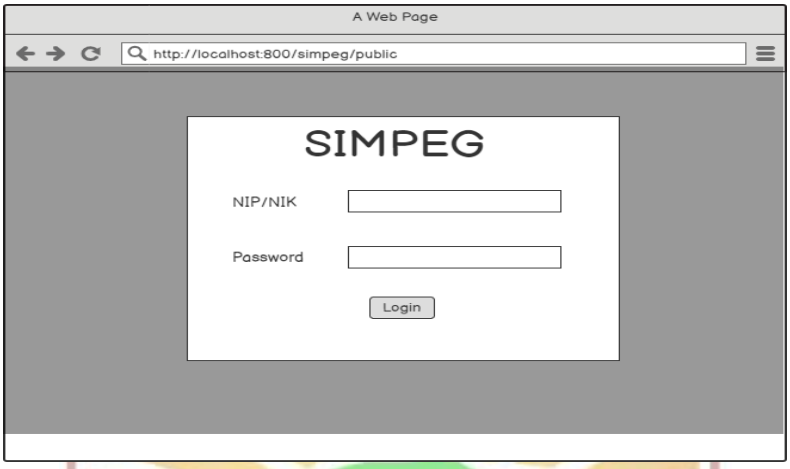
4.2.5 Perancangan Antarmuka

Perancangan antarmuka merupakan gambaran awal dari sistem yang akan dibuat untuk menjadi sarana komunikasi dari sistem ke pengguna. Pada bagian ini akan dijelaskan 4 perancangan antarmuka, yaitu antarmuka halaman login, antarmuka halaman daftar pegawai, antarmuka halaman tambah jabatan fungsional, dan antarmuka *show accesstoken* API. Perancangan antarmuka lainnya dijelaskan pada lampiran F.

4.2.5.1 Perancangan Antarmuka *Login*

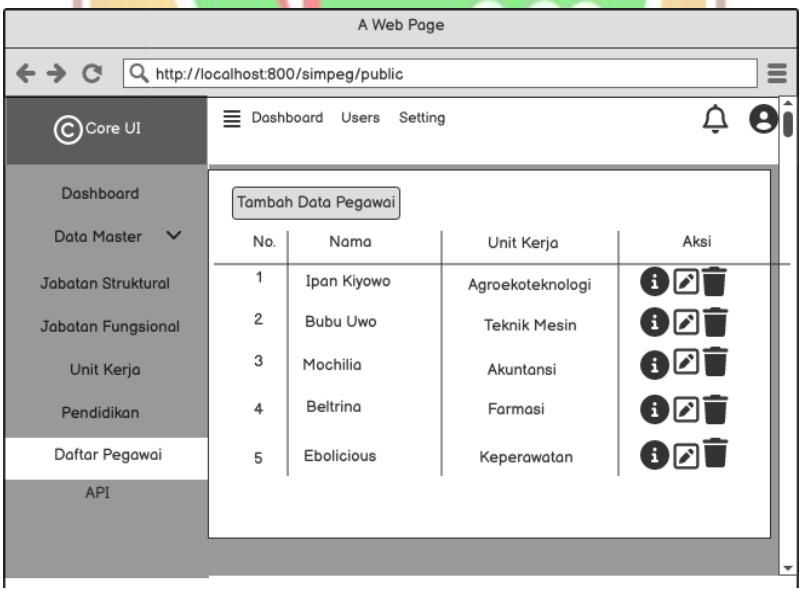
Perancangan antarmuka halaman *login* terdapat *username* dan *password* yang harus diinputkan oleh *user*. *Username* dan *password* yang diisikan harus

sesuai dengan yang data yang terdaftar. Setiap fitur yang ada pada aplikasi harus melakukan *login* terlebih dahulu pada halaman ini. Gambar perancangan antarmuka *login* dapat dilihat pada gambar 4.8.



Gambar 4.8 Antarmuka Login

4.2.5.2 Perancangan Antarmuka Daftar Pegawai

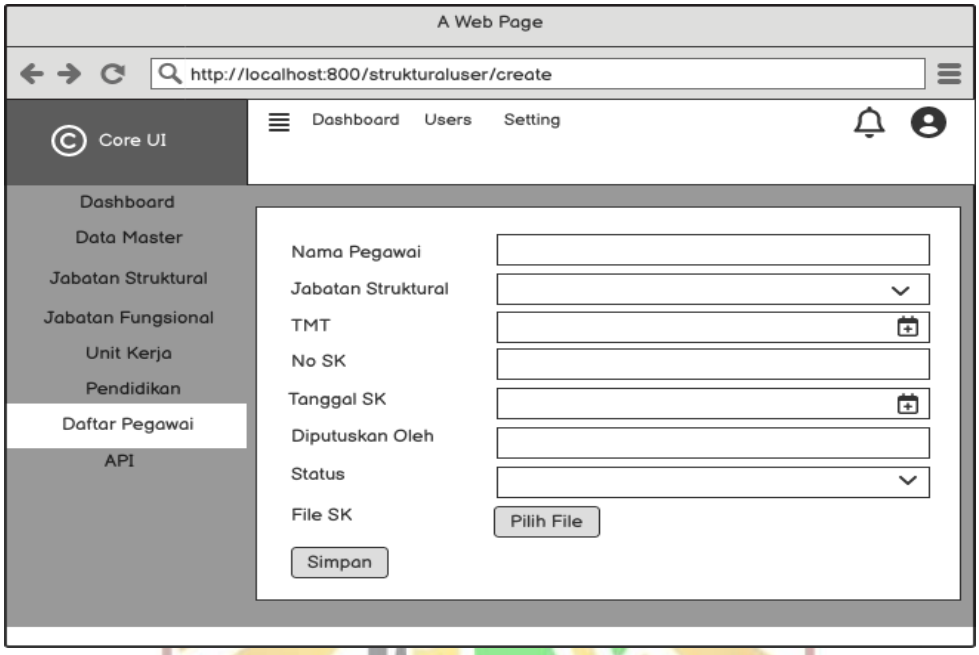


Gambar 4.9 Antarmuka daftar pegawai

Perancangan antarmuka halaman daftar pegawai untuk menampilkan data pegawai yang telah ada. Pada halaman ini admin dapat menambahkan pegawai, melihat data pegawai, mengedit data pegawai dan menghapus data pegawai. Rancangan antrmuka daftar pegawai dapat dilihat pada gambar 4.9.

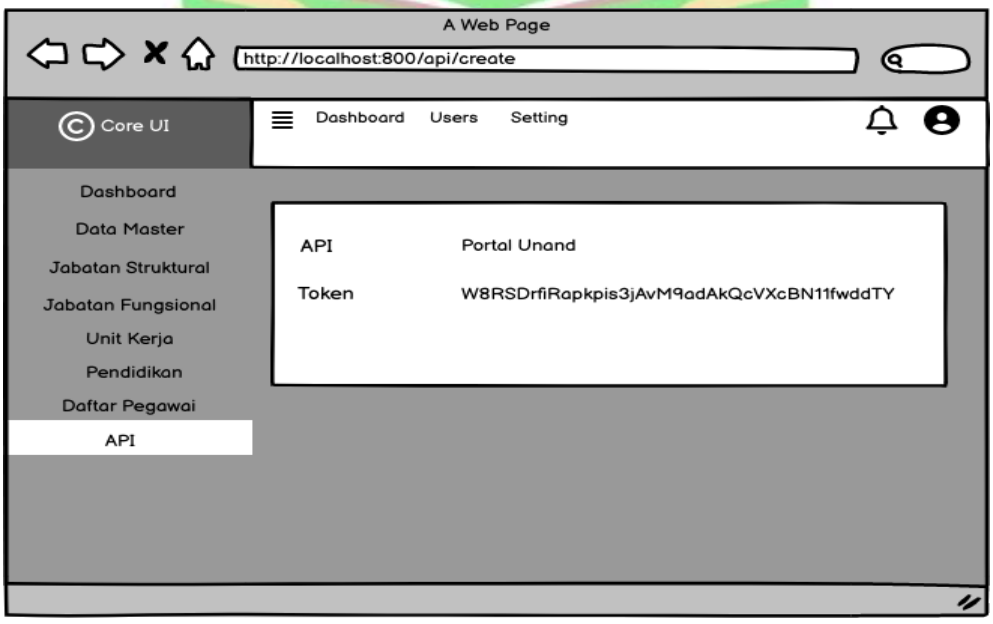
4.2.5.3 Perancangan Antarmuka Tambah Struktural

Halaman tambah struktural dapat diakses oleh pegawai. Pada halaman ini pegawai memasukkan data nama, jabatan struktural, TMT, no SK, tanggal SK, pihak yang meresmikan jabatan, status, dan dan file SK dengan ekstensi pdf. Rancangan antarmuka tambah struktural dapat dilihat pada gambar 4.10.



Gambar 4.10 Antarmuka tambah structural

4.2.5.4 Perancangan Antarmuka *show* API *(Application Programming Interface)*

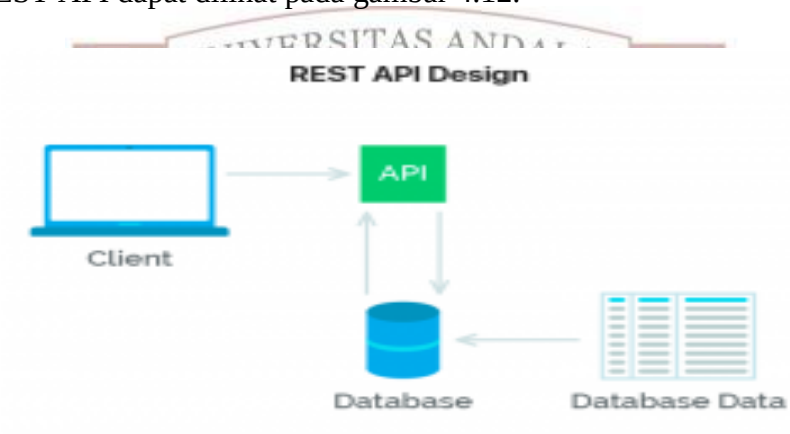


Gambar 4.11 Antarmuka *Show* API

Halaman *show* API dapat diakses oleh pegawai. Pada halaman ini menampilkan nama API dan akses token. Rancangan antarmuka *show* API dapat dilihat pada gambar 4.11.

4.2.6 Perancangan API

Struktur perancangan *web service* REST API yang akan dibuat adalah mengintegrasikan sistem informasi pengelolaan data pegawai Universitas Andalas dengan sistem lain yang membutuhkan data peagwai tersebut. Dengan *website* agar saling berkomunikasi dengan hanya satu pusat data (*datacenter*) saja. Struktur *web service* REST API dapat dilihat pada gambar 4.12.



Gambar 4.12 Perancangan REST API

. Untuk administrasi data input, ubah, dan hapus, biasanya API tidak hanya menggunakan metode POST, PUT, dan DELETE, tetapi juga dapat menggunakan method GET. Kelemahan menggunakan method GET untuk proses administrasi adalah parameter data yang dikirimkan tidak bisa menggunakan data params, tetapi harus menggunakan url params. Perbedaan menggunakan url params dan data params adalah bahwa jika menggunakan url params, nama parameter dan data yang dikirimkan tidak akan sama. Jika menggunakan data params, nama parameter dan data yang dikirimkan akan tidak terlihat pada *URL*. Untuk proses administrasi data, metode GET memiliki kelemahan bahwa data dapat dilihat pada URL dan tidak aman. Sebaliknya, API method GET biasanya digunakan untuk proses lihat data. Untuk memutuskan apakah menggunakan method GET atau POST, perlu diketahui apakah ada perubahan data pada server; jika ada, maka gunakan POST, tetapi jika tidak ada, gunakan GET. Proses pertukaran data melalui API harus aman dan otentikasi. Oleh karena itu, header params diperlukan untuk setiap request. Fungsi

dari header params adalah untuk membedakan setiap request dari yang lain. Parameter kunci yang paling umum digunakan adalah "Content-Type" dan "Authorization". "Content-Type" menunjukkan jenis format respons yang akan diberikan, seperti JSON, XML, HTML, atau teks sederhana. "Authorization" memastikan bahwa permintaan berasal dari user atau asal yang sama, yang membantu menjaga API aman.

BAB V
IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini menjelaskan implementasi sistem sesuai dengan perancangan yang dilakukan pada bab sebelumnya ke dalam bahasa pemrograman. Pengujian kesesuaian sistem yang dibangun dengan alur proses Sistem Informasi Pengelolaan Data Pegawai di Universitas Andalas.

5.1 Implementasi Sistem

Aplikasi sistem informasi pengelolaan data pegawai berbasis *web* dengan fasilitas integrasi berfungsi sebagai sistem yang mengelola data pegawai dan dapat memberikan data pegawai kepada pihak yang membutuhkan *database* pegawai tersebut. Hal ini bertujuan agar memudahkan pihak yang akan menggunakan *database* pegawai sehingga data pegawai tersinkronisasi terhadap semua aplikasi di Universitas Andalas. Fungsional yang dijalankan pada aplikasi diharapkan mampu memberikan efisiensi dan efektifitas terhadap proses pada sistem informasi pengelolaan data pegawai di Universitas Andalas. Implementasi sistem ini dilakukan dengan menggunakan perangkat keras komputer dengan spesifikasi sebagai berikut:

1. Laptop dengan *Processor* Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz.
2. Memori laptop (RAM) 4 GB
3. *Hardisk* berkapasitas 1TB

Spesifikasi perangkat lunak yang digunakan dalam implementasi sistem adalah sebagai berikut.

1. Sistem operasi *Windows* 10 Pro
2. *Web browser* Google Chrome versi 105.0.5195.127
3. *Postman for Windows* versi 8.11.1

Implementasi ini menggunakan bahasa pemograman PHP (*hypertext Preprocessor*) dengan memanfaatkan *Framework* Laravel versi 7.0. *Database* yang digunakan sebagai tempat penyimpanan data dari implementasi sistem ini adalah *database* MySQL.

5.1.1 Pengkodean Program

Pada bagian ini dijelaskan kode program yang dibuat berdasarkan arsitektur aplikasi yang digunakan. Berikut penjelasan terkait fungsi dan kegunaan masing-masing bagian sistem. Pada sub bab ini, kode program yang dijelaskan adalah kode program jabatan struktural dan jabatan fungsional. Penjelasan kode program secara keseluruhan terdapat pada lampiran F.

5.1.1.1 Kode Program *Routing*

Routing menghubungkan *request* ke kontroler terkait *routing* dapat mempermudah pengembangan *website* dan meningkatkan performanya. *Routing* berperan sebagai penghubung antara *user* dengan keseluruhan *framework*.

Routing merupakan bagian yang mengatur perpindahan permintaan (*request*) dari *user*. *Route* dapat mengatur semua perintah yang dideklarasikan dan mengirimkannya ke *controller* serta *method* yang akan digunakan untuk *request* tersebut. Kode program *routing* aplikasi dapat dilihat pada gambar 5.1

```
Route::get('/', 'ProfilDiriUserController@index')>name('profildiriuser.index')-
>middleware('auth');
Route::get('/profil', 'Profilcontroller@profil')-
>name('profil')>middleware('auth');
Route::group(['middleware' => ['role:1']], function () {
// routing untuk role 1 yaitu admin
   Route::get('/profil', 'Profilcontroller@profil')->name('profil');
// route get fungsinya untuk mengarahkan sistem untuk menampilkan controller/fungsi
   Route::resource('pangkatgolongan', 'RankGroupController');
   Route::resource('biodatapegawai', 'BiodataPribadiController');
   Route::resource('fungsional', 'FunctionalController');
   Route::resource('struktural', 'StructuralController');
//route resource untuk untuk membuat CRUD (Create, Read, Update, Destroy)
   Route::resource('education','EducationController');
   Route::resource('unitkerja', 'WorkUnitController');
   Route::resource('biodatakeluarga', 'FamilyController');
   Route::resource('training', 'TrainingController');
   Route::resource('mutasi','EmployeeTransferController');
   Route::resource('datadiri', 'DataDiriController');
   Route::resource( 'fungsionaldetail','FunctionalDetailsController');
   Route::resource( 'strukturaldetail','StructuralDetailsController');
   Route::resource( 'educationdetail','EducationDetailsController');
   Route::resource( 'api','ApiController');
});
Route::group(['middleware' => ['role:2']], function () {
// routing untuk role 2 yaitu user
   Route::get('/profil', 'Profilcontroller@profil')->name('profil');
   Route::resource( 'profildiri','ProfilDiriUserController');
   Route::resource( 'strukturaluser','StructuralUserController');
   Route::resource( 'fungsionaluser','FunctionalUserController');
   Route::resource( 'unitkerjauser','WorkUnitUserController');
   Route::resource( 'educationuser','EducationUserController');
   Route::resource( 'traininguser', 'TrainingUserController');
   Route::resource( 'pangkatgolonganuser', 'RankGroupUserController');
   Route::resource( 'biodatakeluargauser','FamilyUserController');
   Route::resource( 'mutasiuser', 'EmployeeTransferUserController');
 }); });
```

Gambar 5.1 Kode Program *Routing*

5.1.1.2 Kode program *Controller* Jabatan Struktural

Controller adalah bagian yang mengatur seluruh alur dan method pada aplikasi sistem informasi dengan menggunakan arsitektur *model-view-controller* (MVC). Salah satu *controller* yang dibangun adalah StructuralUserController. StructuralUserController berisi atribut dan method yang digunakan untuk memanggil model, mengolah data, dan menampilkan *view*. Kode program StructuralUserController dapat dilihat pada Gambar 5.2.

```
public function index() // untuk mengambil semua data struktural
{
  $strukturaldetails = Structural_Details::all();
  return view('strukturaluser.index', compact('strukturaldetails'));
}
  public function create(Request $request) // Menampilkan form penambahan
{
  $biodatapegawai = User::pluck('real_name','nip_nik');
  $struktural = Structural::pluck('information','structural_id');
  return view('strukturaluser.create', compact('biodatapegawai', 'struktural'));
}
  public function store(Request $request) // Melakukan operasi penambahan
{
  $extension = $request->sk_file->extension();
  if ($extension == "pdf"){
  $fileName = $request->file('sk_file')->getClientOriginalName();
  $strukturaldetails= Structural_Details::create([
  'structural_id'=>$request->input('structural_id'),
  'nip_nik'=>$request->input('nip_nik'),
  'tmt'=>$request->input('tmt'),
  'sign_by'=>$request->input('sign_by'),
  'sk_no'=>$request->input('sk_no'),
  'sk_date'=>$request->input('sk_date'),
  'status'=>$request->input('status'),
  'sk_file' => $request->file('sk_file')->storeAs('sk_file_struktural', $fileName,'public'), ]);
   return redirect('/profildiri');
    }
  else{
   echo "<script>alert('ekstensi file salah')</script>";
   $biodatapegawai = User::pluck('real_name','nip_nik');
   $struktural = Structural::pluck('information','structural_id');
   return view('strukturaluser.create', compact('biodatapegawai', 'struktural')); } }
    public function show($id) //untuk menampilkan data
  {
      $strukturaldetails = Structural_Details::find($id);
      return view('strukturaluser.show', compact('strukturaldetails'));}
    public function edit($structural_id) // Menampilkan form pengubahan
  {
    $biodatapegawai = User::pluck('real_name','nip_nik');
    $strukturaldetails = Structural::find($structural_id);
    $struktural = Structural::pluck('information','structural_id');
  return view('strukturaluser.edit', compact('strukturaldetails','biodatapegawai', 'struktural')); }
  public function update(Request $request, $structural_id)  // Melakukan operasi pembaruan data
  {
    $fileName = $request->file('sk_file')->getClientOriginalName();
    $strukturaldetails= Structural_Details::create([
     'nip_nik'=>$request->input('nip_nik'),
     'tmt'=>$request->input('tmt'),
     'sign_by'=>$request->input('sign_by'),
            'sk_no'=>$request->input('sk_no'),
            'sk_date'=>$request->input('sk_date'),
            'status'=>$request->input('status'),
            'sk_file' => $request->file('sk_file')->storeAs('sk_file', $fileName,'public'),
        ]);
        return redirect()->route('strukturaluser.index'); }
    public function destroy($id) //untuk menghapus data
    {
        $strukturaldetails = Structural_Details::find($id);
        $strukturaldetails->delete();
        return redirect()->route('profildiri.index');  } }
```

Gambar 5.2 Kode program *Controller* Jabatan Struktural

5.1.1.3 Kode Program Model Jabatan Struktural

Model merupakan bagian yang memiliki fungsi untuk memanggil data dari *database* serta mengirimkannya ke *controller* dengan menggunakan *syntax* SQL (*Structured Query Language*). Salah satu model yang dibangun adalah model Structural_Details untuk memanggil dan megirimkan entitas pada *database*. Pada model ini terdapat *class* Structural_Details yang mewakili entitas *structural*. Didalamnya terdapat fungsi yang menggambarkan relasi antara class Structural_Details dengan class yang lain. Kode program model Structural_Details dapat dilihat pada Gambat 5.3.

```
class Structural_Details extends Model
{
  protected $table = 'structural_details'; //nama table yang digunakan
  protected $primaryKey = 'structural_id'; //nama id table yang digunakan
  public $incrementing = false;
  protected $fillable = ['nip_nik','structural_id','tmt', 'sign_by', 'sk_no', 'sk_date'
,'status','sk_file' ];
  public function structural(){
  return $this>belongsTo(Structural::class, 'structural_id','structural_id')}
  public function user(){
  return $this->belongsTo(User::class, 'nip_nik', 'nip_nik'); // belongs to digunakan
di tabel penyambung (yang punya foreign key)
}
```

Gambar 5.3 Kode Program Model Jabatan Struktural

5.1.1.4 Kode Program *View* Struktural

View merupakan antar muka aplikasi akan yang berinteraksi langsung dengan *user*. Data dari *database* dan perintah dari *controller* ditampilkan pada *view*. Pada *framework* laravel, *view* dibangun menggunakan *template engine* yaitu blade. Kelebihan menggunakan *blade* adalah menjadikan *layout* yang didesain dapat digunakan di tampilan lain sehingga menyediakan konsistensi desain dan struktur selama proses pengembangan sistem. Salah satu *view* yang dibangun adalah *view blade* yang berfungsi untuk menampilkan halaman Structural_Details(). Kode program *view* dapat dilihat pada Gambar 5.4.

```
<div class="card-header">
  <div class="d-flex">
    <a href="{{ route('struktural.create') }}" class="btn btn-
primary">Tambah Struktural</a>
    </div> </div>
      <div class="card-body">
         <table class="table table-responsive-sm table-striped">
          <thead>
           <tr>
```

Gambar 5.4 View *Index Stuktural*

```
        <th class="text-center">No</th>
<th class="text-center">Jabatan Struktural</th>
<th class="text-center">Aksi</th>
  </tr> </thead> <tbody>
   @foreach ($struktural as $strukturals)
     <tr>
       <td class="text-center">{{ $loop->iteration }}</td>          //data yang ditampilkan
        <td class="text-center">{{ $strukturals->information}}</td>
          <td class="text-center">
            <a href="{{ route('struktural.show',$strukturals-
>structural_id) }}"class="btn btn-info" id="editButton" data-target="#editPegawai">
              <i class="cil-zoom-in"></i>   //button show data
            </a>
        <a href="{{ route('struktural.edit',$strukturals-
>structural_id) }}"class="btn btn-warning" id="editButton" data-target="#editPegawai">
    <i class="cil-pencil"></i> //button edit data
        </a>
<form action="{{ route('struktural.destroy', $strukturals->structural_id) }}"
 method="post" onclick="return confirm('Anda yakin menghapus data ?')"class="d-inline">
     @csrf
     @method('DELETE')
    <button class="btn btn-youtube">
     <i class="cil-trash"></i></button> //button delete data
     </form>
       </td>
         </tr>
           @endforeach
           </tbody>
            </table>
               </div>
```

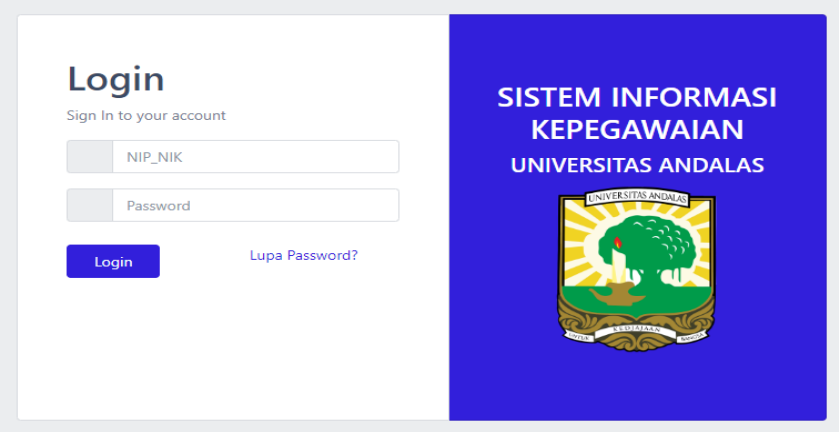
Gambar 5.4 View *Index Stuktural* (lanjutan)

5.1.2 Implementasi Antarmuka Aplikasi

Implementasi antarmuka aplikasi adalah tampilan dari aplikasi yang telah dibangun. Implementasi antarmuka aplikasi terdiri dari beberapa halaman yang ditampilkan sesuai dengan menu yang ada pada pengguna. Pada subbab ini hanya dijelaskan beberapa implementasi antarmuka aplikasi pada aplikasi yang telah dibangun. Untuk implementasi antarmuka lainnya dapat dilihat pada lampiran G.

4.1.1.1 Halaman *Login*

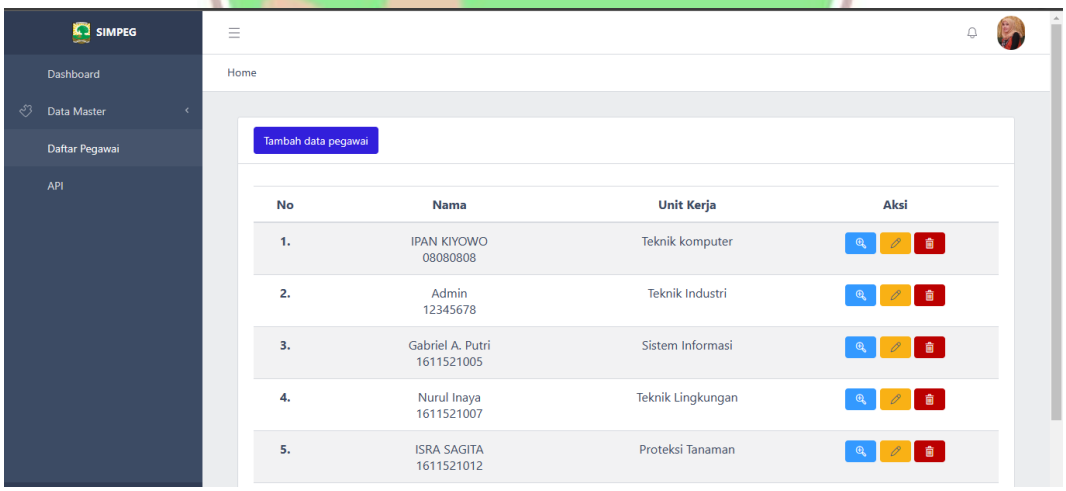
Halaman *login* adalah halaman *web* pertama yang ditampilkan ketika pengguna membuka aplikasi *web*. Hal ini dimaksudkan agar aplikasi ini hanya dapat digunakan oleh pengguna yang terlibat dalam sistem informasi pengelolaan data pegawai dan memiliki hak akses Pengguna *login* dengan memasukkan NIP/NIK dan *password* yang telah terdaftar pada sistem lalu menekan tombol *Login.* Jika NIP/NIK atau *password* yang dimasukkan salah maka akan tampil pemberitahuan. Ketika pengguna berhasil masuk maka aplikasi akan mengarahkan pengguna ke halaman *home*. Antarmuka halaman *login* dapat dilihat pada Gambar 5.5.



Gambar 5.5 Tampilan Halaman *Login*

4.1.1.2 Halaman Daftar Pegawai

Halaman kelola pegawai merupakan halaman yang bisa diakses oleh admin. Halaman ini memiliki fitur *create, read, update, delete* (CRUD) untuk kelola data pegawai. Pada halaman kelola data pegawai ini terdapat dua *field* tabel yaitu nomor, nama, unit kerja, dan aksi. Pada kolom aksi terdapat tiga tombol yaitu edit data, *show* data dan hapus data. Bagian kiri atas dari halaman kelola pegawai ini terdapat *button* tambah data pegawai yang berguna untuk menambah data mahasiswa. Tampilan halaman data pegawai dapat dilihat pada gambar 5.6.

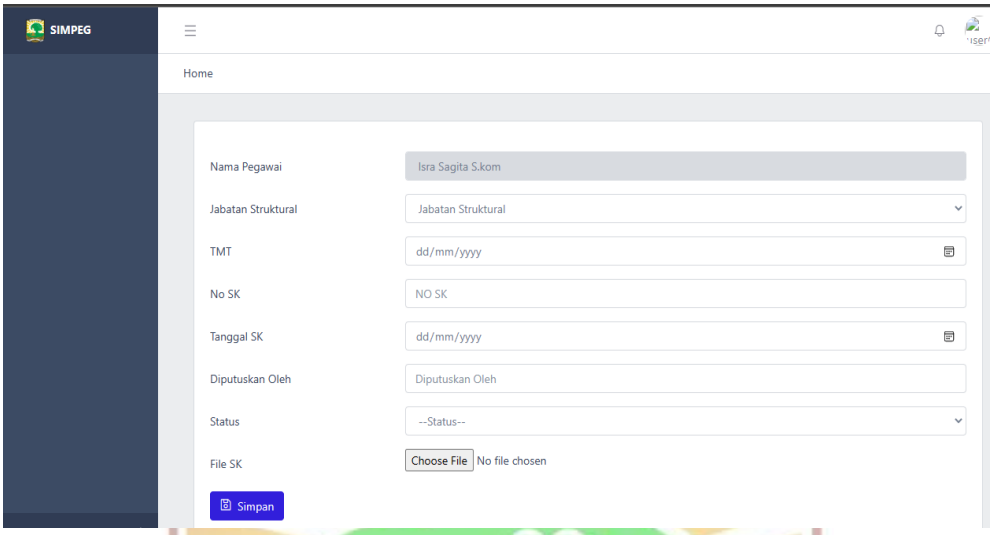


Gambar 5.6 Tampilan Halaman Daftar Pegawai

4.1.1.3 Halaman Tambah Jabatan Struktural

Halaman tambah jabatan struktural diakses oleh pegawai setelah berhasil login ke sistem. Pegawai mengklik button tambah jabatan struktural. Pada halaman ini terdapat beberapa list yang harus diisioleh pegawai, setelah semuanya lengkap

pegawai kemudian mengklik tombol simpan. Halaman tambah jabatan strktural dapat dilihat pada gambar 5.7.



Gambar 5.7 Halaman Tambah Jabatan Struktural

5.2 Pengujian Sistem

Pada tahap pengujian sistem ini, proses untuk memeriksa apakah aplikasi yang dibangun telah berjalan sesuai dengan perancangan sistem yang telah dibuat sebelumnya. Pengujian dilakukan dengan metode black box testing, yang berfokus pada melihat hasil eksekusi melalui data uji dan memeriksa fungsionalitas aplikasi. Pengujian ini dilakukan oleh admin, dan pegawai.

5.2.1 Fokus Pengujian

Fokus pengujian aplikasi ini menggunakan data uji berdasarkan data dan fungsional aplikasi yang telah dibangun. Fokus pengujian dapat dilihat pada tabel 5.1

Tabel 5.1 Fokus Pengujian

No	Item Uji	User	Detail Pengujian
1	Autentifikasi	Admin, Pegawai	*Login, logout*
2	Mengelola data pribadi	Admin, Pegawai	Lihat, tambah, edit, hapus
3	Mengelola data pegawai	Admin	Lihat, tambah, edit, hapus
4	Mengelola jabatan fungsional	Admin, Pegawai	Lihat, tambah, edit, hapus
5	Mengelola jabatan struktural	Admin, Pegawai	Lihat, tambah, edit, hapus

6	Mengelola data riwayat pendidikan	Admin, Pegawai	Lihat, tambah, edit, hapus
7	Mengelola diklat	Admin, Pegawai	Lihat, tambah, edit, hapus
8	Mengelola pangkat golongan	Admin, Pegawai	Lihat, tambah, edit, hapus
9	Mengelola mutasi pegawai	Admin, Pegawai	Lihat, tambah, edit, hapus
10	Mengelola data keluarga pegawai.	Admin, Pegawai	Lihat, tambah, edit, hapus
11	Mengelola unit kerja.	Admin	Lihat, tambah, edit, hapus
12	Mencetak data pegawai.	Admin	Lihat, tambah, edit, hapus
13	Menampilkan data pegawai di *tools* Postman	Admin	Lihat
14	Menampilkan data master struktural di *tools* Postman	Admin	Lihat
15	Melihat seluruh data pegawai di Postman	Admin	Lihat
16	Melihat data pegawai berdasarkan NIK tertentu di Postman	Admin	Lihat
17	Melihat data master jabatan struktural di Postman	Admin	Lihat
18	Melihat data master jabatan fungsional di Postman	Admin	Lihat
19	Melihat data master unit kerja	Admin	Lihat

5.2.2 Kasus hasil Pengujian

Pada bagian membahas kasus dan hasil pengujian aplikasi. Pengujian ini didasarkan pada fokus pengujian yang telah ditentukan sebelumnya dan memeriksa fungsionalitas aplikasi dengan mempertimbangkan masukan ke dalam sistem dan outputnya. Pada sub bagian berikut dijabarkan beberapa kasus hasil pengujian.

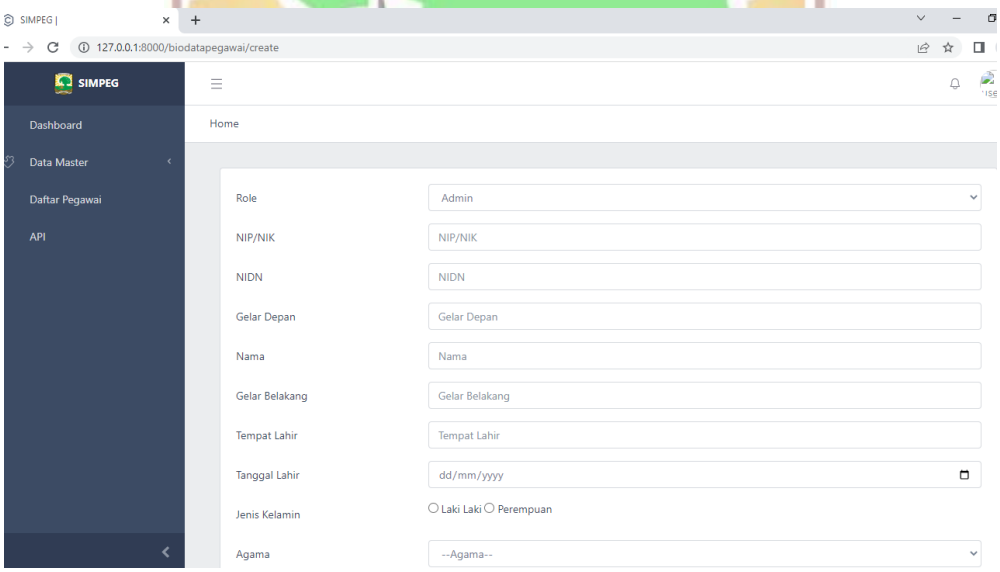
5.2.2.1 Pengujian tambah data pegawai

Pada pengujian ini dilakukan tambah data pendidikan pegawai pada aplikasi. Hasil pengujian dapat dilihat pada Tabel 5.2.

Tabel 5.2 Pengujian Tambah Pegawai

Kasus dan hasil pengujian (sukses)	
Data masukan	Data pribadi pegawai
Hasil yang diharapkan	Data tersimpan dan sistem menampilkan halaman data pegawai
Pengamatan	Data tersimpan dan sistem menampilkan halaman informasi data yang diisi sudah berhasil di-input-kan
Hasil	Sesuai dengan fungsional yang dibangun

Pengujian dilakukan setelah admin melakukan *login* dan masuk ke halaman sistem pengelolaan data pegawai. Kemudian admin mengklik tombol tambah pegawai, lalu sistem menampilkan *form* tambah pegawai yang akan diisi oleh admin. Jika berhasil maka sistem menampilkan kembali halaman daftar pegawai dengan notifikasi berhasil dan data pada tabel ter-*update*. Hasil dari pengujian aplikasi berupa tambah pendidikan pegawai dapat dilihat pada gambar 5.8 dan 5.9.



Gambar 5.8 Form Tambah Data Pegawai

Gambar 5.9 Halaman data pegawai berhasil ditambahkan

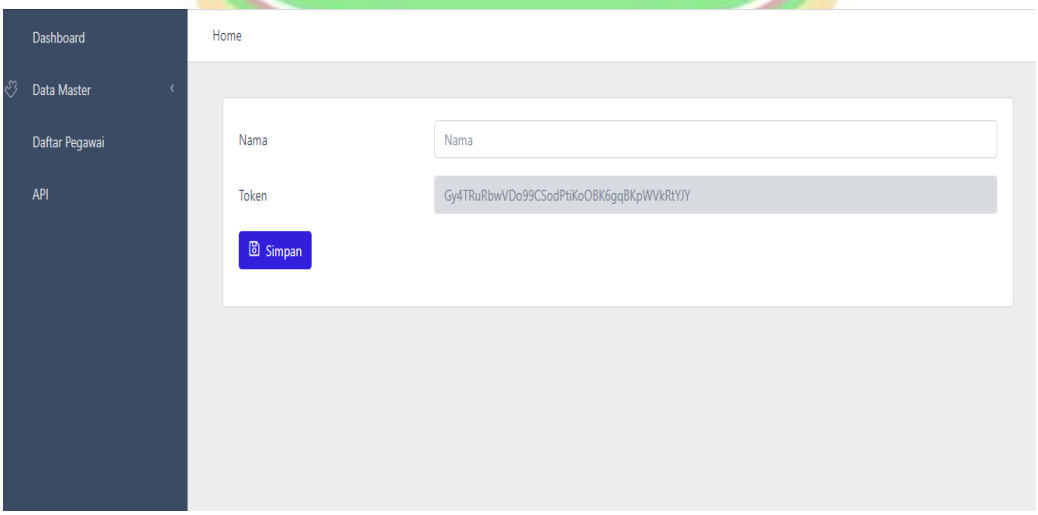
5.2.2.2 Pengujian Menambahkan *Access Token API*

Pada pengujian ini dilakukan proses menambahkan *access token API.* Hasil pengujian dapat dilihat pada tabel 5.3.

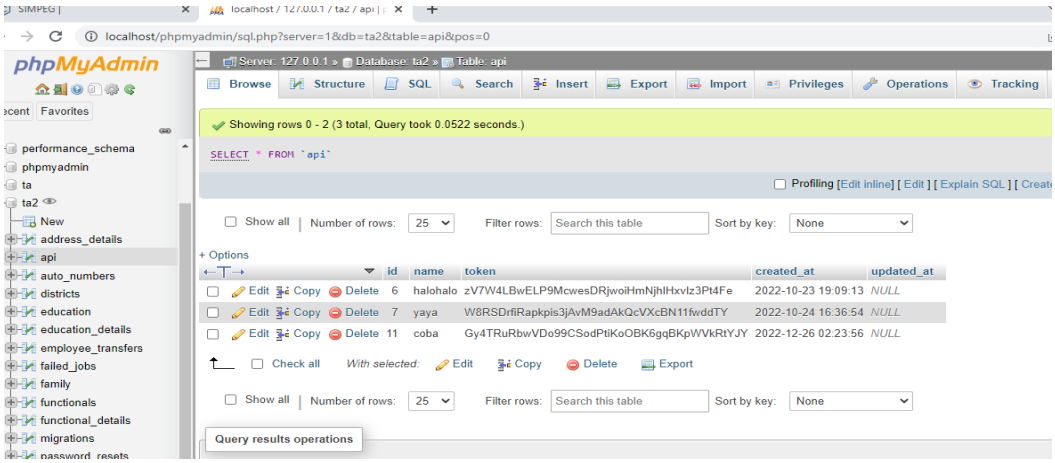
Tabel 5.3 Pengujian Data Pegawai Tampil di *API*

Kasus dan hasil pengujian (sukses)	
Data masukan	Nama sistem
Hasil yang diharapkan	*Access Token API*
Pengamatan	Data *access token* yang telah tersimpan di *database* akan tampil di daftar API pada sistem kelola data pegawai
Hasil	Sesuai dengan fungsional yang dibangun

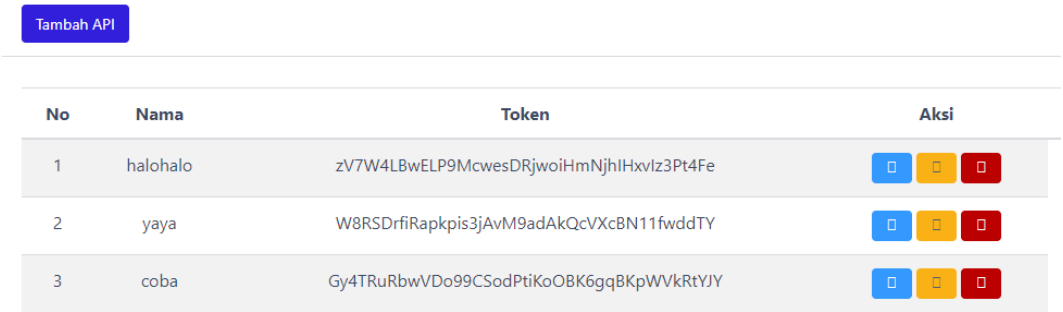
Pengujian dilakukan setelah admin melakukan *login* dan masuk ke halaman pengelolaan data pegawai, admin mengklik tombol tambah API. Maka akan muncul form tambah *acces token API* yang dibutuhkan. Lalu inputkan nama yang diinginkan untuk menyimpan token API. Kemudian *access token* tersebut akan tersimpan di *database*. Jika berhasil maka sistem kelola data pegawai akan menampilkan kode *access token API*. Hasil dari pengujian aplikasi berupa tampilan data pegawai di Postman dapat dilihat pada gambar 5.10 5.11 dan 5.12.



Gambar 5.10 Halaman tambah API



Gambar 5.11 Halaman Database API



Gambar 5.12 Halaman tampilan daftar *access token*

5.2.2.3 Pengujian Menampilkan Data Master Struktural Di API

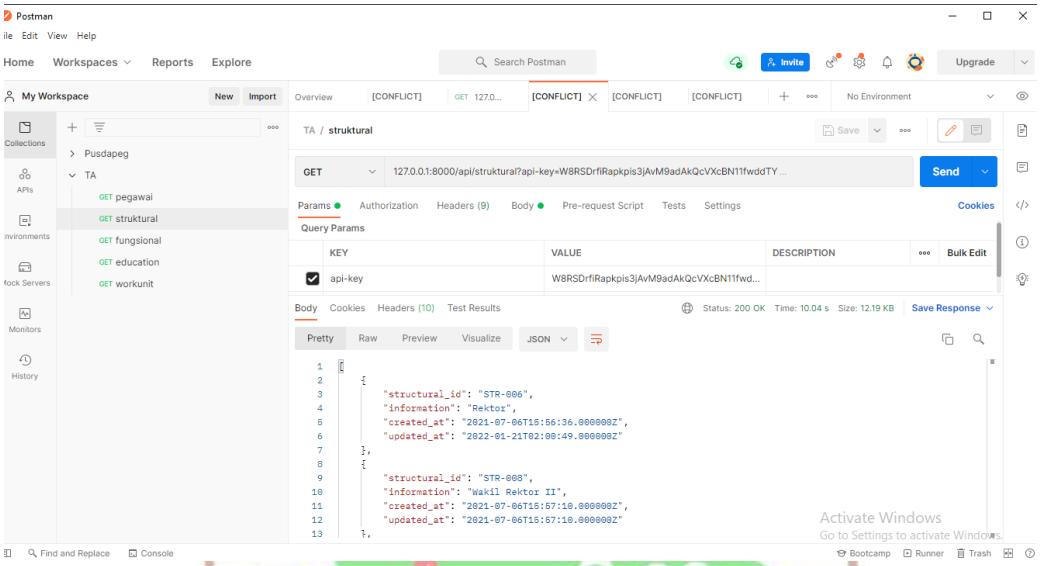
Pada pengujian ini dilakukan proses menampilkan data master struktural di API pada *tools postman*. Hasil pengujian dapat dilihat pada Tabel 5.3.

Tabel 5.3 Pengujian Menampilkan Data Master Struktural di *API*

Kasus dan hasil pengujian (sukses)	
Data masukan	Access token
Hasil yang diharapkan	Data master struktural akan tampil di tools Postman
Pengamatan	Data master struktural yang telah tersimpan di database sebeumnya akan tampil di *tools* Postman.
Hasil	Sesuai dengan fungsional yang dibangun

Pengujian dilakukan setelah admin memiliki *access token API l* dan menginputkan ke *params* bagian *value*. Kemudian admin mengklik tombol *send*

pada pojok kanan atas. Maka akan tampil hasil dari pengujian berikut yaitu data master struktural di *tools* Postman. Hasil dari pengujian aplikasi berupa tampilan data master struktural di Postman dapat dilihat pada gambar 5.13.



Gambar 5.13 Halaman tampilan data master struktural di *tools postman*

5.2.2.3 Pengujian Menampilkan Data Pegawai berdasarkan NIP/NIK tertentu

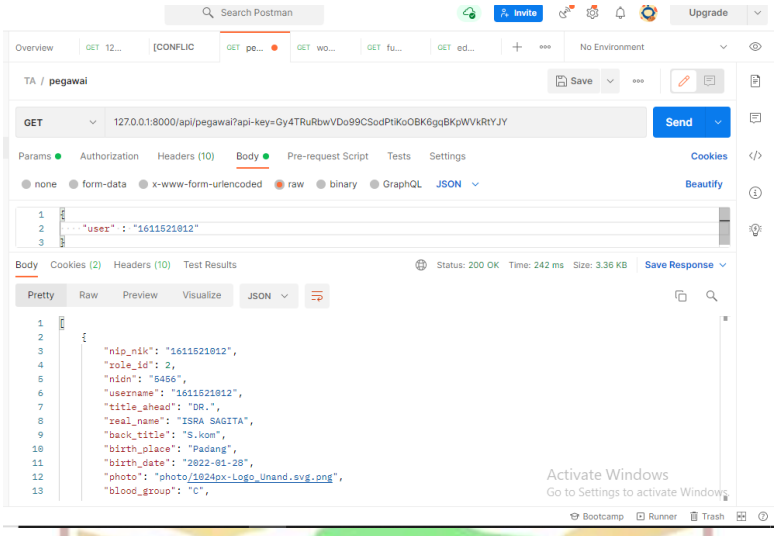
Pada pengujian ini dilakukan proses menampilkan data master struktural di API pada *tools postman*. Hasil pengujian dapat dilihat pada Tabel 5.3.

Tabel 5.3 Pengujian Menampilkan Data Pegawai Berdasarkan NIP/NIK tertentu

Kasus dan hasil pengujian (sukses)	
Data masukan	Access token
Hasil yang diharapkan	Data pegawai tertentu akan tampil di tools Postman
Pengamatan	Data yang telah tersimpan di database sebeumnya akan tampil di *tools* Postman.
Hasil	Sesuai dengan fungsional yang dibangun

Pengujian dilakukan setelah admin memiliki *access token API* dan menginputkan ke *params* bagian *value*. Kemudian admin menginputkan NIP/NIK yang akan ditampilkan ke bagian body. Lalu admin mengklik tombol *send* pada pojok kanan atas. Maka akan tampil hasil dari pengujian berikut yaitu data pegawai berdasarkan NIP/NIK yang diinputkan.pada *tools* Postman. Hasil dari pengujian

aplikasi berupa tampilan data master struktural di Postman dapat dilihat pada gambar 5.14.



Gambar 5.14 Halaman tampilan NIP/NIK tertentu pada *tools postman*

5.2.3 Kesimpulan Hasil Pengujian

Berdasarkan pengujian yang telah dilakukan, diperoleh hasil antara perancangan dengan keluaran sistem. Selama pengujian tidak ditemukannya kegagalan dalam setiap proses pada masing-masing fungsional. Dapat disimpulkan bahwa sistem informasi pengelolaan data pegawai berbasis integrasi telah berjalan sesuai dengan fungsional yang dirancang. Hasil pengujian secara lengkap dijelaskan pada lampiran I. Kesimpulan dari hasil pengujian sistem dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil pengujian

No	Item Uji	User	Detail Pengujian	Hasil
1	Autentifikasi	Admin, Pegawai	*Login, logout*	Sesuai
2	Mengelola data pribadi	Admin, Pegawai	Lihat, tambah, edit, hapus	Sesuai
3	Mengelola data pegawai	Admin	Lihat, tambah, edit, hapus	Sesuai
4	Mengelola jabatan fungsional	Admin, Pegawai	Lihat, tambah, edit, hapus	Sesuai
5	Mengelola jabatan struktural	Admin, Pegawai	Lihat, tambah, edit, hapus	Sesuai
6	Mengelola data riwayat pendidikan	Admin, Pegawai	Lihat, tambah, edit, hapus	Sesuai

7	Mengelola diklat	Admin, Pegawai	Lihat, tambah, edit, hapus	Sesuai
8	Mengelola pangkat golongan	Admin, Pegawai	Lihat, tambah, edit, hapus	Sesuai
9	Mengelola mutasi pegawai	Admin, Pegawai	Lihat, tambah, edit, hapus	Sesuai
10	Mengelola data keluarga pegawai.	Admin, Pegawai	Lihat, tambah, edit, hapus	Sesuai
11	Mengelola unit kerja.	Admin	Lihat, tambah, edit, hapus	Sesuai
12	Mencetak data pegawai.	Admin	Lihat, tambah, edit, hapus	Sesuai
13	Menampilkan data pegawai di *tools* Postman	Admin	Lihat	Sesuai
14	Menampilkan data master struktural di *tools* Postman	Admn	Lihat	Sesuai
15	Melihat seluruh data pegawai di Postman	Admin	Lihat	Sesuai
16	Melihat data pegawai berdasarkan NIK tertentu di Postman	Admin	Lihat	Sesuai
17	Melihat data master jabatan struktural di Postman	Admin	Lihat	Sesuai
18	Melihat data master jabatan fungsional di Postman	Admin	Lihat	Sesuai
19	Melihat data master unit kerja	Admin	Lihat	Sesuai

Dari tabel hasil pengujian aplikasi dapat diketahui bahwa semua fungsional yang ada sesuai dengan sistem yang dibangun dan dari analisis yang telah dilakukan maka aplikasi ini memiliki beberapa kelebihan yaitu.

1. Sistem informasi pengelolaan data pegawai berbasis *web* memudahkan pihak yang berkepentingan dapat mempergunakan aplikasi ini untuk memanajemen kepegawaian pada instansi yang membutuhkannya.
2. Memudahkan pihak Universitas Andalas dalam mensinkronkan *database* pegawai sehingga data yang digunakan oleh semua pihak Unand terintegrasi.
3. Data yang tersimpan merupakan data yang paling terbaru.

BAB VI
PENUTUP

Bab ini menjelaskan tentang kesimpulan dan saran pada laporan tugas akhir ini. Kesimpulan merupakan pencapaian dari tujuan yang telah ditetapkan pada penelitian, sedangkan saran adalah harapan berkaitan dengan penelitian kedepannya.

6.1. Kesimpulan

Pembangunan sistem informasi pengelolaan data pegawai berbasis *web* yang terintegrasi telah selesai dibangun. Metode pegembangan sistem yang digunakan untuk membangun sistem ini menggunakan model waterfall yaitu pengumpulan dan analisis data, perancangan, implementasi dan pengujian sistem. Data dan informasi yang berhubungan dengan penelitian dikumpulkan kemudian dianalisis sehingga menghasilkan 20 fungsional sistem dan tiga aktor yaitu admin, pegawai, dan sistem yang memerlukan *database* tersebut. Sistem ini dibangun pada *platform web* menggunakan bahasa pemrograman PHP, *database MySQL* dan *framework* Laravel. Pengujian dilakukan dengan metode *blackbox testing*. Pada pengujian ini didapatkan hasil bahwa fungsional yang terdapat pada sistem ini tersedia dan telah berjalan sesuai dengan kebutuhan dan hasil rancangan. Dengan demikian, permasalahan yang ada pada proses pengelolaan data pegawai ini telah berhasil diselesaikan.

Dengan pembangunan sistem ini semua proses pengelolaan data pegawai memudahkan pegawai untuk mengelola data dari semua pegawai di Universitas Andalas. Pegawai tidak perlu lagi memindahkan data secara manual. Sistem ini juga bisa digunakan oleh sistem lain di Universitas Andalas yang membutuhkan *database* pegawai Universitas Andalas. Sistem ini sudah menyediakan API *(Application Programming Interface)* sehingga memudahkan integrasi data antar sistem tersebut. Integrasi diperlukan agar tidak terjadi redudansi data dari masing masing sistem di Unand yang menggunakan data pegawai. Sistem ini membuat pekerjaan pegawai lebih efektif dan efisien.

6.2 Saran

Aplikasi sistem informasi pengelolaan data pegawai Universitas Andalas berbasis web yang terintegrasi ini masih membutuhkan pengembangan lebih lanjut yang sejalan dengan kebutuhan user fungsional kedepannya. Diharapkan aplikasi ini dapat dikembangkan dan terus diperbarui dengan menambah fungsional seperti mengelola SKP dan remun sehingga keseluruhan proses semakin mudah. Serta diharapkan agar sistem dikembangkan dalam bentuk mobile sehingga dapat dijalankan di berbagai *platform* (*multiplatform*) sehingga lebih memudahkan dalam pengelolaan data kedepannya.

DAFTAR PUSTAKA

Andi Sunyoto. 2007. Pemrograman Database dengan Visual Basic dan Microsoft SQL 2000. Yogyakarta: Andi Offset.

Amalia, Santoso, Rihandoyo. (2012). Evaluasi Sistem Informasi Manajemen Kepegawaian Berbasis Electronic Government Di Badan Kepegawaian Daerah (Bkd) Kabupaten Wonosobo. *Indonesian Journal of Public Policy and Management Review*.1.

Aminudin. (2015). *Cara Efektif Belajar Framework Laravel*. Yogyakarta: Lokomedia.

Beni A.P., Astria H., Akmal J. (2018). *Perancangan Application Programming Interface (API) Berbasis Web Menggunakan Gaya Arsitektur Representational State Transfer (REST) Untuk Pengembangan Sistem Informasi Administrasi Pasien Klinik Perawatan Kulit.* Ilmu Komputer. Universitas Lampung.

Cahyanti, A. N., dan Purnama, B. E. (2012). *Pembangunan Sistem Informasi Manajemen Puskesmas Pakis Baru Nawangan*. Journal Speed, Vol 4 No 4.

Eko, S. "Laravel VS Code Igniter". Internet: https://teknik-informatika s1.stekom.ac.id/informasi/baca/Laravel-VS-Code Igniter/66c4d1ec6dadbb7073bb7c5132e479ca4d2b59cf, [16 Mei 2023].

Faizal, M. dan Putri, S. L. (2017). *Sistem Informasi Pengolahan Data Pegawai Berbasis Web (Studi Kasus di PT Perkebunan Nusantara VIII Tambaksari).* STMIK Subang.

Haryadi. S, "Mengenal RESTful API," 2016.

Hamidi, F., Meshkat, M., Rezaee, M., & Jafari, M. (2011). *Information Technology In Education. Procedia Computer Science*, 3, 369–373.

Heldiansyah, Noor A., Syarifah S.A., Sistem Informasi Kepegawaian Pada Mtsn Mulawarman Banjarmasin Berbasis Web. Jurnal Positif, Volume 2, No.1, November 2016: 28 – 33.

Hendri A.P, Dede K. Yosep S., Aplikasi Pengelolaan Data Pegawai Berbasis REST API untuk Transfer Data Real Time dengan Framework Codeigniter. Institut Teknologi Garut. Garut. 2022.

Kadir, A. (2008). *Tuntunan Praktis Belajar Database Menggunakan MySQL.*

Yogyakarta: C.V Andi Offset.

Kevin S, Magdalena A. Ineke P., Perancangan Sistem Informasi Pendataan Pegawai Pada Dinas Lingkungan Hidup Salatiga Berbasis Web Menggunakan Framework Laravel, Universitas Kristen Satya Wacana, Salatiga, 2021.

Kodrat I.S, Rinta K, Integrasi Data Kepegawaian Aplikasi Sub Sistem di Universitas Diponegoro Melalui *Web Service.* Universitas Diponegoro, Semarang. 2014.

Kurniawan. H, "Implementasi REST Web Service Untuk Sales Order Dan Sales Tracking Berbasis Mobile," Jurnal EKSIS, Vol.7 No.1, Mei 2014.

M. Faizal dan Sanda L.P. Sistem Informasi Pengolahan Data Karyawan Berbasis Web (Studi Kasus di PT Perkebunan Nusantara VIII Tambaksari), STMIK Subang, Subang, 2017.

Miyanto, Thoha. (2015). Analisis sistem reservasi hotel d'griya serang dan Perancangan Jurnal Pengembangan Riset dan Observasi Sistem Komputer vol.2, No.2 2015.

McCool, S. (2012). *Laravel Starter*. Birmingham: Packt Publishing Ltd

Nugroho, B., 2013. "*Dasar Pemograman Web PHP - MySQL dengan Dreamweaver*". Yogyakarta: Gava Media.

Nurfaizah, Anugrah A.P, "Implementasi Sistem Terintegrasi pada Pengolahan Data Karyawan" Jurnal Telematika Vol.10 No.1 Februari 2017.

Pamungkas, Canggih Ajika. (2017). Pengantar dan Implementasi Basis Data. Yogyakarta: Deepublish.

Pardede J, Ungkawa, Uunk, Kurnia R, Adli, 2013, *Implementasi Web Service Composite* (Studi Kasus Aplikasi Pariwisata), ITENAS, Bandung.

Priyatno, Dewi. 2008, Mandiri Belajar SPSS - Bagi Mahasiswa dan Umum. Yogyakarta: MediaKom.

Rahman, M.A., Kuswardayan, I. dan Hariadi, R.R.(2013*) Perancangan dan Implementasi RESTful Web Service untuk Game Sosial Food Merchant Saga pada Perangkat Android.* Teknik Informatika ITS, 1(2),

Ramadhani, M.F. (2015). *Pembangunan Aplikasi Informasi, Pengaduan, Kritik dan Saran Seputar Kota Cimahi pada Platform Android.* Bandung: Universitas Komputer Indonesia.

R. Choirudin and A. Adil. (2019). "*Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa,"* MATRIK : Jurnal Manajemen Teknik Informatika dan Rekayasa Komputer, vol. 18, no. 2, pp. 284–293, May 2019, doi: 10.30812/matrik.v18i2.407

Reddy, Martin (2011). *API Design for C++*. Elsevier

Romney, Marshall B Dan Paul John Steinbart. 2015. *Sistem Informasi Akuntansi*. Jakarta: Salemba Empat.

Rosa A.S dan M.Shalahuddin. 2011. *Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek.,* Bandung.

Simarmata, J. (2010). *Rekaya Perangkat Lunak*. Yogyakarta: Andi.

Siswoutomo Wiwit. (2004). *Membangun Web Service Open Source Menggunakan PHP*, PT Elexmedia Komputindo, Jakarta.

Siti L.O, "Analisa Integrasi Data SINTA (Science and Technology Index) Menggunakan Website Internasional Dengan Manajamen Sistem Informasi EIS (Executive Information System)", Universitas Sriwijaya, Palembang, 2018.

Sofika E., Larissa N.R, Fadli H. Pengembangan Sistem Informasi Administrasi Kepegawaian Pada Fakultas Ilmu Sosial Ilmu Politik Universitas Andalas Padang Berbasis *Web*. UPI YPTK, Padang, 2017.

Sommerville. (2011). *Software Engineering 9th Edition.* Wesley: Addison.

Somyat, R. dan Nathanael, T. (2017). *Pengembangan Sistem Informasi Pelatihan Berbasis Web Menggunakan Teknologi Web Service Dan Framework Laravel Teknologi Web Service.* Universitas Kristen Satya Wacana.

Supono, dan Vidiandry, P. (2016). *Pemrograman Web dengan Menggunakan PHP* dan *Framework Codeigniter.* Yogyakarta: Deepublish.

Suprihadi, Rini K.H, Lina S.W. (2013). *Rancang Bangun Sistem Jejaring Klaster Berbasis Web Menggunakan Metode Model View Controller*.

Sutanto. Y. (2017).Integrasi Sistem Pelaporan Kerusakan Laboratorium dan SMS Gateway Untuk Realtime Notification Laporan Kerusakan Melalui SMS Dengan Pendekatan *Service Oriented Architecture*," Respati, vol. XIII, no. November, pp. 1-12.

Valarezo, R., & Guarda, T. (2018). *Comparative Analysis of the Laravel and Codeigniter Frameworks*. 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), 1–6.

Wardana. (2010) *Menjadi Master PHP dengan Framework Codeigniter*. Jakarta: PT Elex Media Komputindo.

Winarno, Edy; Ali Zaki, SmithDev. 2014. "Pemrograman Web Berbasis HTML5, PHP, dan JavaScript". Jakarta: PT Elex Media Komputindo.

Zulfian F. R., (2022). *Aplikasi Monitoring Kesehatan Dengan Memanfaatkan Smartwatch Berbasis Android*. Thesis. Universitas Komputer Indonesia. Bandung.

LAMPIRAN A
(USECASE SCENARIO)

1. Login

Use Case	Login
Actor	Admin, Pegawai
Entry Condition	Aktor berada di halaman *login*
Flow of Event	1. Aktor menginputkan NIP/NIK dan password 2. Aktor klik tombol *login* 3. Sistem memvalidasi data jika data valid, jika valid pengguna akan diarahkan ke halaman utama
Exit Condition	Sistem menampilkan halaman utama

2. Mengelola data pribadi pegawai

a. Edit data pribadi pegawai

Use Case	Edit data pribadi
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar pegawai
Flow of Event	1. Aktor mengklik *icon* edit 2. Sistem menampilkan *form* edit data pegawai 3. Aktor mengubah data pegawai dan klik tombol simpan 4. Sistem menyimpan data 5. Sistem menampilkan halaman daftar pegawai
Exit Condition	Aktor berhasil mengubah data pegawai

b. Lihat data pribadi pegawai

Use Case	Lihat data pribadi
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar pegawai
Flow of Event	1. Aktor mengklik *icon* lihat 2. Sistem menampilkan halaman data pegawai
Exit Condition	Aktor berhasil melihat data pegawai

c. Hapus data pribadi pegawai

Use Case	Hapus data pribadi
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar pegawai
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data pegawai

3. Mengelola data master jabatan struktural

a. Tambah jabatan struktural

Use Case	Tambah jabatan struktural
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar jabatan struktural
Flow of Event	1. Aktor mengklik *icon* tambah jabatan struktural 2. Sistem menampilkan *form* tambah data jabatan struktural

	3. Aktor mengisi data jabatan struktural dan klik tombol simpan 4. Sistem menyimpan data 5. Sistem menampilkan halaman daftar jabatan struktural
Exit Condition	Aktor berhasil menyimpan daftar jabatan struktural

b. Edit jabatan struktural

Use Case	Edit jabatan struktural
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar jabatan struktural
Flow of Event	1. Aktor mengklik *icon* edit 2. Sistem menampilkan *form* edit data 3. Aktor mengubah data jabatan struktural dan klik tombol simpan 4. Sistem menyimpan data jabatan struktural 5. Sistem menampilkan halaman daftar jabatan struktural
Exit Condition	Aktor berhasil mengubah daftar jabatan struktural

c. Lihat jabatan struktural

Use Case	Lihat jabatan struktural
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar jabatan struktural
Flow of Event	1. Aktor mengklik *icon* lihat 2. Sistem menampilkan halaman data jabatan struktural
Exit Condition	Aktor berhasil melihat data jabatan struktural

d. Hapus jabatan struktural

Use Case	Hapus jabatan struktural
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar jabatan struktural
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data jabatan struktural

4. Mengelola data master jabatan fungsional

a. Tambah jabatan fungsional

Use Case	Tambah jabatan fungsional
Actor	Admin
Entry Condition	Aktor berada pada halaman jabatan fungsional
Flow of Event	1. Aktor mengklik *icon* tambah jabatan fungsional 2. Sistem menampilkan *form* tambah data jabatan fungsional 3. Aktor mengisi data jabatan fungsional dan klik tombol simpan

	4. Sistem menyimpan data 5. Sistem menampilkan halaman daftar jabatan fungsional
Exit Condition	Aktor berhasil mengubah data jabatan fungsional

b. Edit jabatan fungsional

Use Case	Edit jabatan fungsional
Actor	Admin
Entry Condition	Aktor berada pada halaman jabatan fungsional
Flow of Event	1. Aktor mengklik *icon* edit jabatan fungsional 2. Sistem menampilkan *form* edit data 3. Aktor mengubah data jabatan fungsional dan klik tombol simpan 4. Sistem menyimpan data 5. Sistem menampilkan halaman daftar jabatan fungsional
Exit Condition	Aktor berhasil menambahkan data jabatan fungsional

c. Lihat jabatan fungsional

Use Case	Lihat jabatan fungsional
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar jabatan fungsional
Flow of Event	1. Aktor mengklik *icon* lihat 2. Sistem menampilkan halaman data jabatan fungsional
Exit Condition	Aktor berhasil melihat data jabatan fungsional

d. Hapus jabatan fungsional

Use Case	Hapus jabatan fungsional
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar jabatan fungsional
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data jabatan fungsional

5. Mengelola unit kerja

a. Tambah unit kerja

Use Case	Tambah unit kerja
Actor	Admin
Entry Condition	Aktor berada pada halaman unit kerja
Flow of Event	1. Aktor mengklik *icon* tambah unit kerja 2. Sistem menampilkan *form* tambah unit kerja 3. Aktor mengisi data unit kerja dan klik tombol simpan 4. Sistem menyimpan data unit kerja 5. Sistem menampilkan halaman unit kerja
Exit Condition	Aktor berhasil mengubah data unit kerja

b. Edit unit kerja

Use Case	Edit unit kerja
Actor	Admin
Entry Condition	Aktor berada pada halaman unit kerja
Flow of Event	1. Aktor mengklik *icon* edit unit kerja 2. Sistem menampilkan *form* edit data unit kerja 3. Aktor mengubah data unit kerja dan klik tombol simpan 4. Sistem menyimpan data unit kerja 5. Sistem menampilkan halaman daftar unit kerja
Exit Condition	Aktor berhasil menambahkan data unit kerja

c. Lihat unit kerja

Use Case	Lihat unit kerja
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar unit kerja
Flow of Event	3. Aktor mengklik *icon* lihat data 4. Sistem menampilkan halaman data unit kerja
Exit Condition	Aktor berhasil melihat data unit kerja

d. Hapus unit kerja

Use Case	Hapus unit kerja
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar unit kerja
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data unit kerja

6. Mengelola pendidikan

a. Tambah pendidikan

Use Case	Tambah pendidikan
Actor	Admin
Entry Condition	Aktor berada pada halaman pendidikan
Flow of Event	1. Aktor mengklik *icon* tambah pendidikan 2. Sistem menampilkan *form* tambah pendidikan 3. Aktor mengisi data pendidikan dan klik tombol simpan 4. Sistem menyimpan data pendidikan 5. Sistem menampilkan halaman pendidikan
Exit Condition	Aktor berhasil mengubah data pendidikan

b. Edit pendidikan

Use Case	Edit pendidikan
Actor	Admin
Entry Condition	Aktor berada pada halaman pendidikan
Flow of Event	1. Aktor mengklik *icon* edit pendidikan 2. Sistem menampilkan *form* edit data pendidikan 3. Aktor mengubah data pendidikan dan klik tombol simpan

	4. Sistem menyimpan data pendidikan 5. Sistem menampilkan halaman daftar pendidikan
Exit Condition	Aktor berhasil menambahkan data pendidikan

c. Lihat pendidikan

Use Case	Lihat pendidikan
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar pendidikan
Flow of Event	1. Aktor mengklik *icon* lihat data 2. Sistem menampilkan halaman data pendidikan
Exit Condition	Aktor berhasil melihat data pendidikan

d. Hapus pendidikan

Use Case	Hapus pendidikan
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar pendidikan
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data pendidikan

7. Mengelola *access token* API

a. Tambah *access token* API

Use Case	Tambah *access token* API
Actor	Admin
Entry Condition	Aktor berada pada halaman *access token* API
Flow of Event	1. Aktor mengklik *icon* tambah *access token* API 2. Sistem menampilkan *form* tambah *access token* API 3. Aktor mengisi data *access token* API dan klik tombol simpan 4. Sistem menyimpan data *access token* API 5. Sistem menampilkan halaman *access token* API
Exit Condition	Aktor berhasil mengubah data *access token* API

b. Edit *access token* API

Use Case	Edit *access token* API
Actor	Admin
Entry Condition	Aktor berada pada halaman *access token* API
Flow of Event	1. Aktor mengklik *icon* edit *access token* API 2. Sistem menampilkan *form* edit data 3. Aktor mengubah data *access token* API dan klik tombol simpan 4. Sistem menyimpan data *access token* API 5. Sistem menampilkan halaman daftar *access token* API
Exit Condition	Aktor berhasil menambahkan data *access token* API

c. Lihat *access token* API

Use Case	Lihat *access token API*
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar *access token API*
Flow of Event	1. Aktor mengklik *icon* lihat data 2. Sistem menampilkan halaman data *access token API*
Exit Condition	Aktor berhasil melihat data *access token API*

d. Hapus *access token* API

Use Case	Hapus *access token* API
Actor	Admin
Entry Condition	Aktor berada pada halaman daftar *access token* API
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data *access token* API

8. Mengelola riwayat pendidikan

a. Tambah riwayat pendidikan

Use Case	Tambah riwayat pendidikan
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat pendidikan 2. Sistem menampilkan *form* tambah riwayat pendidikan 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat pendidikan 5. Sistem menampilkan halaman pendidikan
Exit Condition	Aktor berhasil mengubah data riwayat pendidikan

b. Edit riwayat pendidikan

Use Case	Edit riwayat pendidikan
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat pendidikan 2. Sistem menampilkan *form* tambah riwayat pendidikan 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat pendidikan 5. Sistem menampilkan halaman riwayat pendidikan
Exit Condition	Aktor berhasil mengubah data riwayat pendidikan

c. Lihat riwayat pendidikan

Use Case	Lihat riwayat pendidikan
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* lihat data riwayat pendidikan

	2. Sistem menampilkan halaman data riwayat pendidikan
Exit Condition	Aktor berhasil melihat data riwayat pendidikan

d. Hapus riwayat pendidikan

Use Case	Hapus riwayat pendidikan
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data riwayat pendidikan

9. Mengelola riwayat jabatan struktural

a. Tambah riwayat jabatan struktural

Use Case	Tambah riwayat jabatan struktural
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat jabatan struktural 2. Sistem menampilkan *form* tambah riwayat jabatan struktural 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat jabatan struktural 5. Sistem menampilkan halaman jabatan struktural
Exit Condition	Aktor berhasil mengubah data riwayat jabatan structural

b. Edit riwayat jabatan struktural

Use Case	Edit riwayat jabatan struktural
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat jabatan struktural 2. Sistem menampilkan *form* tambah riwayat jabatan struktural 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat jabatan struktural 5. Sistem menampilkan halaman riwayat jabatan struktural
Exit Condition	Aktor berhasil mengubah data riwayat jabatan structural

c. Lihat riwayat jabatan struktural

Use Case	Lihat riwayat jabatan struktural
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	3. Aktor mengklik *icon* lihat data riwayat jabatan struktural

	4. Sistem menampilkan halaman data riwayat jabatan structural
Exit Condition	Aktor berhasil melihat data riwayat jabatan structural

d. Hapus riwayat jabatan struktural

Use Case	Hapus riwayat jabatan struktural
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data riwayat jabatan struktural

10. Mengelola riwayat jabatan fungsional

a. Tambah riwayat jabatan fungsional

Use Case	Tambah riwayat jabatan fungsional
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat jabatan fungsional 2. Sistem menampilkan *form* tambah riwayat jabatan fungsional 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat jabatan fungsional 5. Sistem menampilkan halaman jabatan fungsional
Exit Condition	Aktor berhasil mengubah data riwayat jabatan fungsional

b. Edit riwayat jabatan fungsional

Use Case	Edit riwayat jabatan fungsional
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat jabatan fungsional 2. Sistem menampilkan *form* tambah riwayat jabatan fungsional 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat jabatan fungsional 5. Sistem menampilkan halaman riwayat jabatan fungsional
Exit Condition	Aktor berhasil mengubah data riwayat jabatan fungsional

c. Lihat riwayat jabatan fungsional

Use Case	Lihat riwayat jabatan fungsional
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* lihat data riwayat jabatan fungsional 2. Sistem menampilkan halaman data riwayat jabatan fungsional
Exit Condition	Aktor berhasil melihat data riwayat jabatan fungsional

d. Hapus riwayat jabatan fungsional

Use Case	Hapus riwayat jabatan struktural
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data riwayat jabatan struktural

11. Mengelola riwayat pangkat golongan

a. Tambah riwayat pangkat golongan

Use Case	Tambah riwayat pangkat golongan
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat pangkat golongan 2. Sistem menampilkan *form* tambah riwayat pangkat golongan 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat pangkat golongan 5. Sistem menampilkan halaman pangkat golongan
Exit Condition	Aktor berhasil mengubah data riwayat pangkat golongan

b. Edit riwayat pangkat golongan

Use Case	Edit riwayat pangkat golongan
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat pangkat golongan 2. Sistem menampilkan *form* tambah riwayat pangkat golongan 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat pangkat golongan 5. Sistem menampilkan halaman riwayat pangkat golongan
Exit Condition	Aktor berhasil mengubah data riwayat pangkat golongan

c. Lihat riwayat pangkat golongan

Use Case	Lihat riwayat pangkat golongan
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* lihat data riwayat pangkat golongan 2. Sistem menampilkan halaman data riwayat pangkat golongan
Exit Condition	Aktor berhasil melihat data riwayat pangkat golongan

d. Hapus riwayat pangkat golongan

Use Case	Hapus riwayat pangkat golongan
Actor	*User*

Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data riwayat pangkat golongan

12. Mengelola riwayat diklat

a. Tambah riwayat diklat

Use Case	Tambah riwayat diklat
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat diklat 2. Sistem menampilkan *form* tambah riwayat diklat 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat diklat 5. Sistem menampilkan halaman diklat
Exit Condition	Aktor berhasil mengubah data riwayat diklat

b. Edit riwayat diklat

Use Case	Edit riwayat diklat
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat diklat 2. Sistem menampilkan *form* tambah riwayat diklat 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat diklat 5. Sistem menampilkan halaman riwayat diklat
Exit Condition	Aktor berhasil mengubah data riwayat diklat

c. Lihat riwayat diklat

Use Case	Lihat riwayat diklat
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* lihat data riwayat diklat 2. Sistem menampilkan halaman data riwayat diklat
Exit Condition	Aktor berhasil melihat data riwayat diklat

d. Hapus riwayat diklat

Use Case	Hapus riwayat diklat
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data riwayat diklat

13. Mengelola riwayat mutasi

a. Tambah riwayat mutasi

Use Case	Tambah riwayat mutasi
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat mutasi 2. Sistem menampilkan *form* tambah riwayat mutasi 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat mutasi 5. Sistem menampilkan halaman mutasi
Exit Condition	Aktor berhasil mengubah data riwayat mutasi

b. Edit riwayat mutasi

Use Case	Edit riwayat mutasi
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah riwayat mutasi 2. Sistem menampilkan *form* tambah riwayat mutasi 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data riwayat mutasi 5. Sistem menampilkan halaman riwayat mutasi
Exit Condition	Aktor berhasil mengubah data riwayat mutase

c. Lihat riwayat mutasi

Use Case	Lihat riwayat mutasi
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* lihat data riwayat mutasi 2. Sistem menampilkan halaman data riwayat mutasi
Exit Condition	Aktor berhasil melihat data riwayat mutasi

d. Hapus riwayat mutasi

Use Case	Hapus riwayat mutasi
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data riwayat mutase

14. Mengelola data keluarga

a. Tambah data keluarga

Use Case	Tambah data keluarga
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah data keluarga

	2. Sistem menampilkan *form* tambah data keluarga 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data keluarga 5. Sistem menampilkan halaman data keluarga
Exit Condition	Aktor berhasil mengubah data keluarga

b. Edit data keluarga

Use Case	Edit data keluarga
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* tambah data keluarga 2. Sistem menampilkan *form* tambah data keluarga 3. Aktor mengisi data dan klik tombol simpan 4. Sistem menyimpan data data keluarga 5. Sistem menampilkan halaman data keluarga
Exit Condition	Aktor berhasil mengubah data keluarga

c. Lihat data keluarga

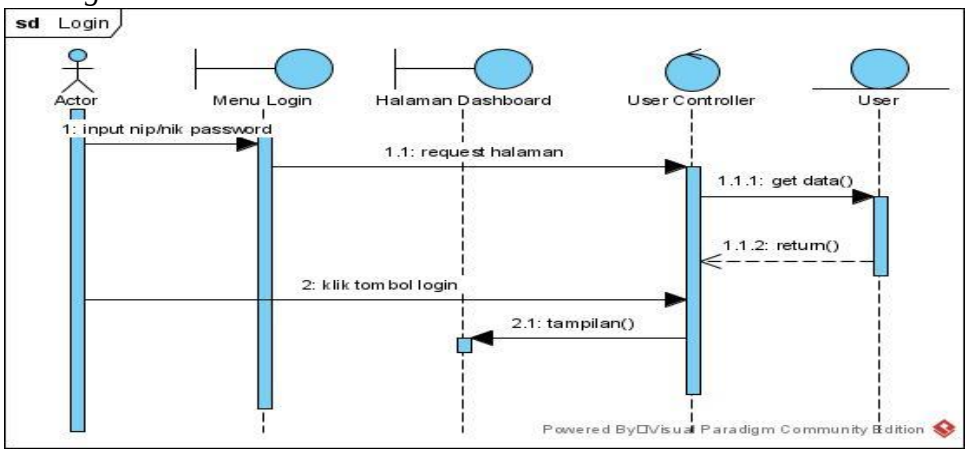
Use Case	Lihat data keluarga
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	3. Aktor mengklik *icon* lihat data keluarga 4. Sistem menampilkan halaman data keluarga
Exit Condition	Aktor berhasil melihat data keluarga

d. Hapus data keluarga

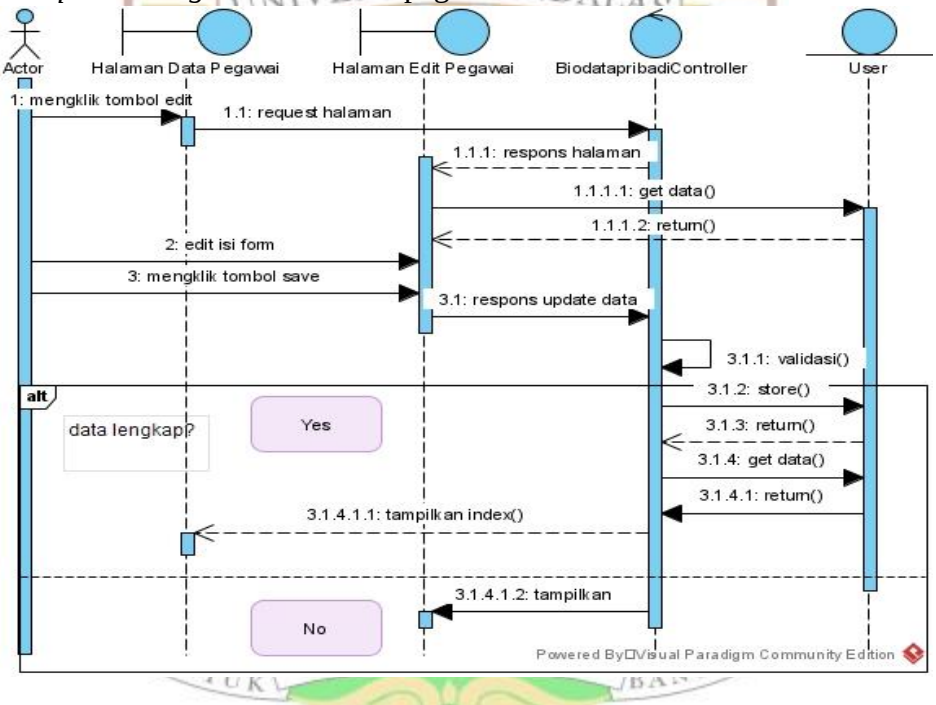
Use Case	Hapus data keluarga
Actor	*User*
Entry Condition	Aktor berada pada halaman data pegawai
Flow of Event	1. Aktor mengklik *icon* hapus 2. Sistem menampilkan validasi hapus data 3. Aktor mengklik OK 4. Sistem menampilkan notifikasi data berhasil dihapus
Exit Condition	Aktor berhasil menghapus data keluarga

LAMPIRAN B
(SEQUENCE DIAGRAM)

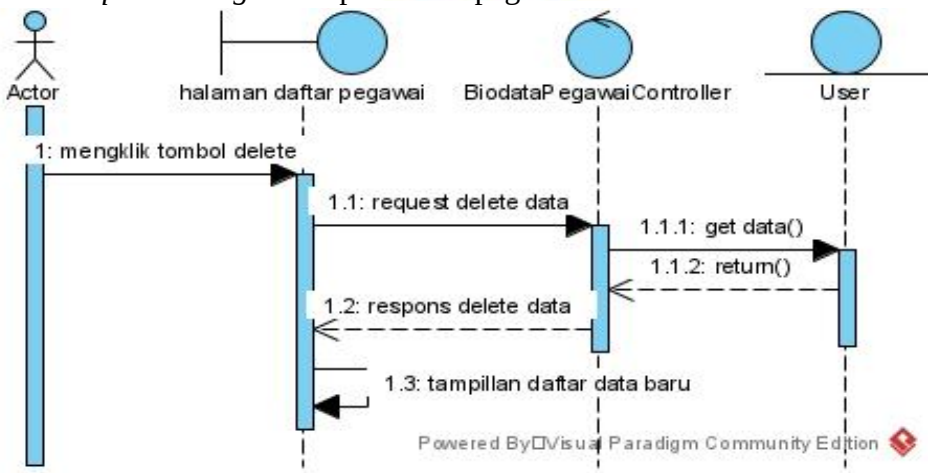
1. *Login*



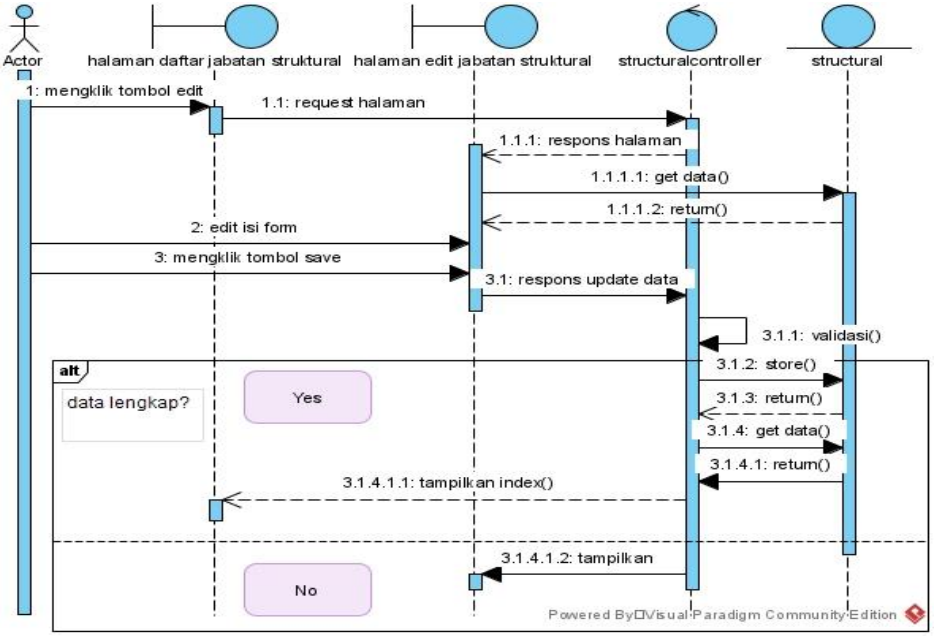
2. *Sequence diagram* edit daftar pegawai



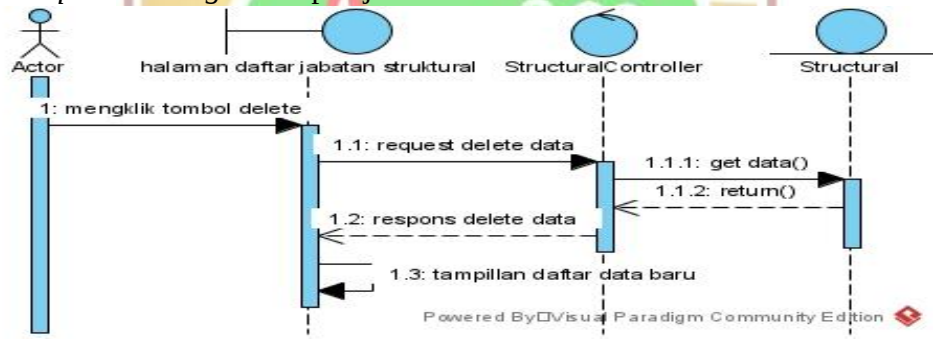
3. *Sequence diagram* hapus daftar pegawai



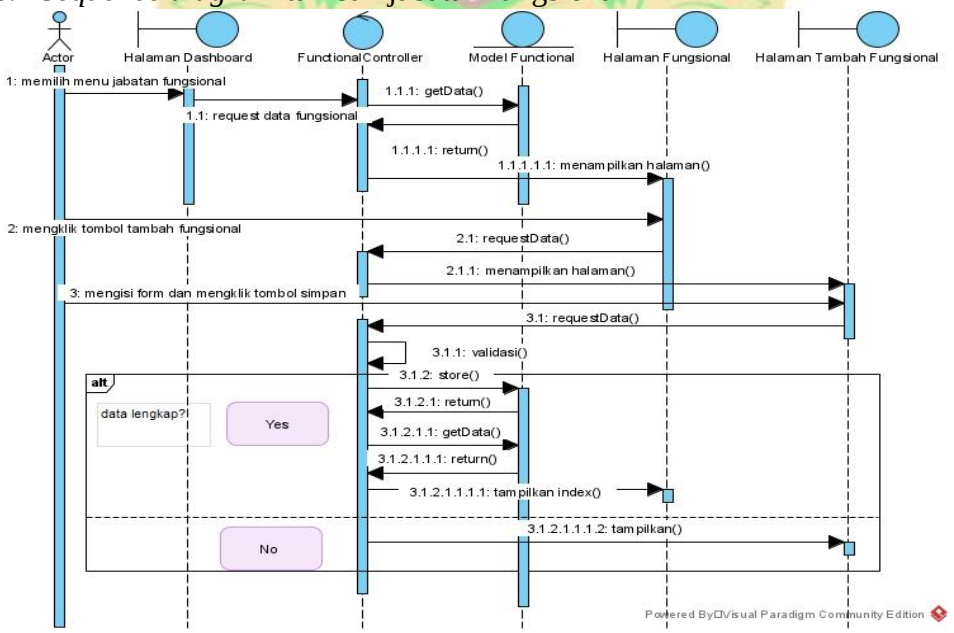
4. *Sequence diagram* edit jabatan struktural



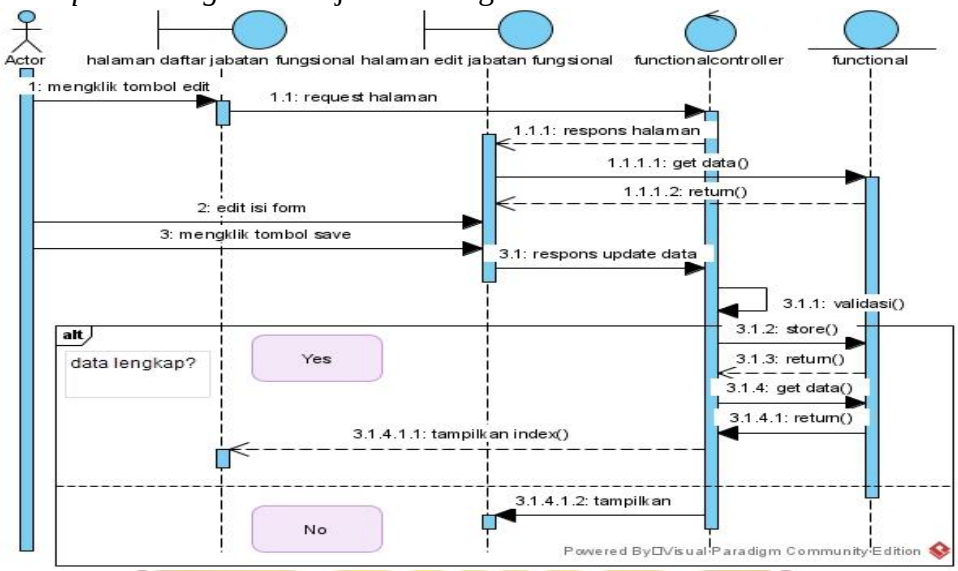
5. *Sequence diagram* hapus jabatan struktural



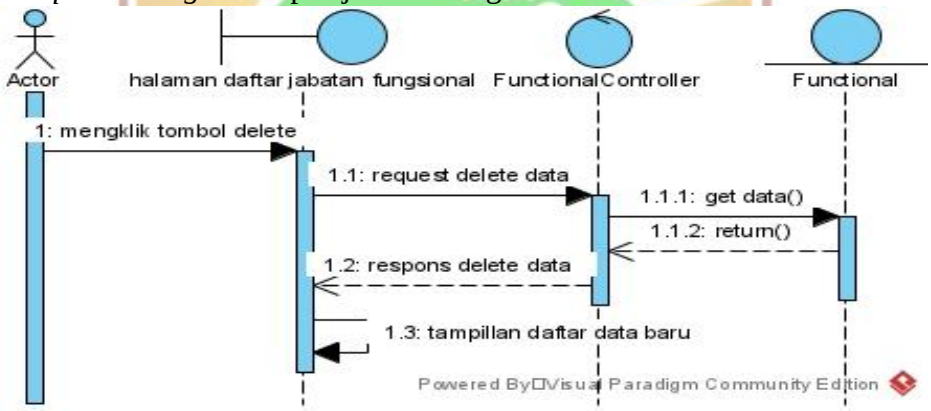
6. *Sequence diagram* tambah jabatan fungsional



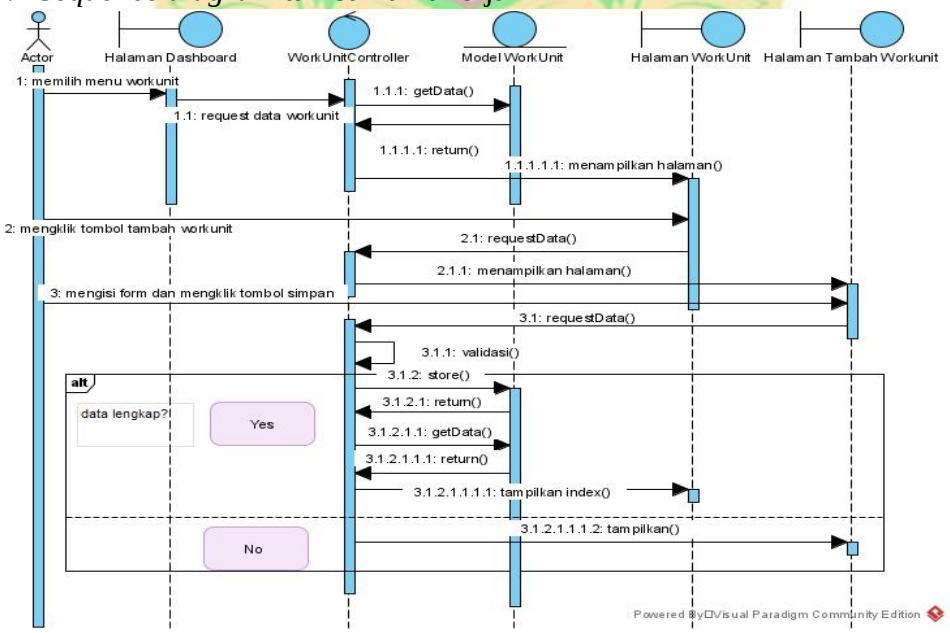
7. *Sequence diagram* edit jabatan fungsional



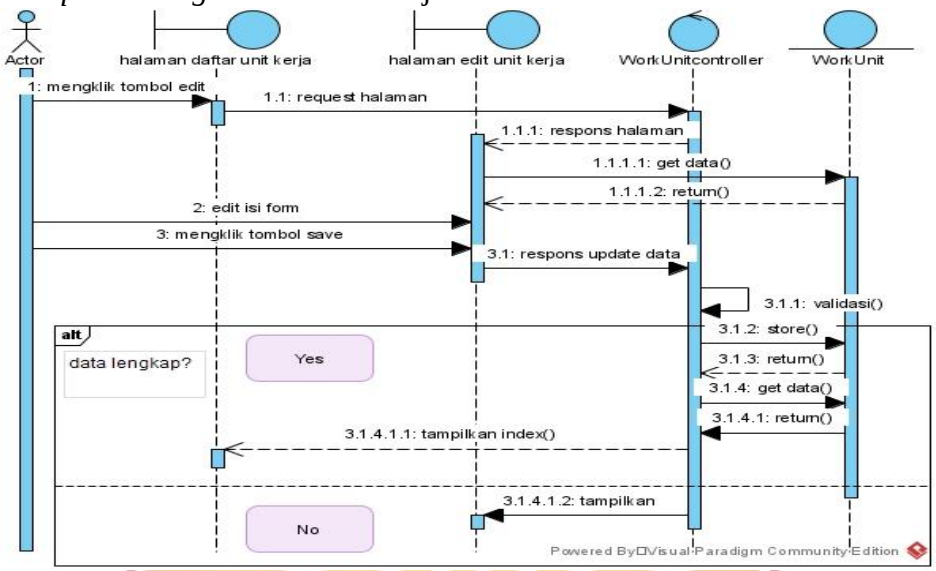
8. *Sequence diagram* hapus jabatan fungsional



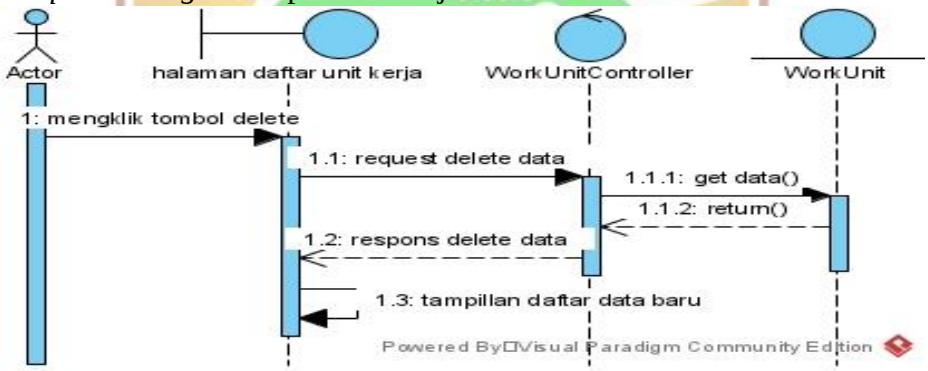
9. *Sequence diagram* tambah unit kerja



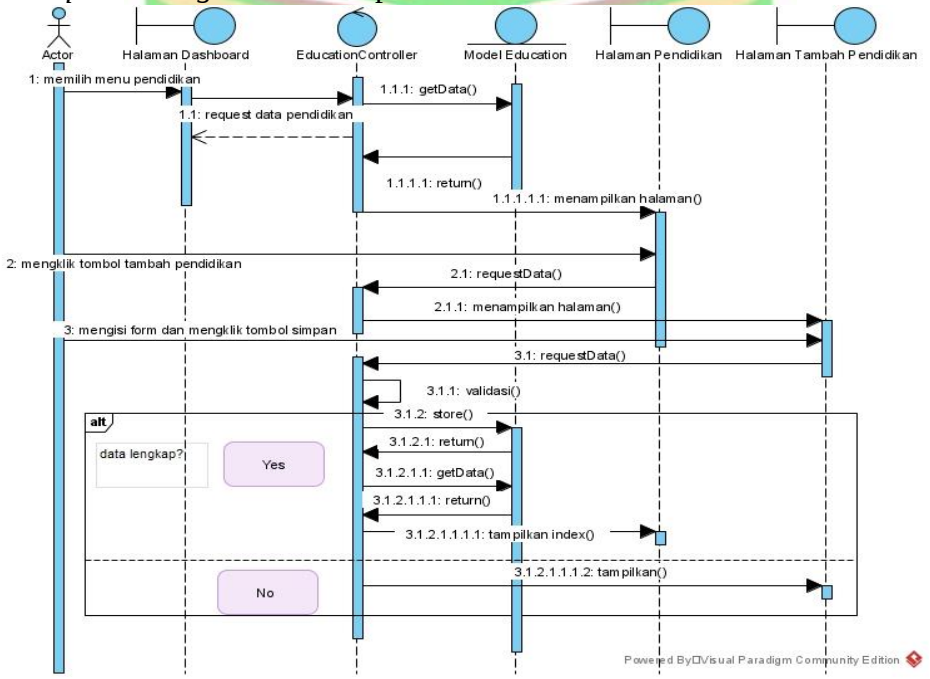
10. *Sequence diagram* edit unit kerja



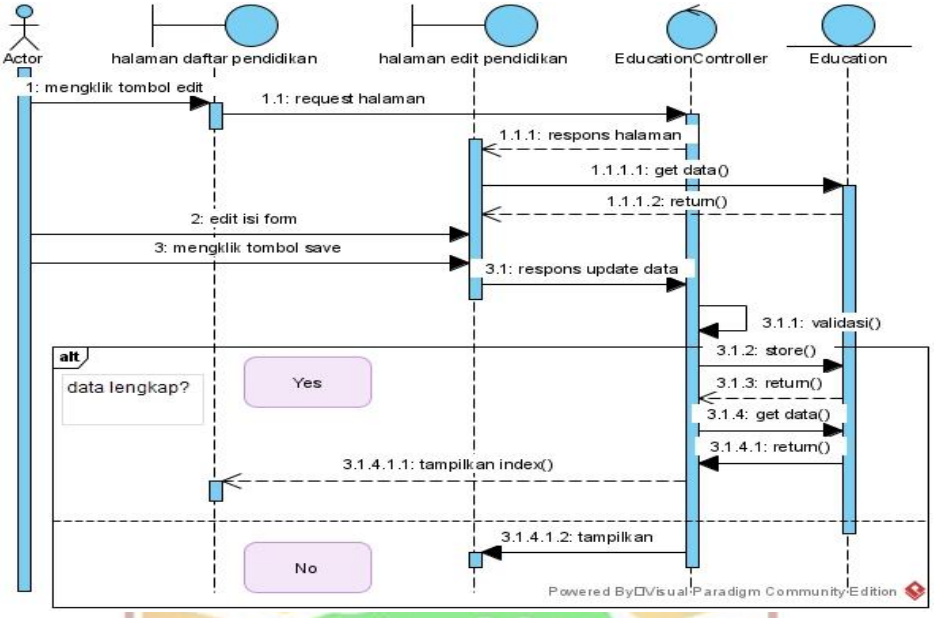
11. *Sequence diagram* hapus unit kerja



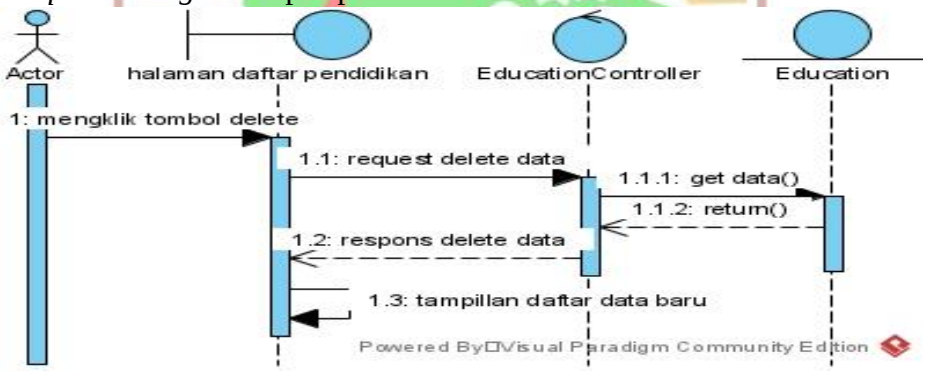
12. *Sequence diagram* tambah pendidikan



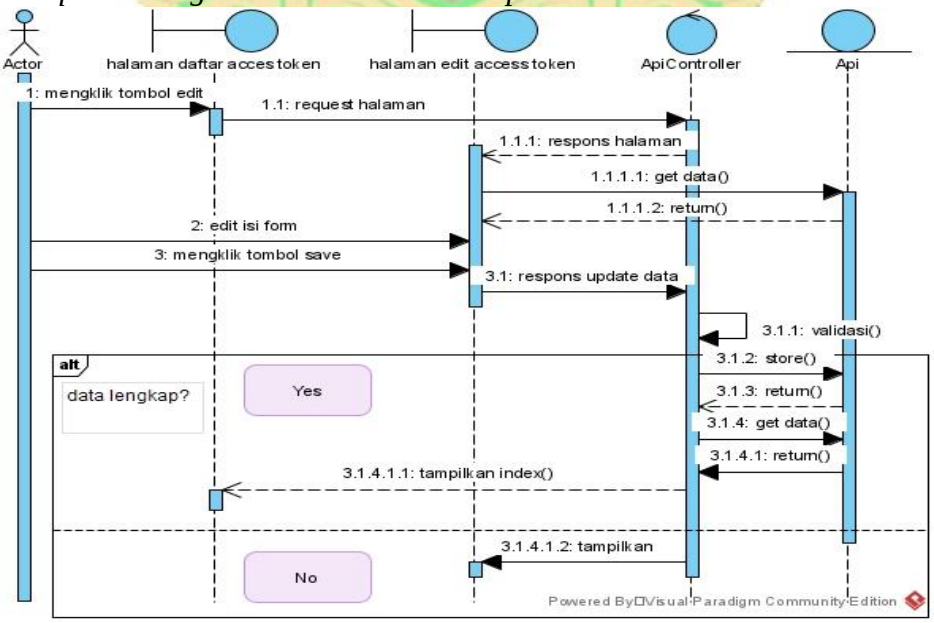
13. *Sequence diagram* edit pendidikan



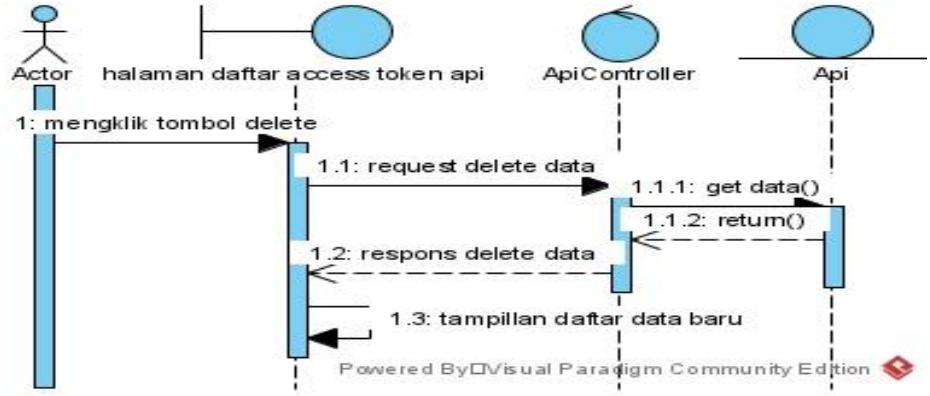
14. *Sequence diagram* hapus pendidikan



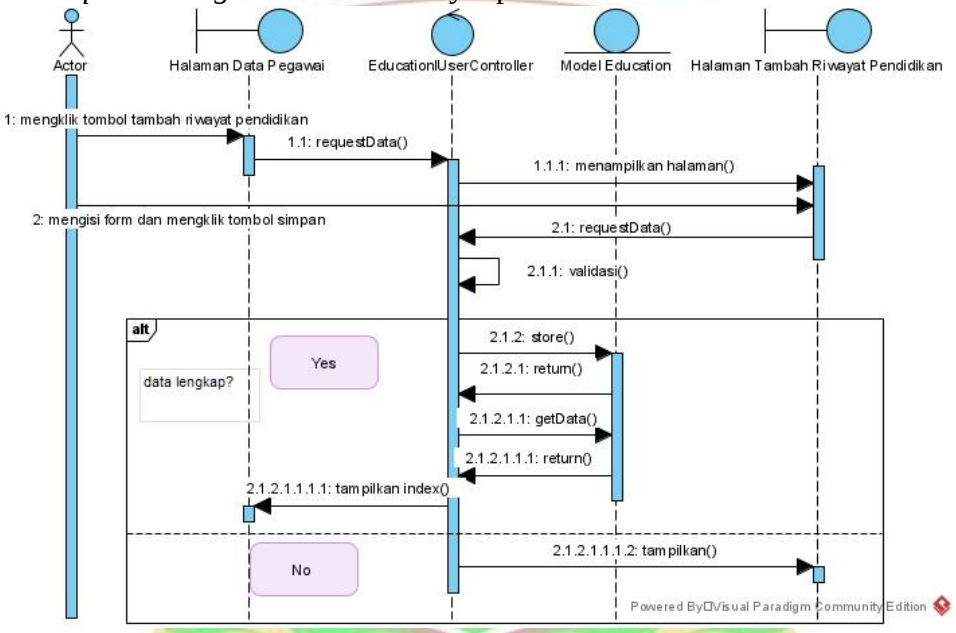
15. *Sequence diagram* edit *access token api*



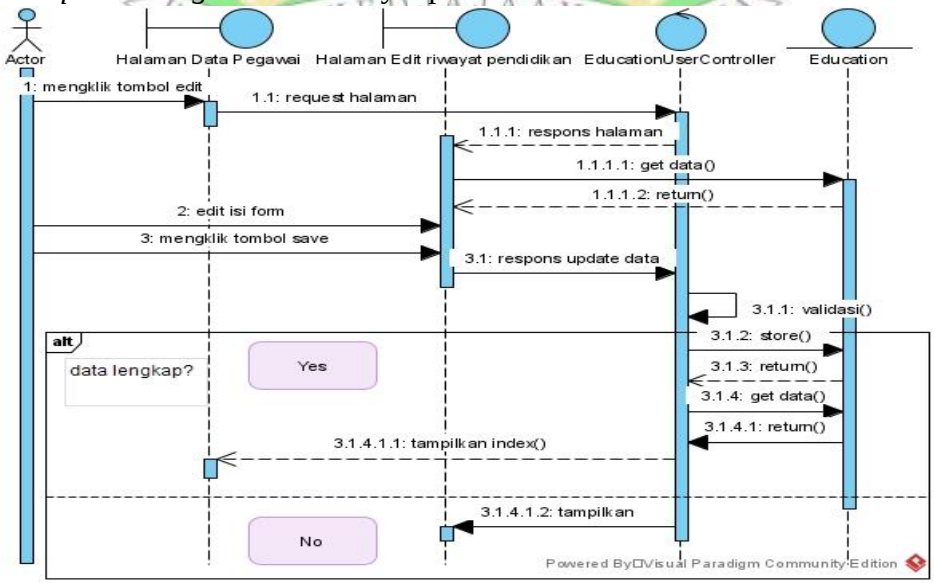
16. *Sequence diagram* hapus *access token api*



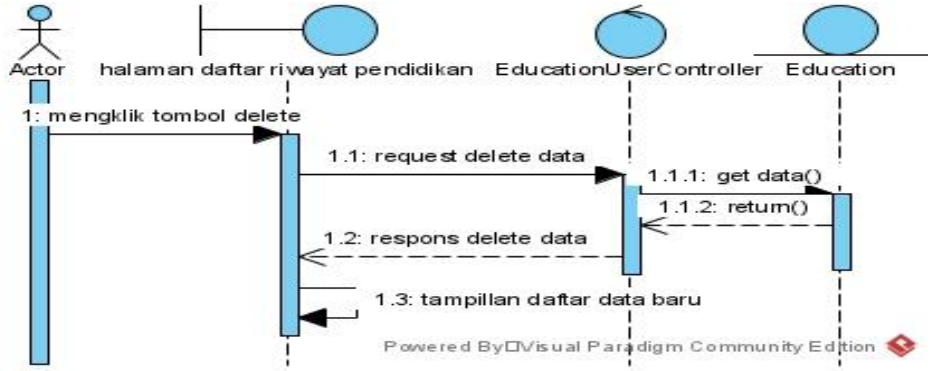
17. *Sequence diagram* tambah riwayat pendidikan



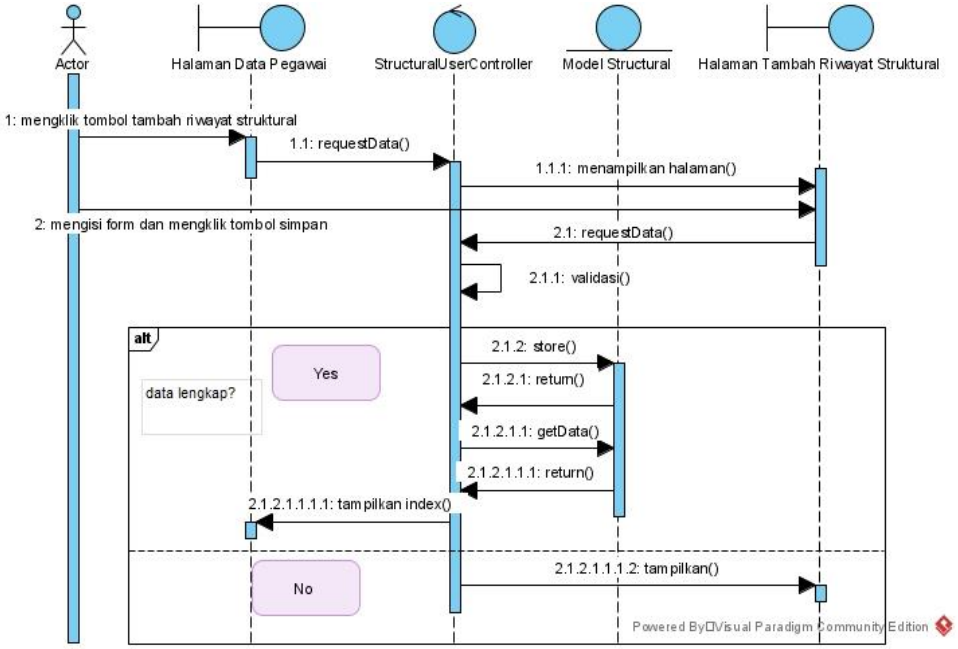
18. *Sequence diagram* edit riwayat pendidikan



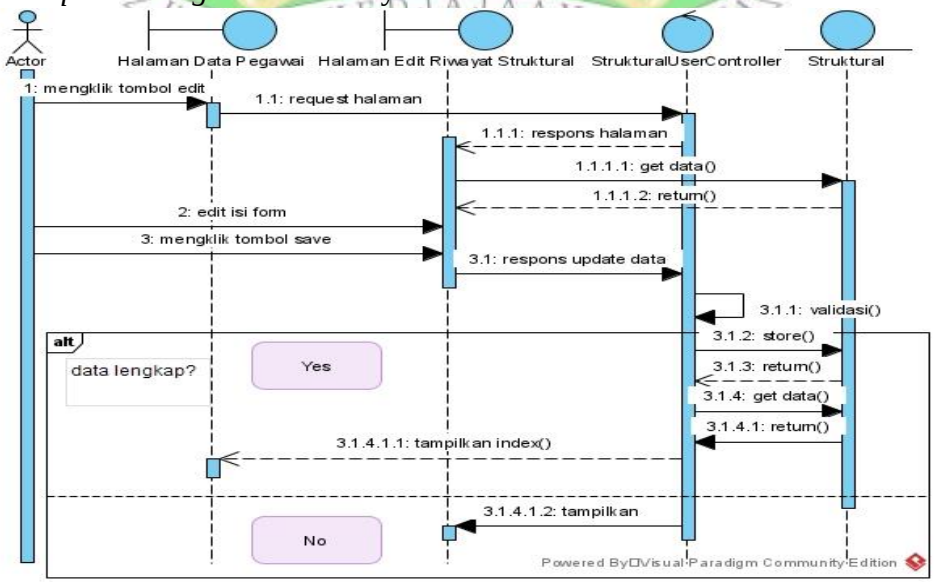
19. *Sequence diagram* hapus riwayat pendidikan



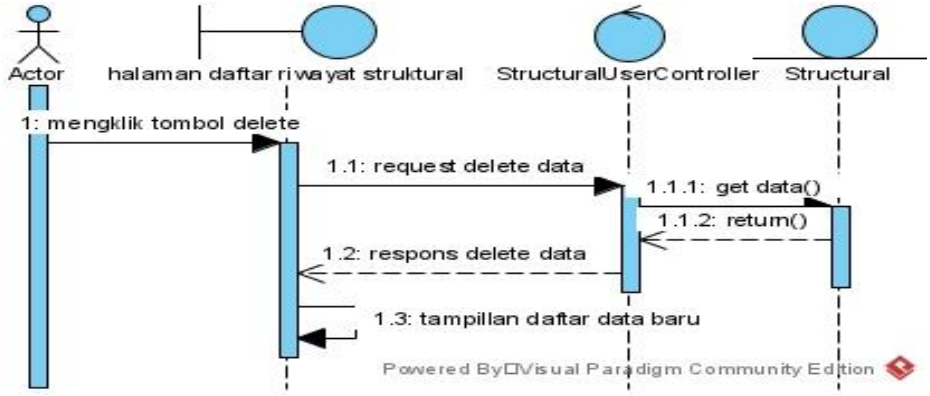
20. *Sequence diagram* tambah riwayat struktural



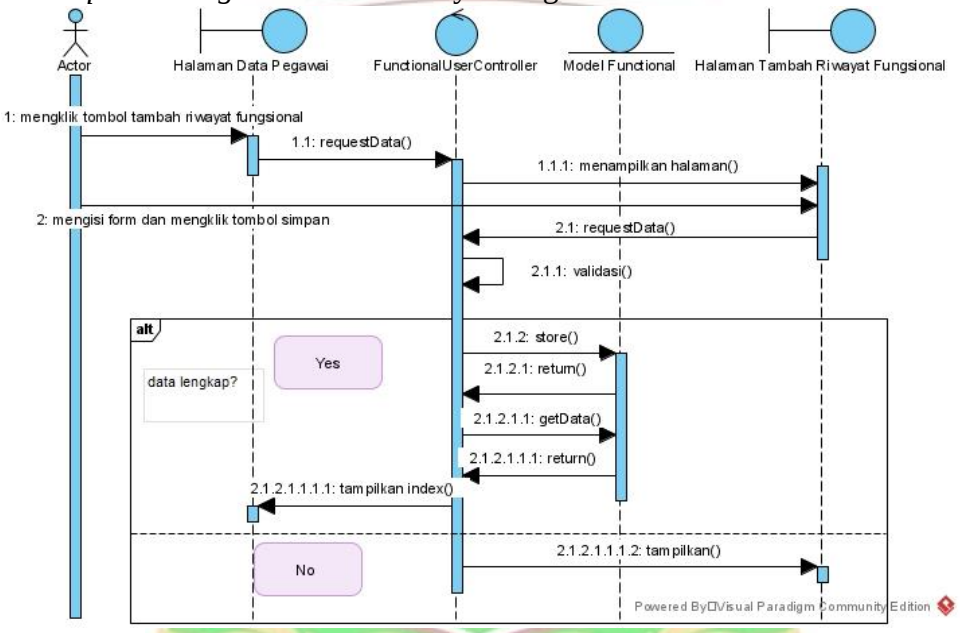
21. *Sequence diagram* edit riwayat struktural



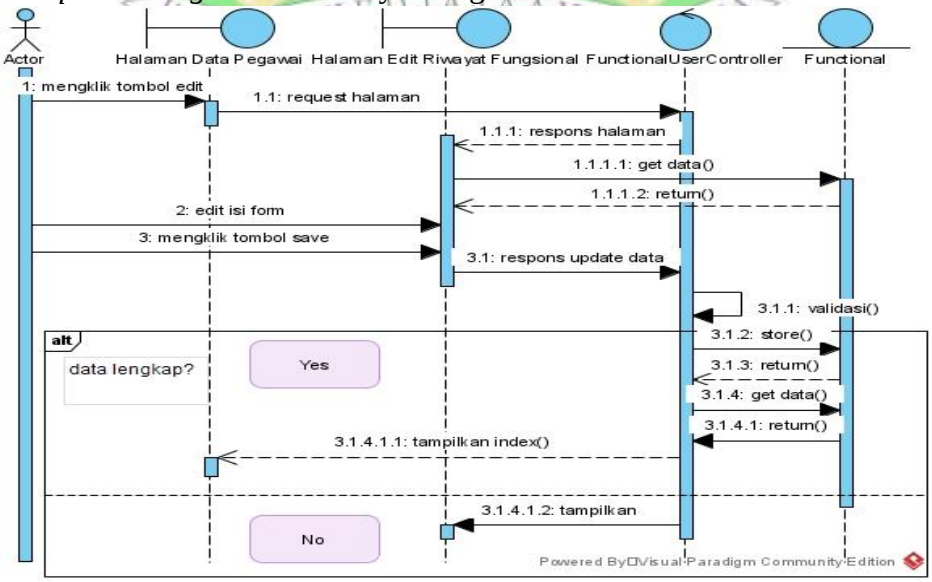
22. *Sequence diagram* hapus riwayat struktural



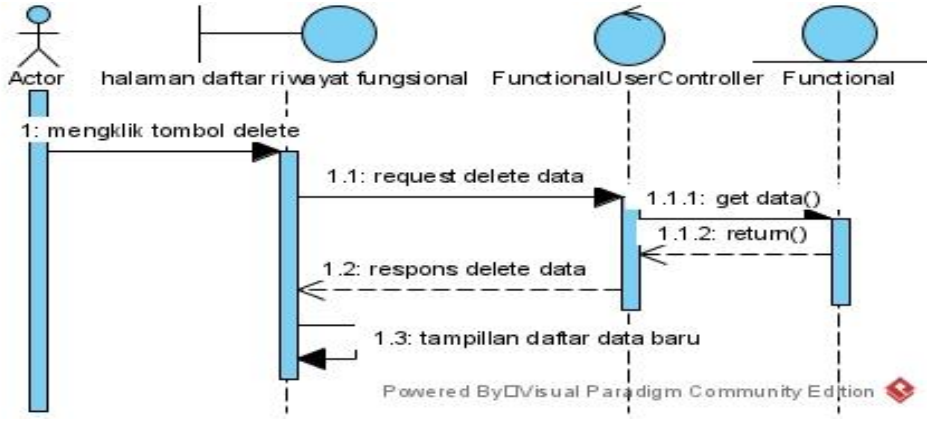
23. *Sequence diagram* tambah riwayat fungsional



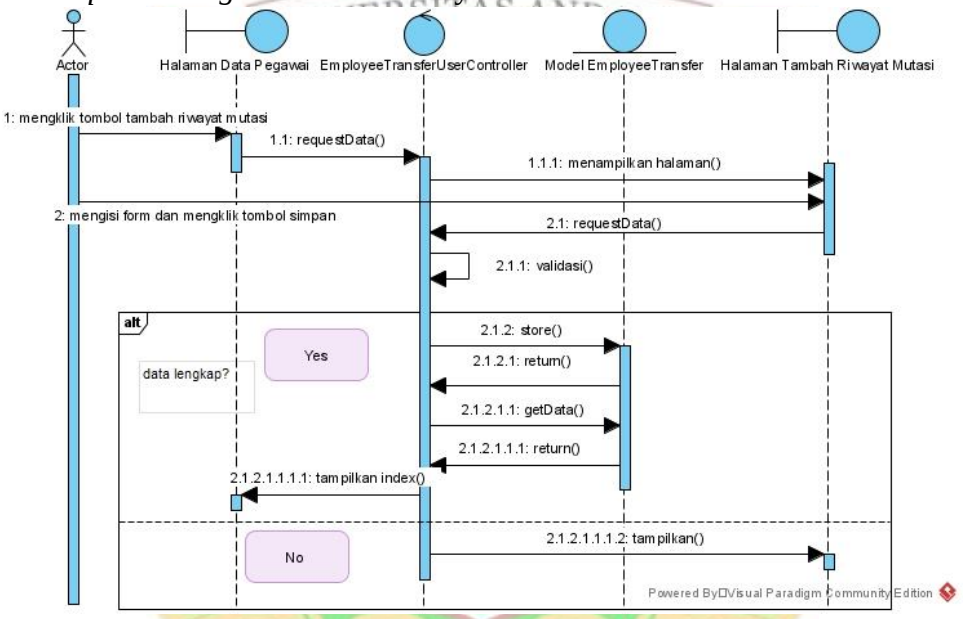
24. *Sequence diagram* edit riwayat fungsional



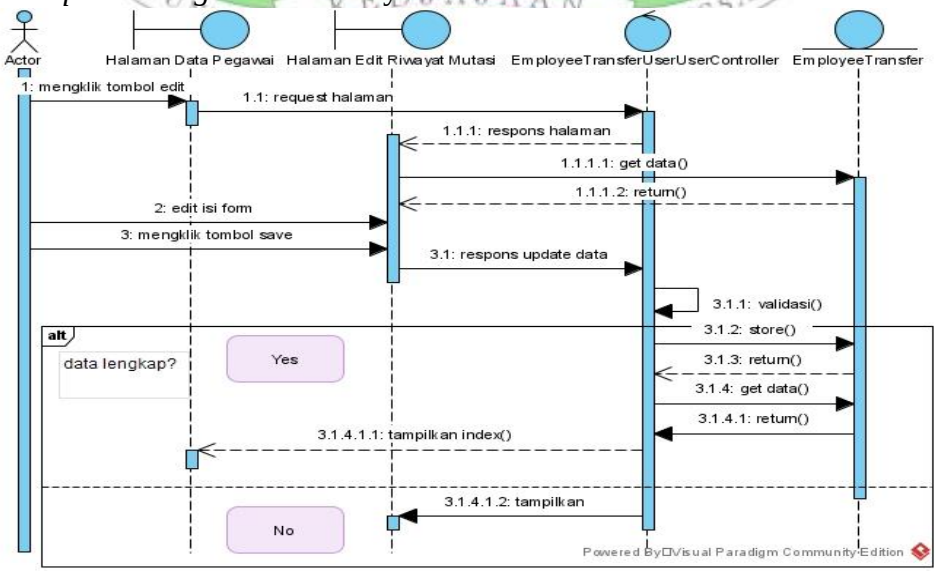
25. *Sequence diagram* hapus riwayat fungsional



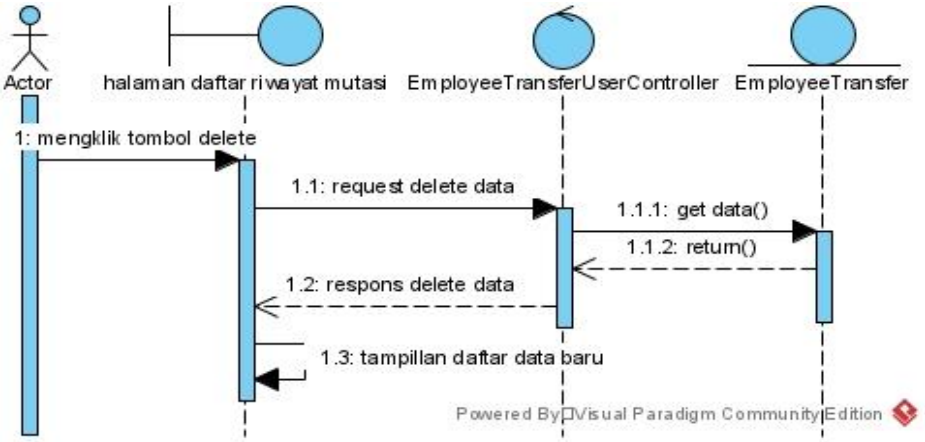
26. *Sequence diagram* tambah riwayat mutasi



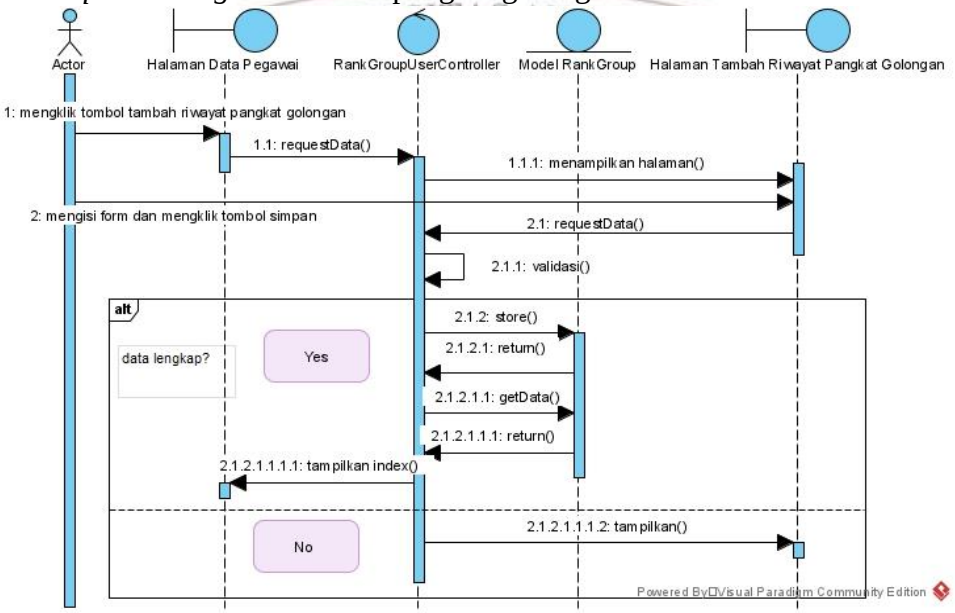
27. *Sequence diagram* edit riwayat mutasi



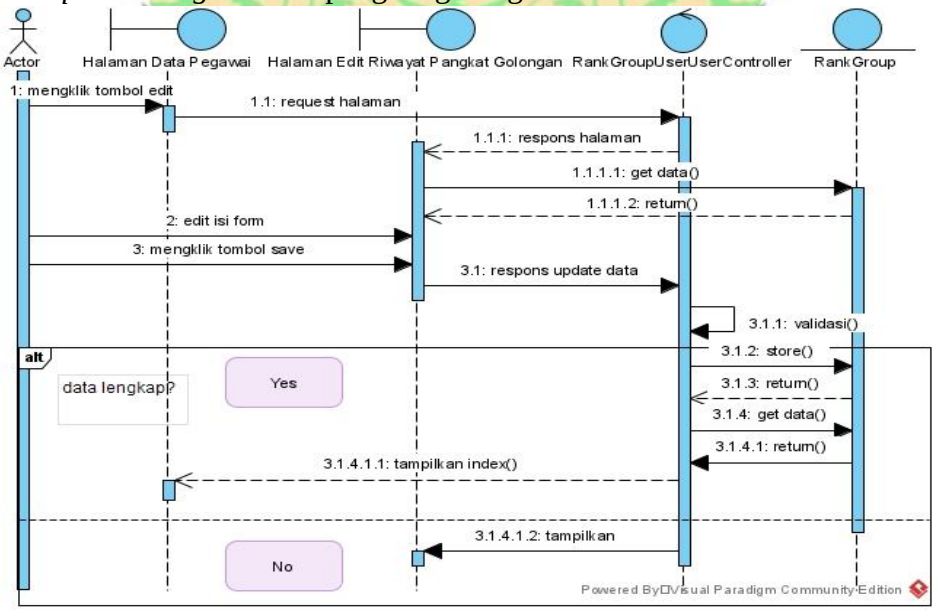
28. *Sequence diagram* hapus riwayat mutasi



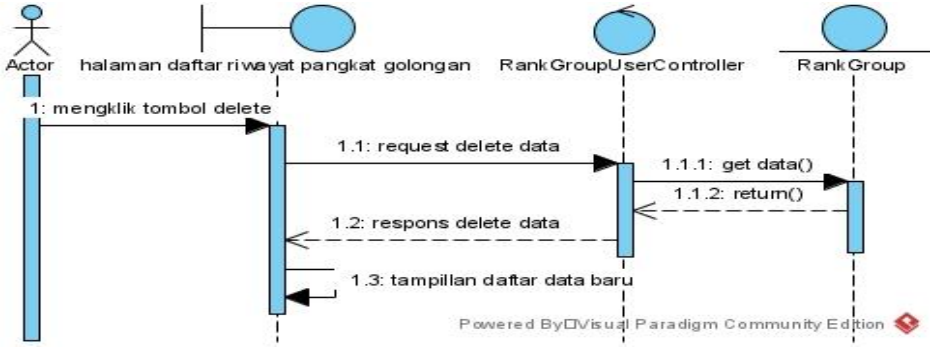
29. *Sequence diagram* tambah pangkat golongan



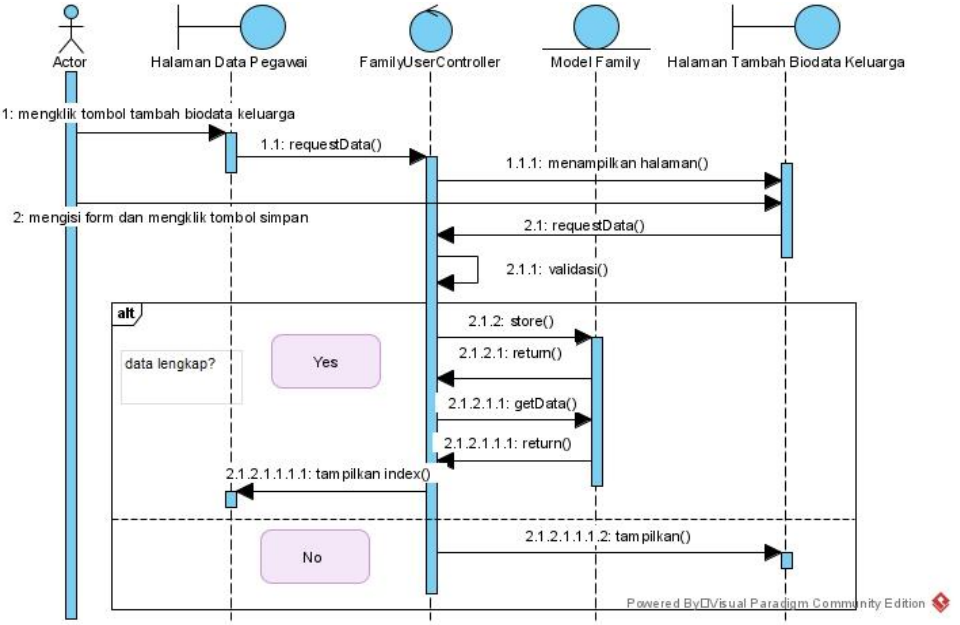
30. *Sequence diagram* edit pangkat golongan



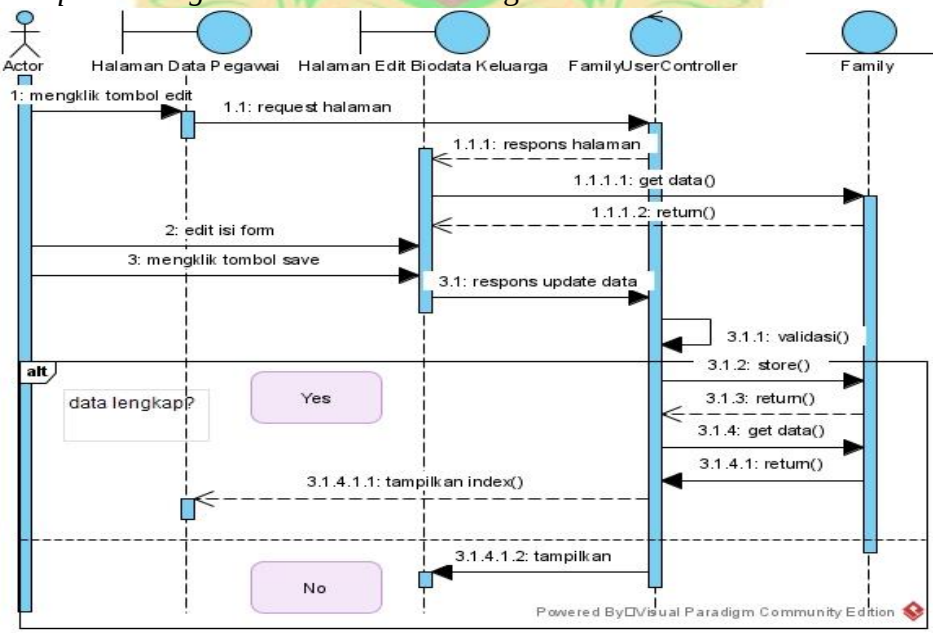
31. *Sequence diagram* hapus pangkat golongan



32. *Sequence diagram* tambah biodata keluarga



33. *Sequence diagram* edit biodata keluarga



LAMPIRAN C
(STRUKTUR DATABASE)

1. Roles

Nama atribut	Tipe Data	Keterangan
id	bigint(20)	PK
nama_role	varchar(30)	
created_at	timestamp	
updated_at	timestamp	

2. Users

Nama atribut	Tipe Data	Keterangan
nip_nik	Varchar(30)	PK
role_id	bigint	FK
nidn	Varchar(30)	
username	Varchar(50)	
password	Varchar(20)	
title_ahead	Varchar(50)	
real_name	Varchar(255)	
back_title	Varchar(50)	
birth_place	Varchar(50)	
birth_date	Date	
photo	Text	
blood_group	Varchar(50)	
height	Varchar(50)	
weight	Varchar(50)	
handicap	Varchar(50)	
phone_number	Varchar(15)	
email	Varchar(50)	
npwp	Varchar(50)	
bpjs	Varchar(50)	
gender	Enum('laki-laki','perempuan')	
religion	Varchar(50)	
marital_status	Varchar(50)	
citizen_status	Varchar(50)	
retirement_age_limit	Varchar(50)	
employee_status	Varchar(50)	
remember_token	Varchar(50)	
created_at	timestamp	
updated_at	timestamp	

3. Work_Units

Nama atribut	Tipe Data	Keterangan
work_unit_id	Varchar(20)	PK
name	Varchar(50)	
created_at	Timestamp	
updated_at	Timestamp	

4. Functionals

Nama atribut	Tipe Data	Keterangan
functional_id	Varchar(20)	PK
information	Varchar(50)	
created_at	timestamp	
updated_at	timestamp	

5. Functional_Details

Nama atribut	Tipe Data	Keterangan
nip_nik	Varchar(20)	PFK
functional_id	Varchar(50)	PFK
tmt	Date	
sign_by	Varchar(255)	
sk_no	Text	
sk_date	Date	
status	Varchar(50)	
sk_file	Varchar(50)	
created_at	timestamp	
updated_at	timestamp	

6. Structural_Details

Nama atribut	Tipe Data	Keterangan
nip_nik	Varchar(20)	PFK
structural_id	Varchar(50)	PFK
tmt	Date	PK
sign_by	Varchar(255)	
sk_no	Text	
sk_date	Date	
status	Varchar(50)	
sk_file	Varchar(50)	
created_at	timestamp	
updated_at	timestamp	

7. Family

Nama atribut	Tipe Data	Keterangan
id_number	Varchar(20)	PK
nip_nik	Varchar(20)	FK
Name	Varchar(255)	
relationship	Varchar(50)	
phone_number	Varchar(20)	
birth_date	Date	
birth_place	Varchar(100)	
status	Varchar(50)	
occupation	Varchar(100)	
last_education	Varchar(100)	
npwp_no	Varchar(50)	
created_at	timestamp	

8. Education_Details

Nama atribut	Tipe Data	Keterangan
education_details_id	varchar(30)	PK
education_id	varchar(30)	FK
nip_nik	varchar(30)	FK
name	varchar(255)	
major	varchar(100)	
graduation_year	year(4)	
country	varchar(100)	
dean/headmaster	varchar(255)	
certificate_file	varchar(100)	
created_at	timestamp	
updated_at	timestamp	

9. Educations

Nama atribut	Tipe Data	Keterangan
education_id	varchar(30)	PK
level	varchar(30)	
created_at	timestamp	
updated_at	timestamp	

10. Address_Details

Nama atribut	Tipe Data	Keterangan
address_details_id	varchar(30)	PK
nip_nik	varchar(30)	FK
district_id	varchar(30)	FK
address	varchar(255)	
created_at	timestamp	
updated_at	timestamp	

11. Regencies

Nama atribut	Tipe Data	Keterangan
regency_id	varchar(30)	PK
regency_name	varchar(255)	
created_at	timestamp	
updated_at	timestamp	

12. Districts

Nama atribut	Tipe Data	Keterangan
district_id	varchar(30)	PK
regency_id	varchar(30)	FK
district_name	varchar(255)	
created_at	timestamp	
updated_at	timestamp	

13. API

Nama atribut	Tipe Data	Keterangan
id	varchar(30)	PK

name	varchar(30)	
token	varchar(255)	
created_at	timestamp	
updated_at	timestamp	

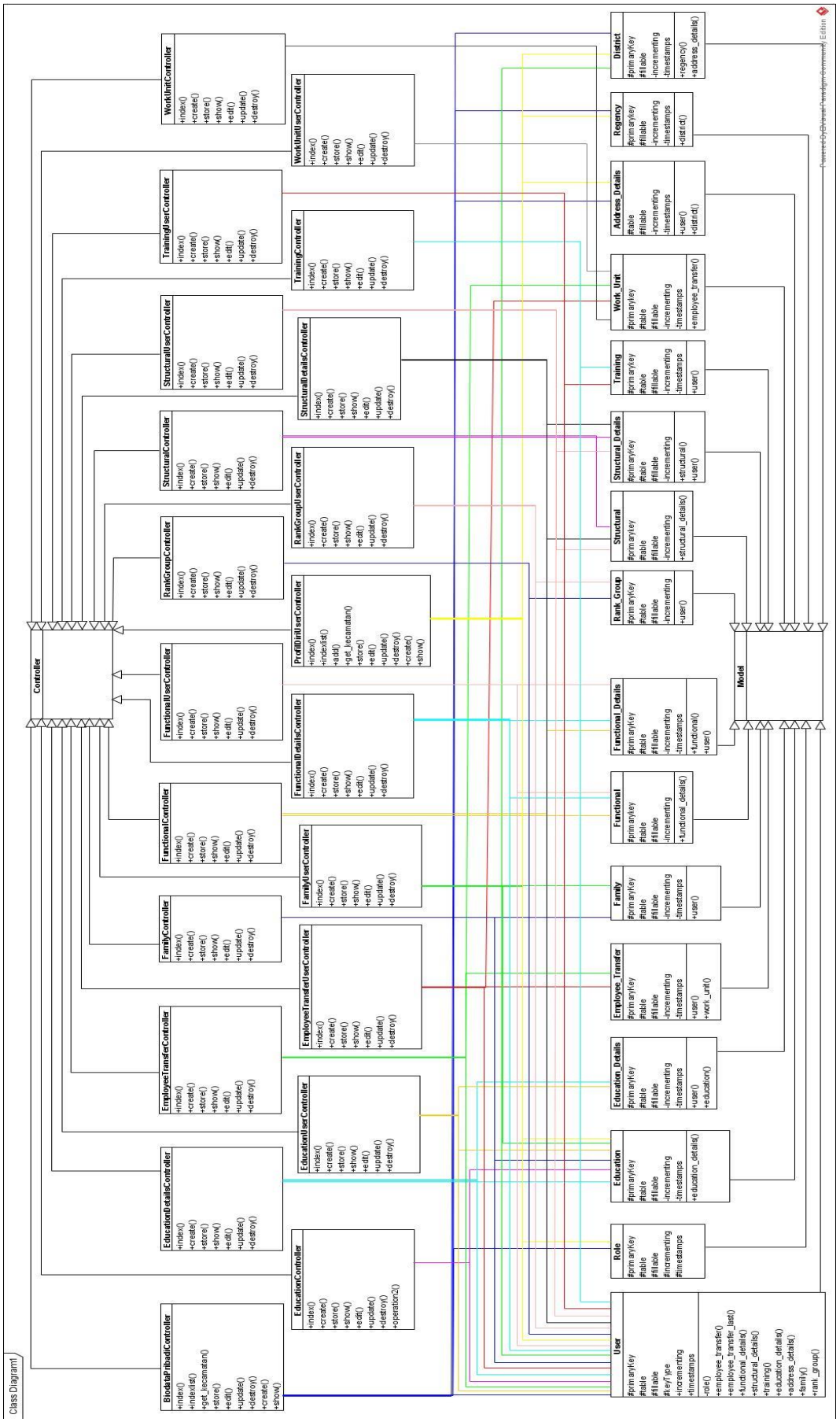
14. Employee_transfer

Nama atribut	Tipe Data	Keterangan
employee_transfer_id	varchar(10)	PK
nip_nik	varchar(20)	FK
work_unit_id	varchar(10)	FK
employee_transfer_date	Date	
sign_by	varchar(100)	
sk_no	Text	
sk_file	varchar(100)	
created_at	Timestamp	
updated_at	Timestamp	

15. Rank group

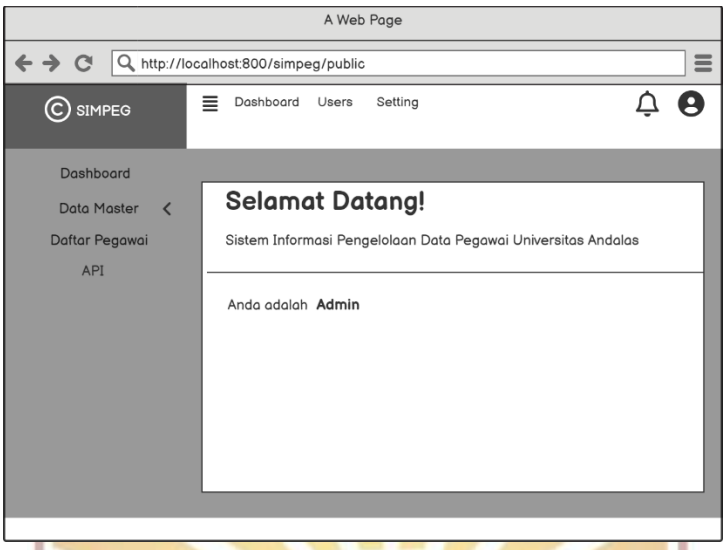
Nama atribut	Tipe Data	Keterangan
rank_group_id	varchar(10)	PK
nip_nik	varchar(20)	FK
Name	varchar(10)	
tmt	Date	
sign_by	varchar(100)	
sk_no	Text	
sk_date	Date	
status	varchar(100)	
sk_file	varchar(100)	
created_at	Timestamp	
updated_at	Timestamp	

LAMPIRAN D
(CLASS DIAGRAM)

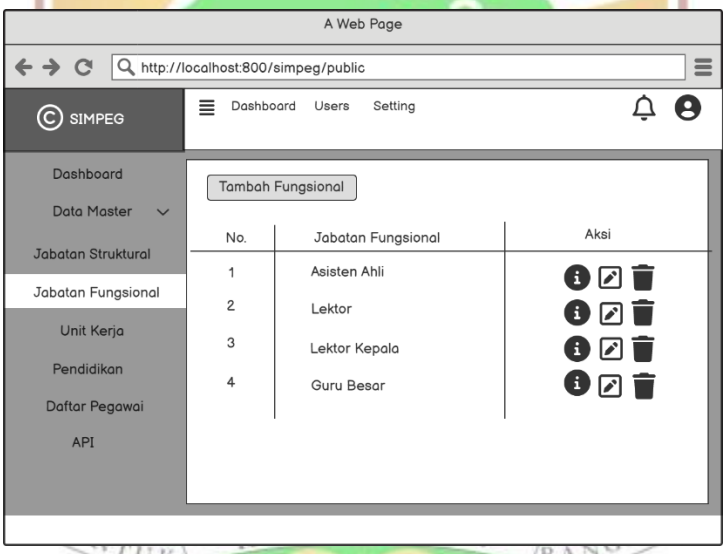

Class Diagram1
Controller
Model
BiodataPribadiController
+index()
+indexlist()
+get_kecamatan()
+store()
+edit()
+update()
+destroy()
+create()
+show()
EducationController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
+operation2()
EducationDetailsController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
EducationUserController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
EmployeeTransferController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
EmployeeTransferUserController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
FamilyController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
FamilyUserController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
FunctionalController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
FunctionalDetailsController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
FunctionalUserController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
ProfilBiruUserController
+index()
+indexlist()
+add()
+get_kecamatan()
+store()
+edit()
+update()
+destroy()
+create()
+show()
RankGroupController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
RankGroupUserController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
StructuralController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
StructuralDetailsController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
StructuralUserController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
TrainingController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
TrainingUserController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
WorkUnitController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
WorkUnitUserController
+index()
+create()
+store()
+show()
+edit()
+update()
+destroy()
User
#primaryKey
#table
#fillable
#key/Type
-incrementing
-timestamps
-role()
+employee_transfer()
+employee_transfer_last()
+functional_details()
+structural_details()
+training()
+education_details()
+address_details()
+family()
+rank_group()
Role
#primaryKey
#table
#fillable
#incrementing
#timestamps
Education
#primaryKey
#table
#fillable
-incrementing
-timestamps
+education_details()
Education_Details
#primaryKey
#table
#fillable
-incrementing
-timestamps
+user()
+education()
Employee_Transfer
#primaryKey
#table
#fillable
-incrementing
-timestamps
+user()
+work_unit()
Family
#primaryKey
#table
#fillable
-incrementing
-timestamps
+user()
Functional
#primaryKey
#table
#fillable
-incrementing
+functional_details()
Functional_Details
#primaryKey
#table
#fillable
-incrementing
-timestamps
+functional()
+user()
Rank_Group
#primaryKey
#table
#fillable
-incrementing
+user()
Structural
#primaryKey
#table
#fillable
-incrementing
+structural_details()
Structural_Details
#primaryKey
#table
#fillable
-incrementing
+structural()
+user()
Training
#primaryKey
#table
#fillable
-incrementing
-timestamps
+user()
Work_Unit
#primaryKey
#table
#fillable
-incrementing
-timestamps
+employee_transfer()
Address_Details
#table
#fillable
-incrementing
-timestamps
+user()
+districtId()
Regency
#primaryKey
#fillable
-incrementing
-timestamps
+districtId()
District
#primaryKey
#fillable
-incrementing
-timestamps
+regency()
+address_details()

LAMPIRAN E
(USER INTERFACE)

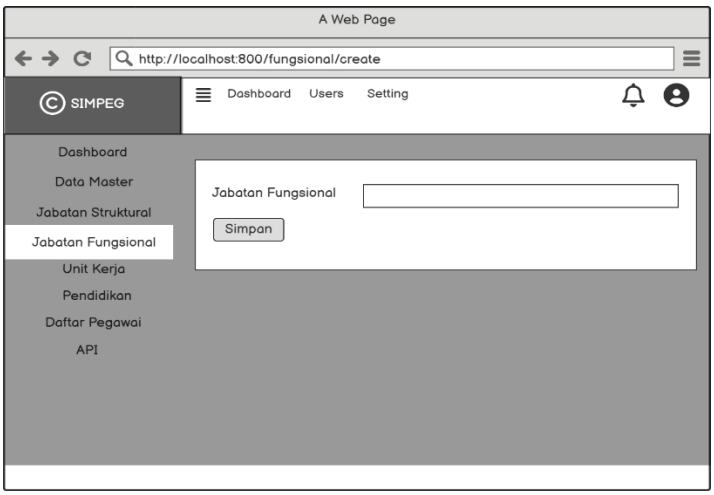
1. Perancangan antarmuka *Homepage* admin



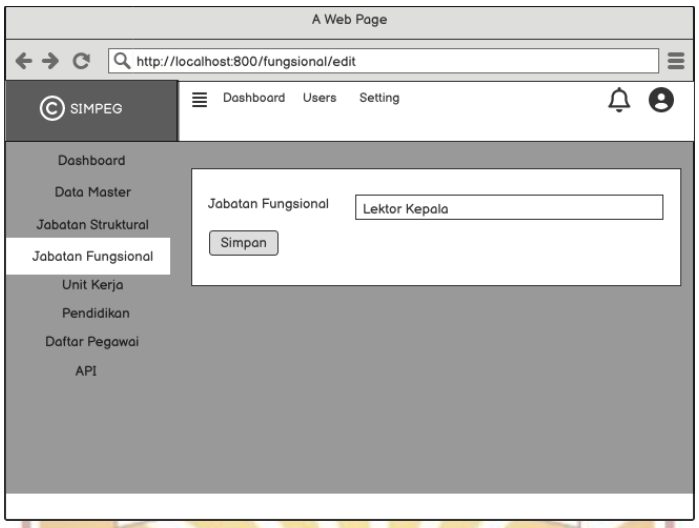
2. Perancangan antarmuka daftar jabatan fungsional admin



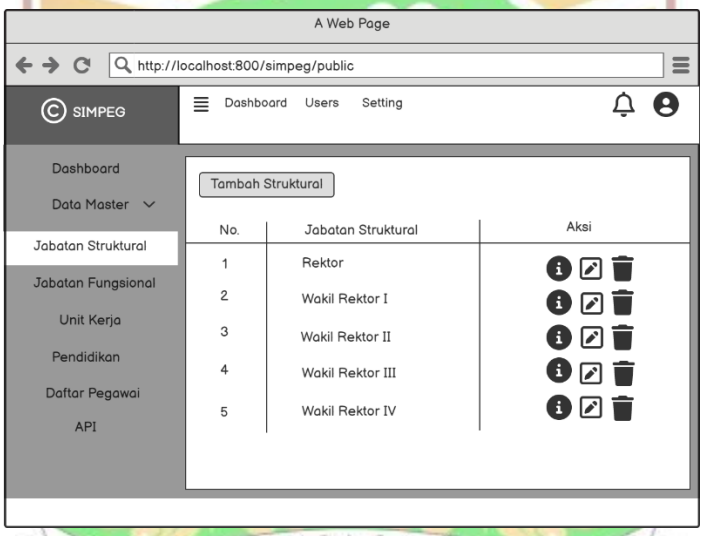
3. Perancangan antarmuka tambah jabatan fungsional admin



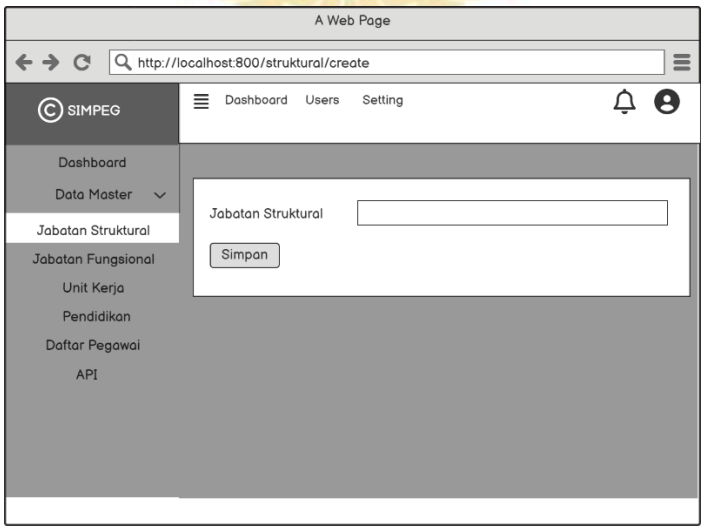
4. Perancangan antarmuka edit jabatan fungsional admin



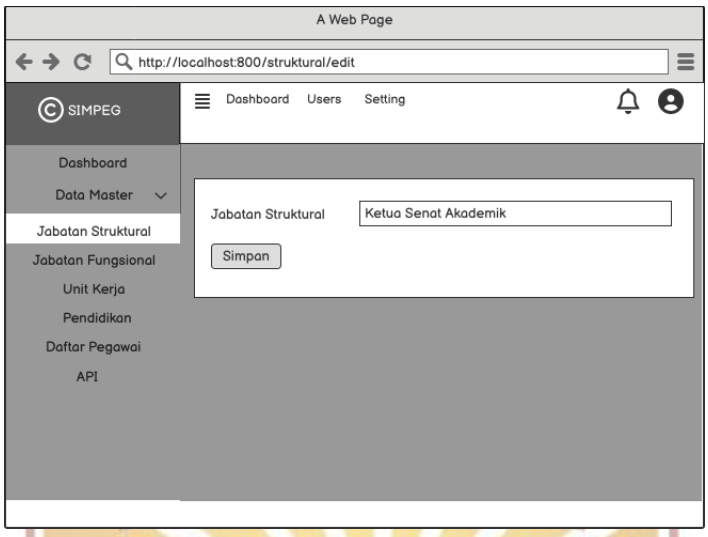
5. Perancangan antarmuka daftar jabatan struktural admin



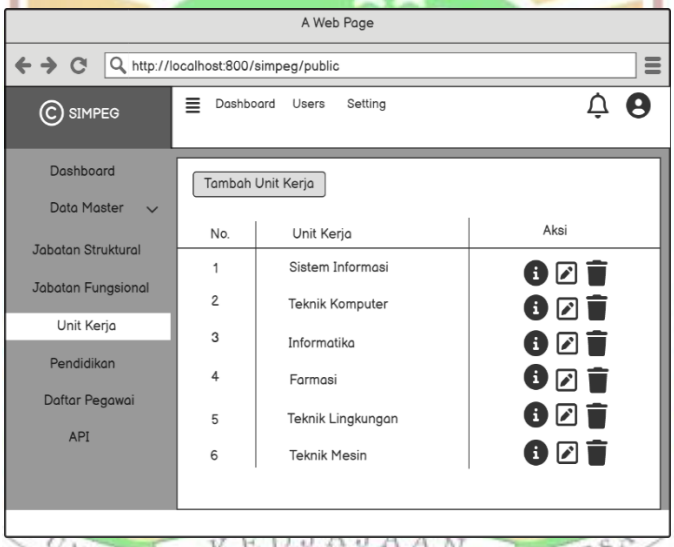
6. Perancangan antarmuka tambah jabatan struktural admin



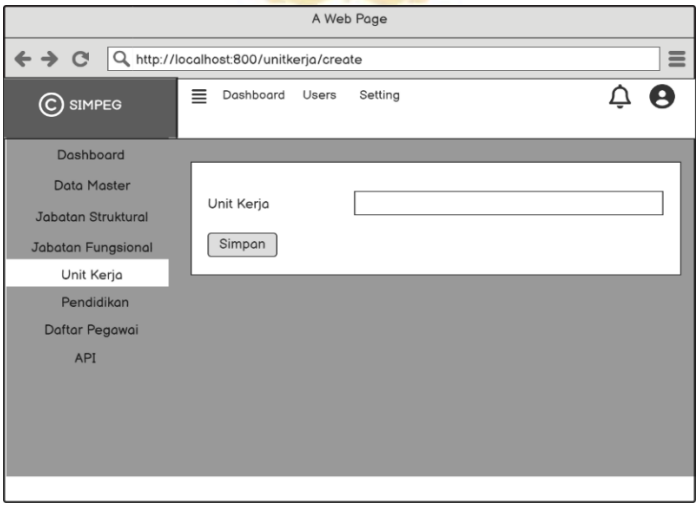
7. Perancangan antarmuka edit jabatan struktural admin



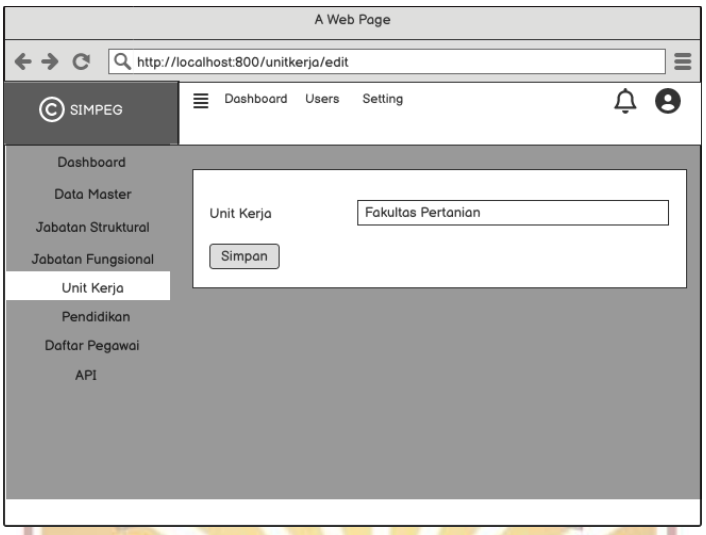
8. Perancangan antarmuka daftar unit kerja admin



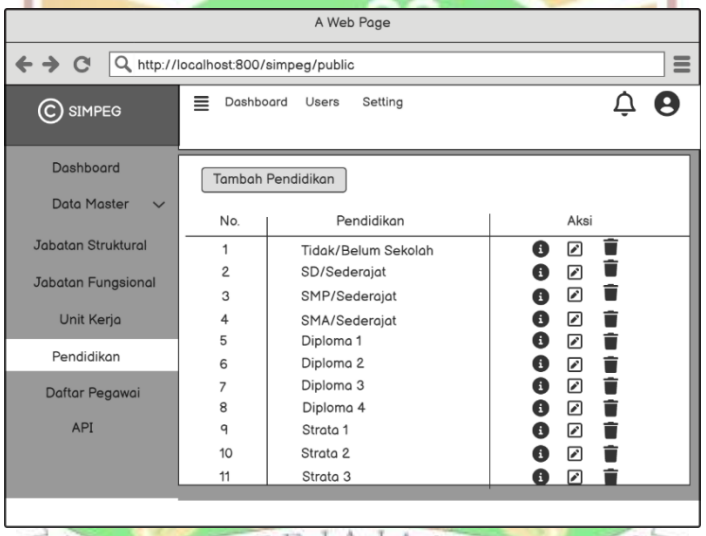
9. Perancangan antarmuka tambah unit kerja admin



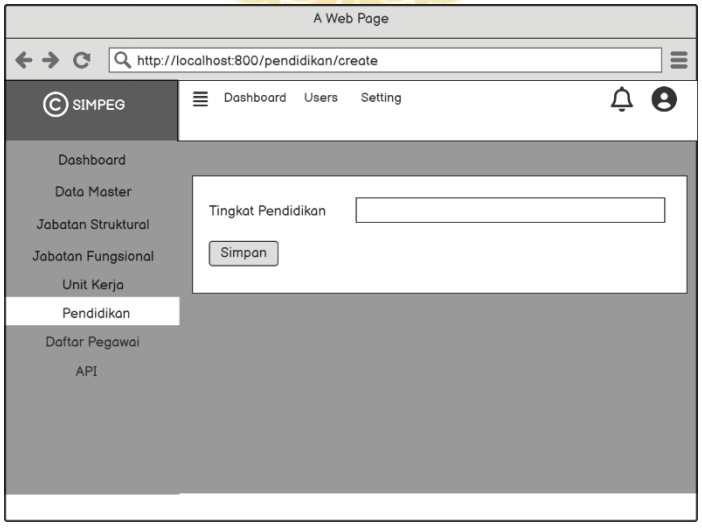
10. Perancangan antarmuka edit unit kerja admin



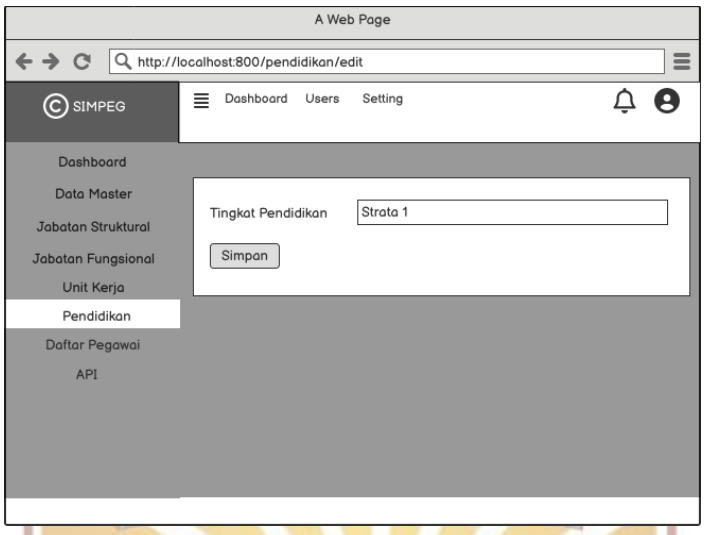
11. Perancangan antarmuka daftar pendidikan admin



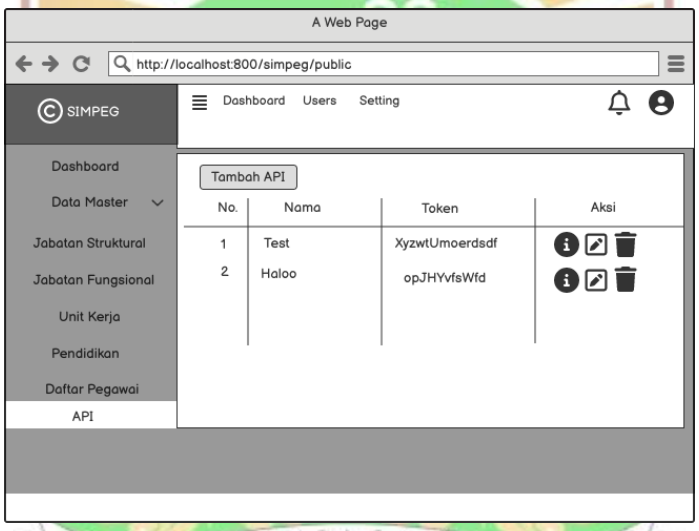
12. Perancangan antarmuka tambah pendidikan admin



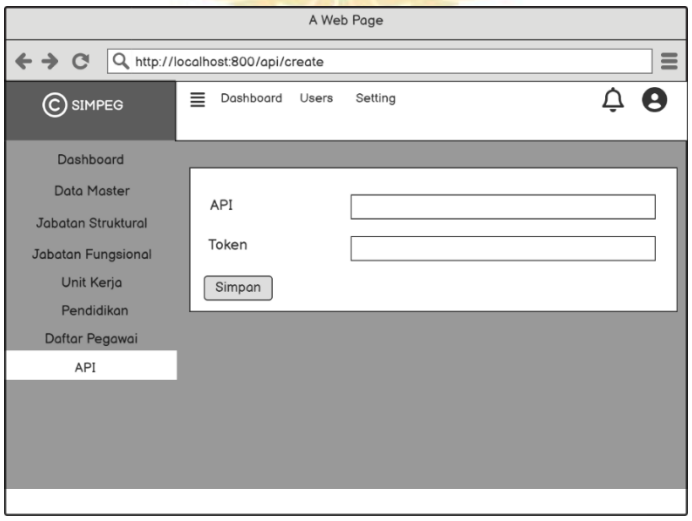
13. Perancangan antarmuka edit pendidikan admin



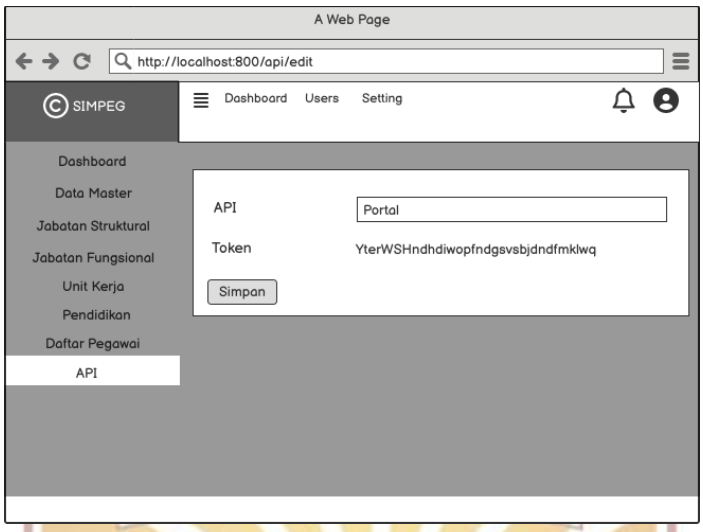
14. Perancangan antarmuka daftar API admin



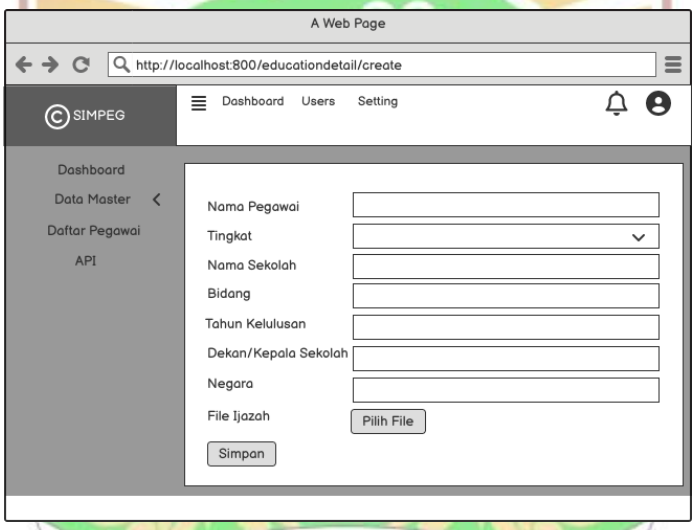
15. Perancangan antarmuka tambah API admin



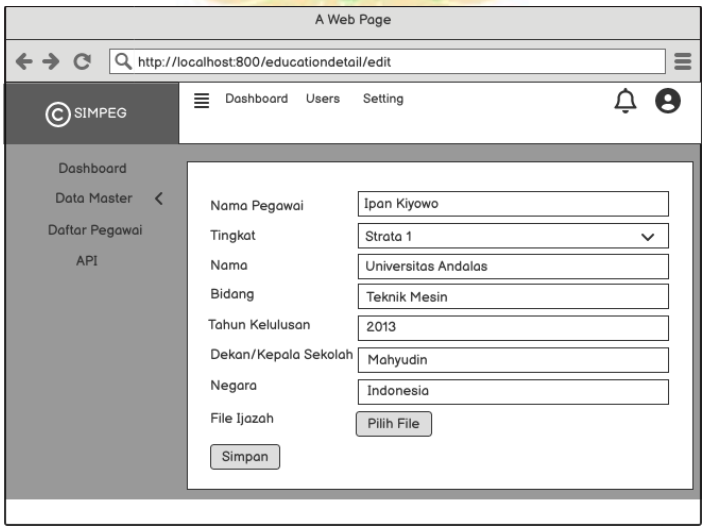
16. Perancangan antarmuka edit API admin



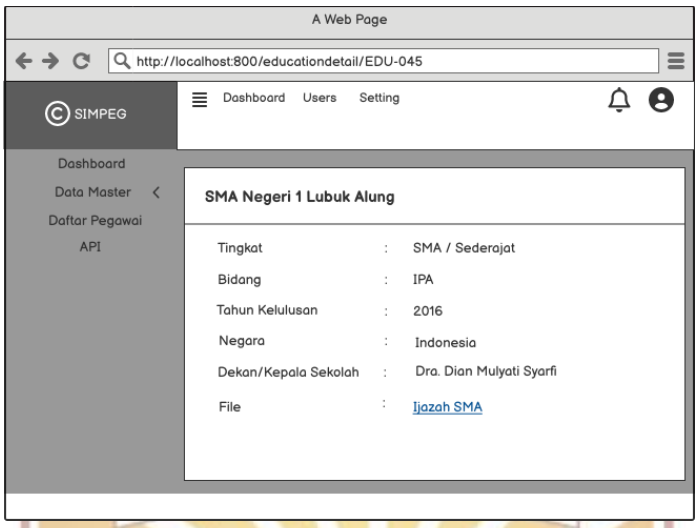
17. Perancangan antarmuka tambah riwayat pendidikan admin



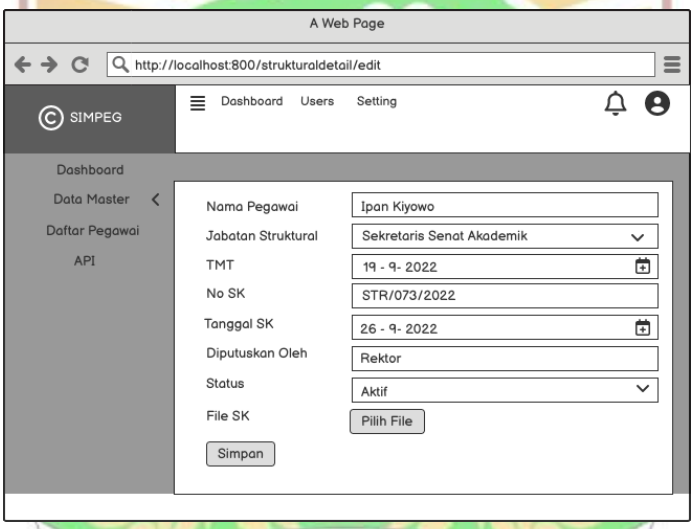
18. Perancangan antarmuka edit riwayat pendidikan admin



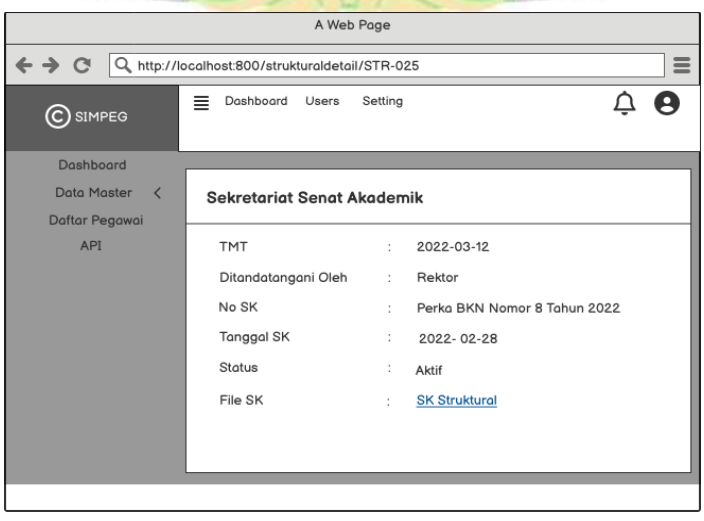
19. Perancangan antarmuka lihat riwayat pendidikan admin



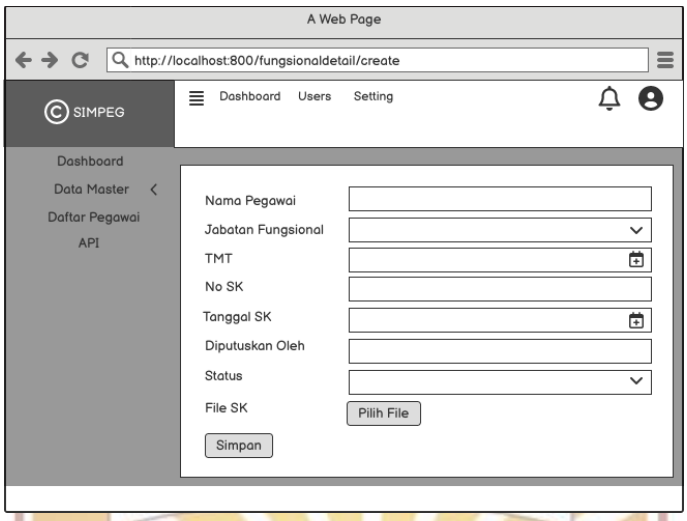
20. Perancangan antarmuka edit riwayat jabatan strukural admin



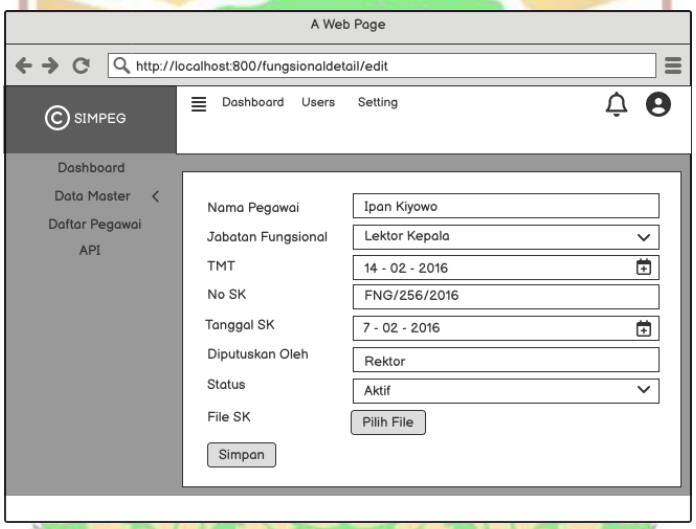
21. Perancangan antarmuka lihat riwayat jabatan struktural admin



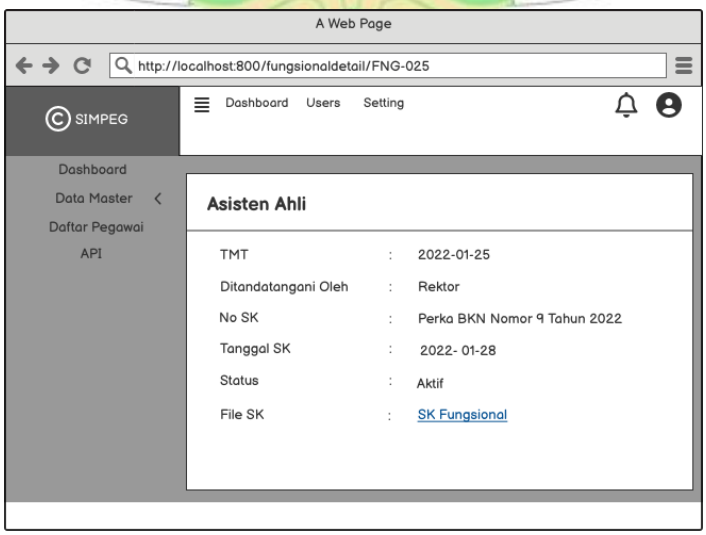
22. Perancangan antarmuka tambah riwayat jabatan fungsional admin



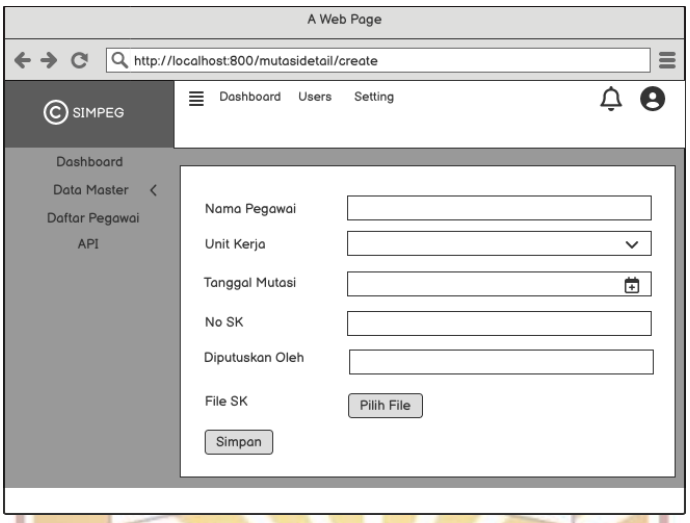
23. Perancangan antarmuka edit riwayat jabatan fungsional admin



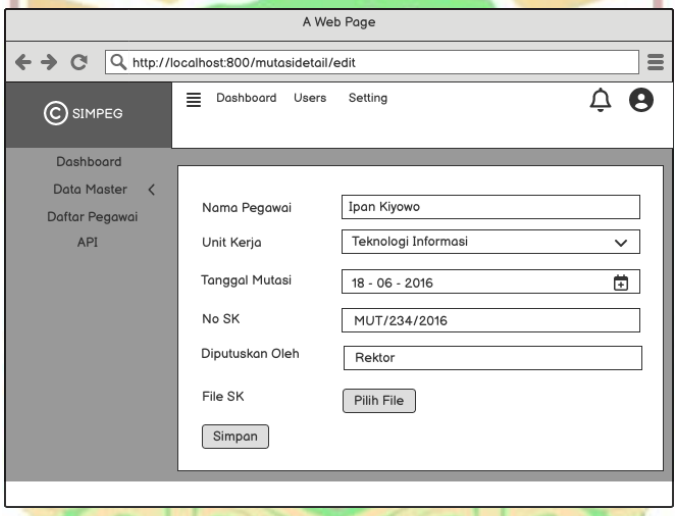
24. Perancangan antarmuka lihat jabatan fungsional admin



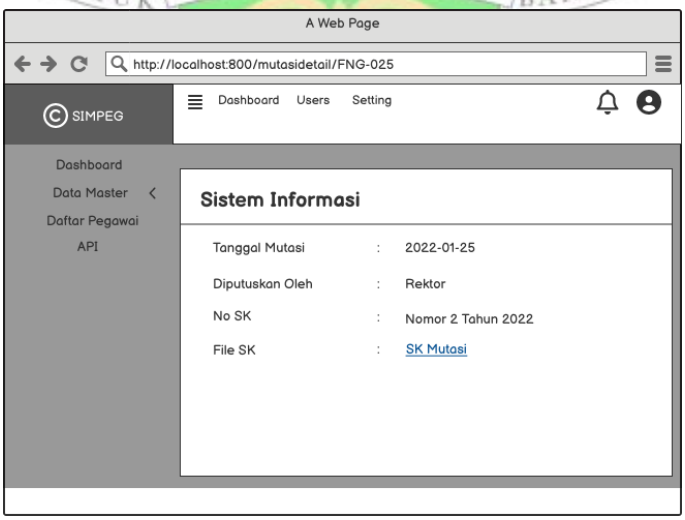
25. Perancangan antarmuka tambah riwayat mutasi admin



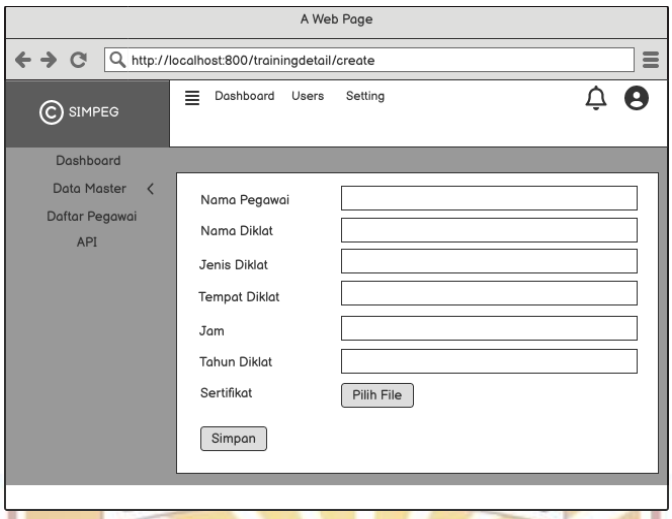
26. Perancangan antarmuka edit riwayat mutasi admin



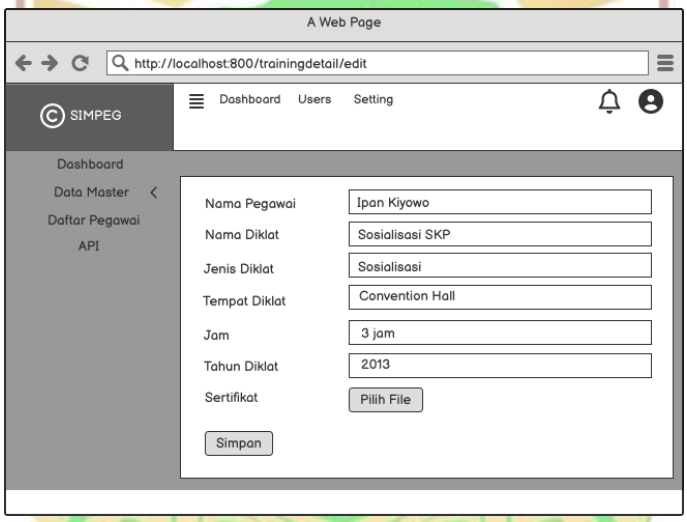
27. Perancangan antarmuka lihat riwayat mutasi admin



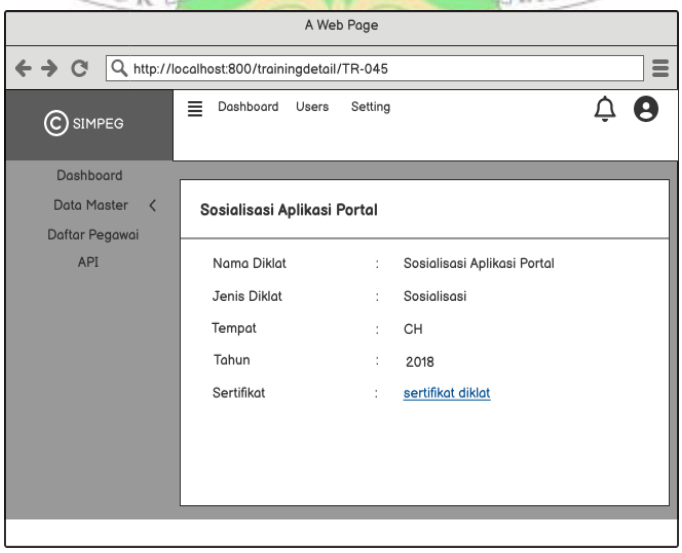
28. Perancangan antarmuka tambah riwayat diklat admin



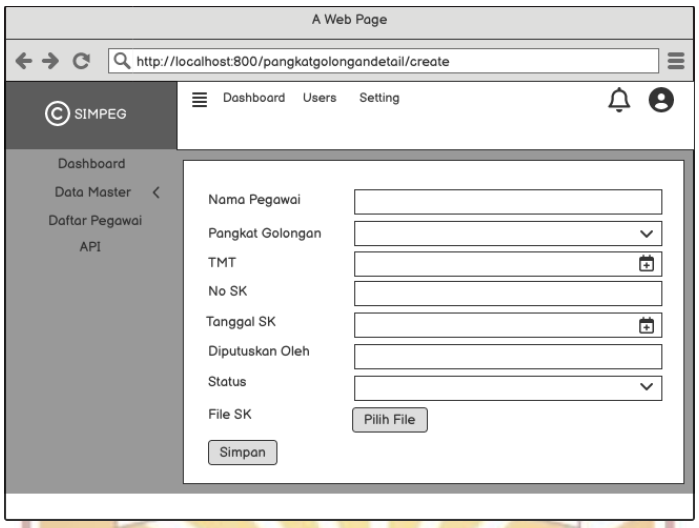
29. Perancangan antarmuka edit riwayat diklat admin



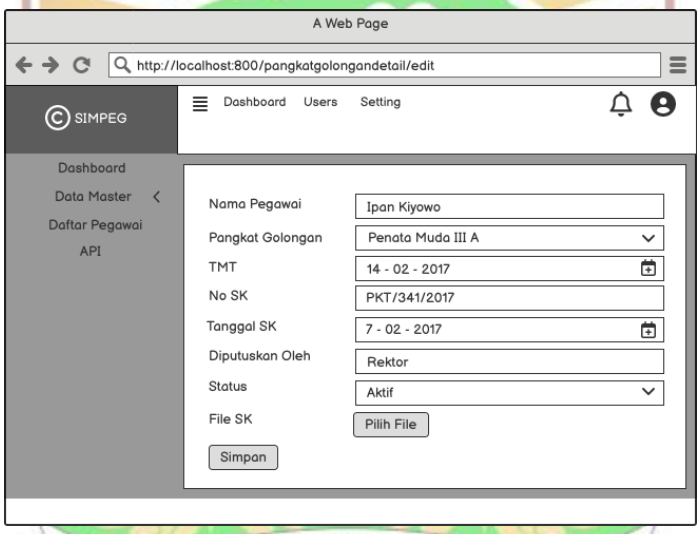
30. Perancangan antarmuka lihat riwayat diklat admin



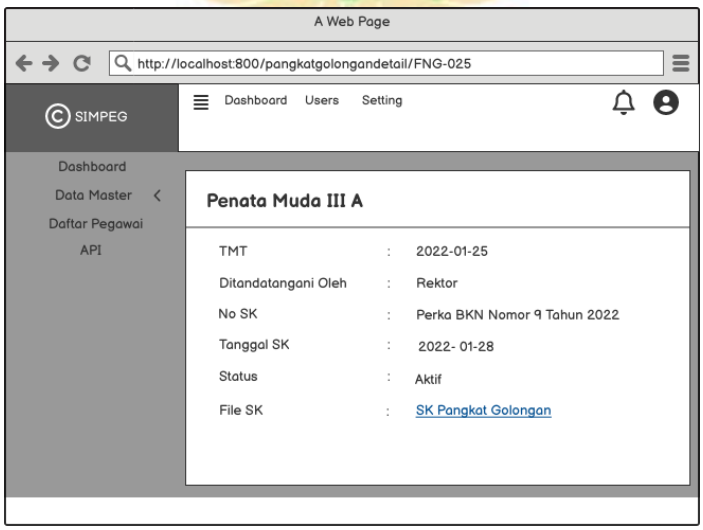
31. Perancangan antarmuka tambah riwayat pangkat golongan admin



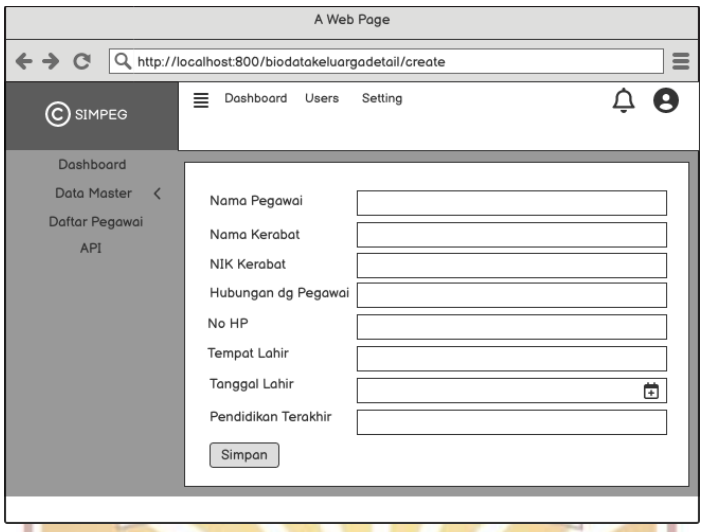
32. Perancangan antarmuka edit riwayat pangkat golongan admin



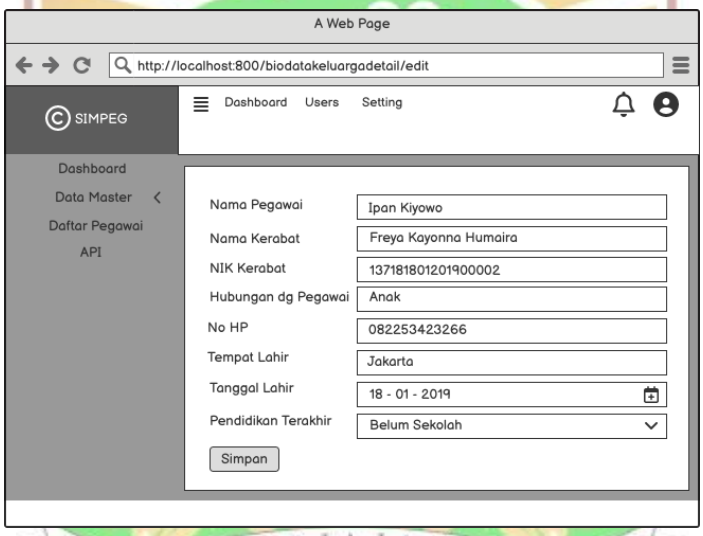
33. Perancangan antarmuka lihat riwayat pangkat golongan admin



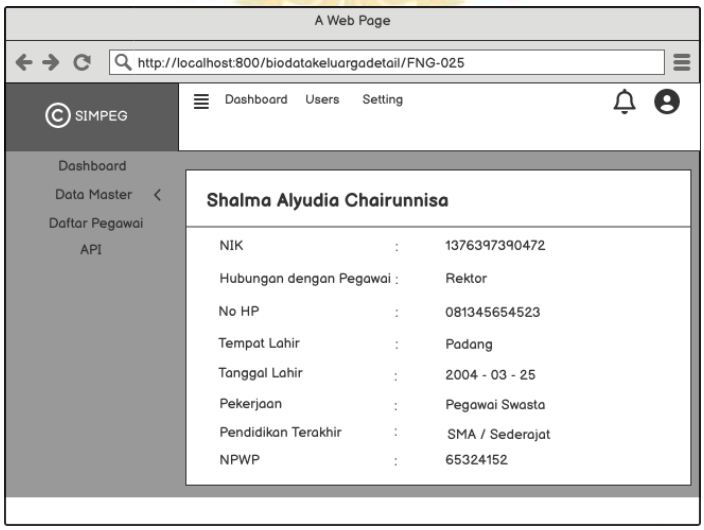
34. Perancangan antarmuka tambah biodata keluarga admin



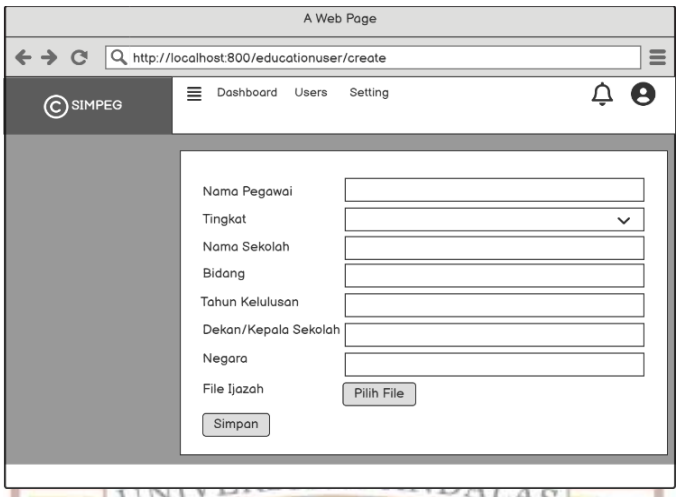
35. Perancangan antarmuka edit biodata keluarga admin



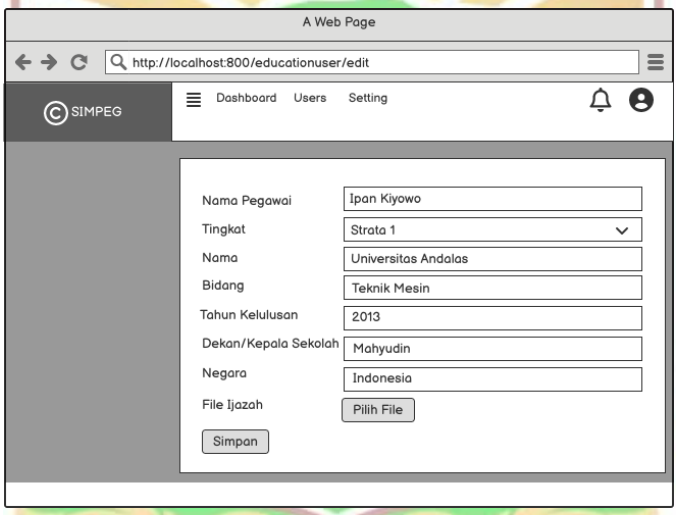
36. Perancangan antarmuka lihat biodata keluarga admin



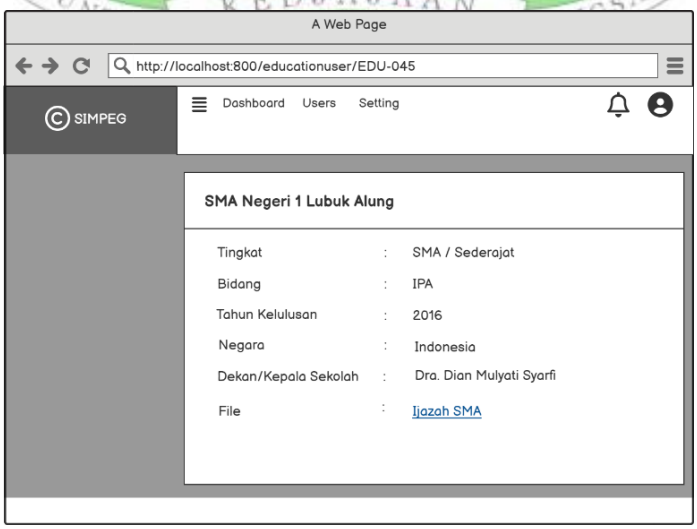
37. Perancangan antarmuka tambah riwayat pendidikan user



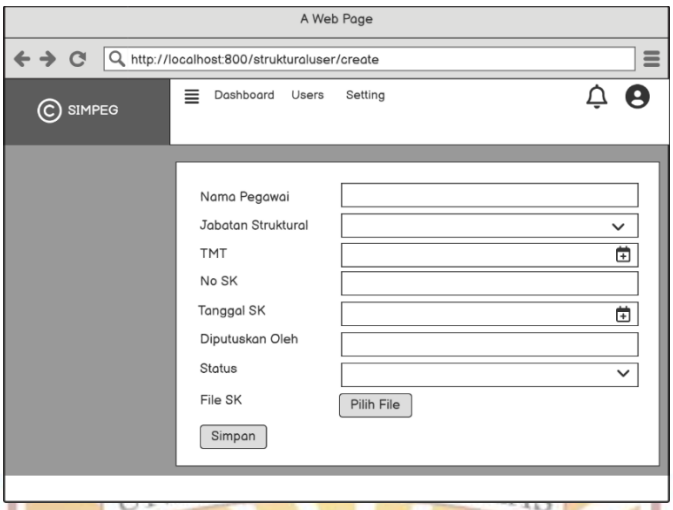
38. Perancangan antarmuka edit riwayat pendidikan user



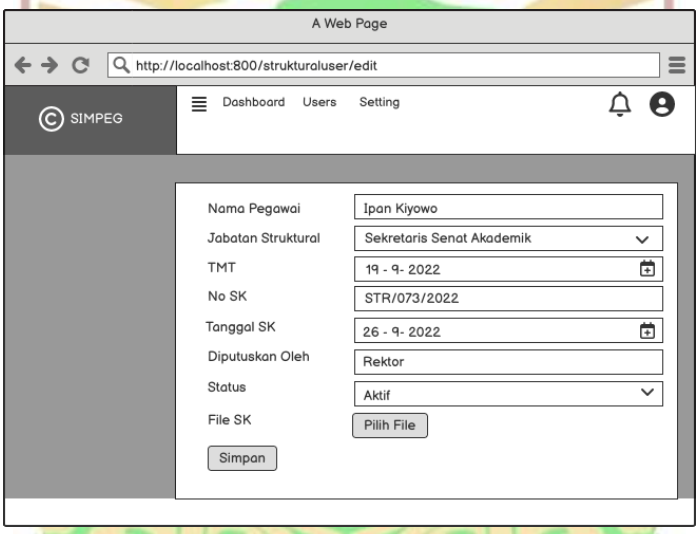
39. Perancangan antarmuka lihat riwayat pendidikan user



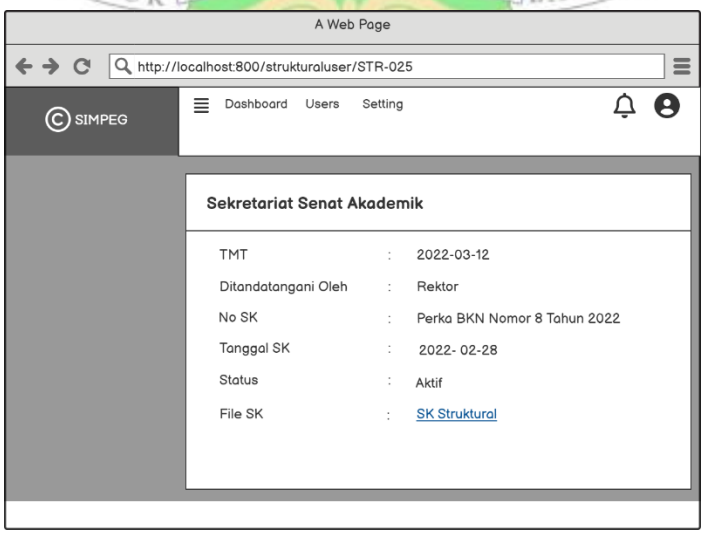
40. Perancangan antarmuka tambah riwayat jabatan struktural user



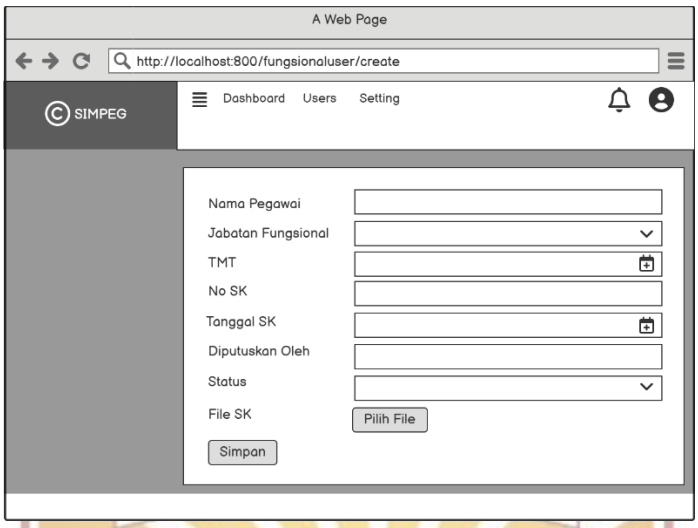
41. Perancangan antarmuka edit riwayat jabatan strukural user



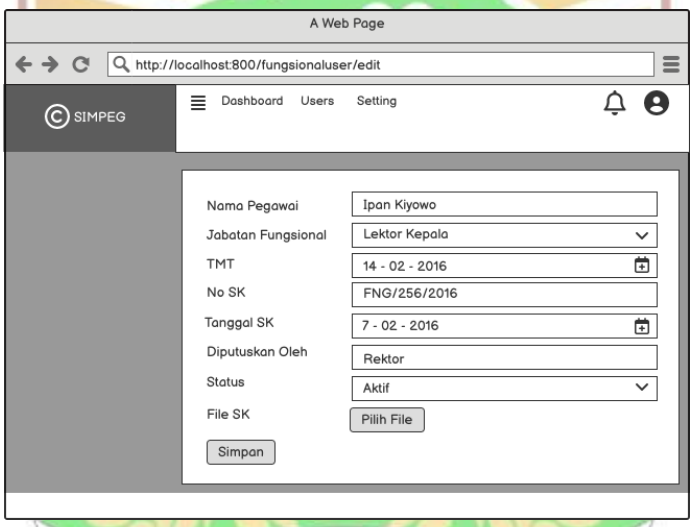
42. Perancangan antarmuka lihat riwayat jabatan struktural user



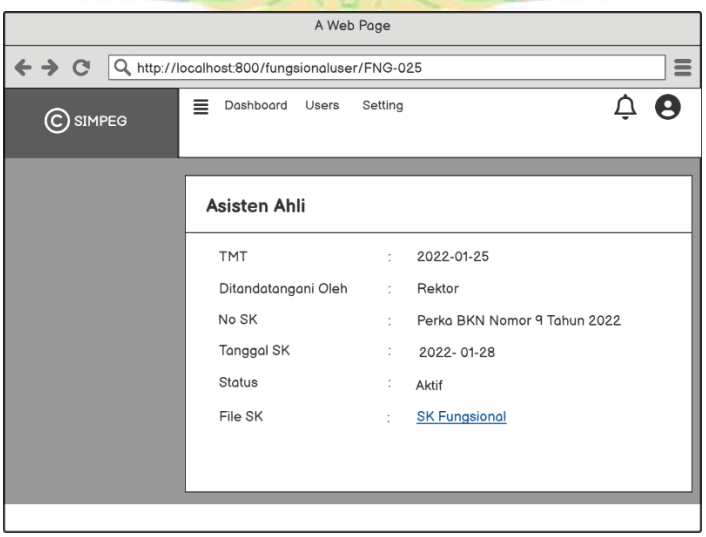
43. Perancangan antarmuka tambah riwayat jabatan fungsional user



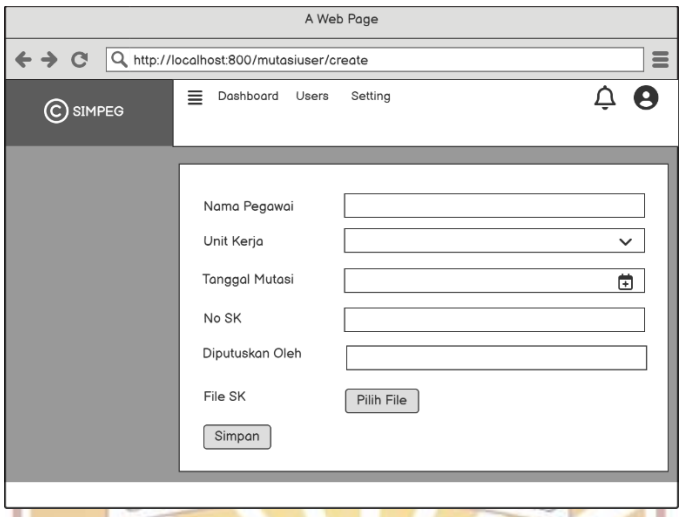
44. Perancangan antarmuka edit riwayat jabatan fungsional user



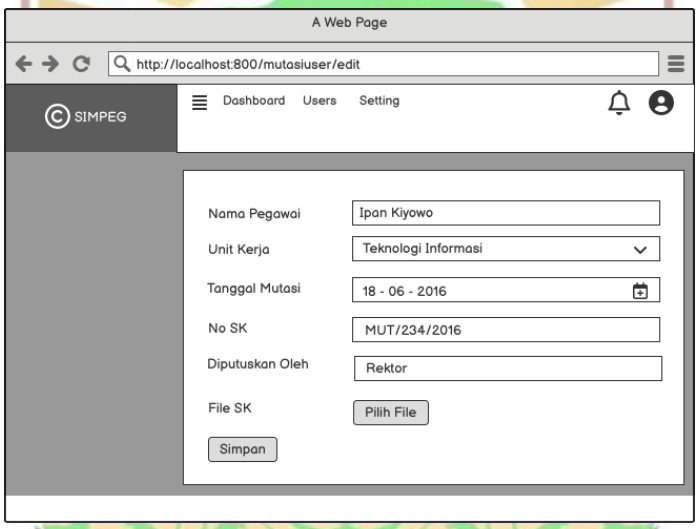
45. Perancangan antarmuka lihat jabatan fungsional user



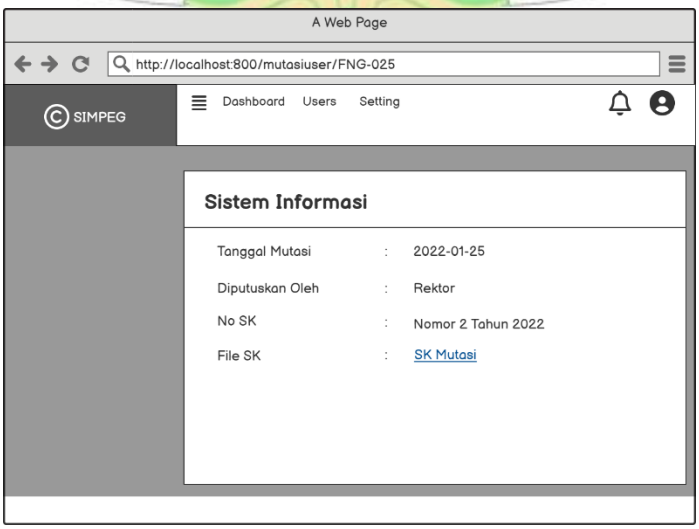
46. Perancangan antarmuka tambah riwayat mutasi user



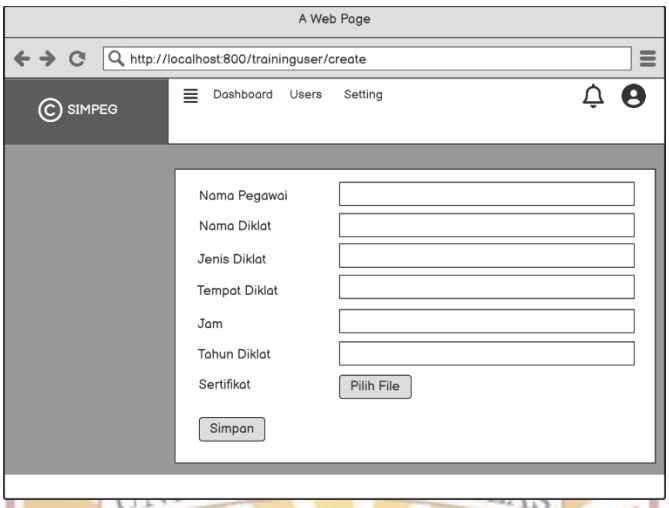
47. Perancangan antarmuka edit riwayat mutasi user



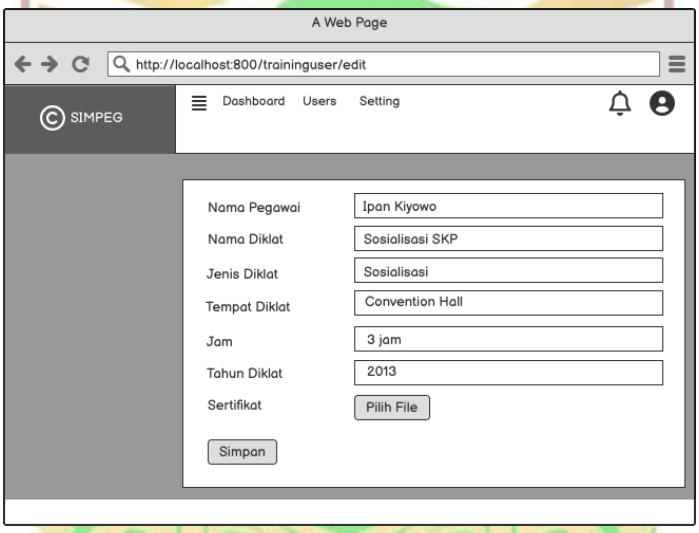
48. Perancangan antarmuka lihat riwayat mutasi user



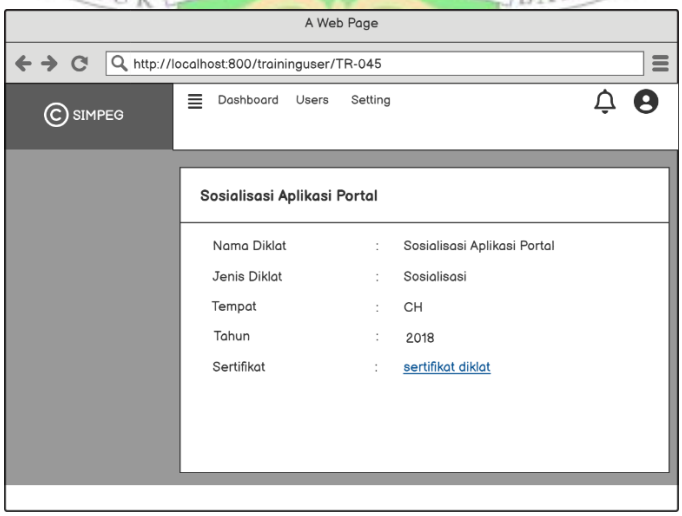
49. Perancangan antarmuka tambah riwayat diklat user



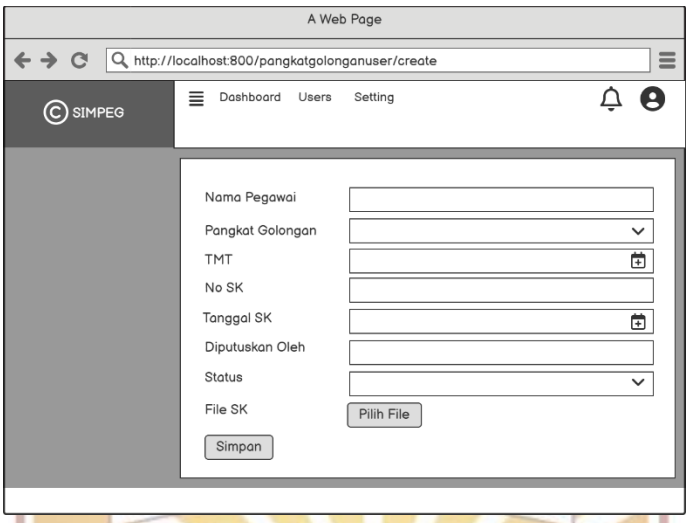
50. Perancangan antarmuka edit riwayat diklat user



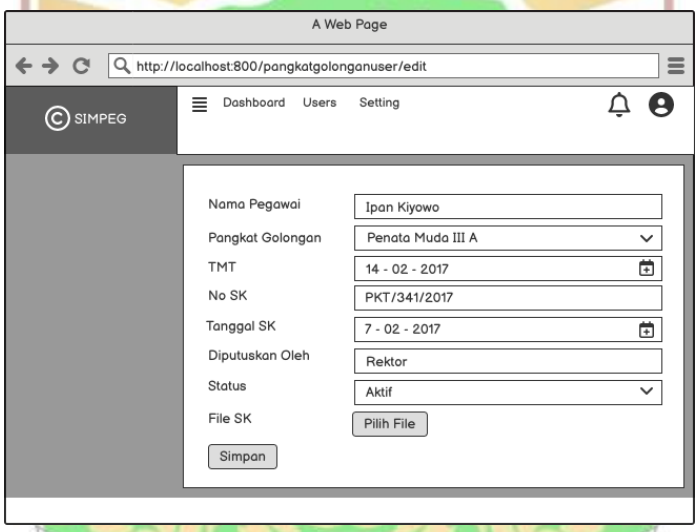
51. Perancangan antarmuka lihat riwayat diklat user



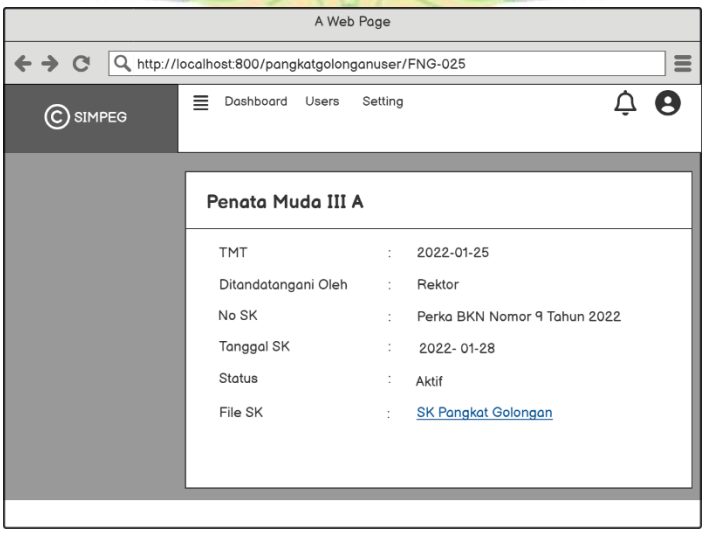
52. Perancangan antarmuka tambah riwayat pangkat golongan user



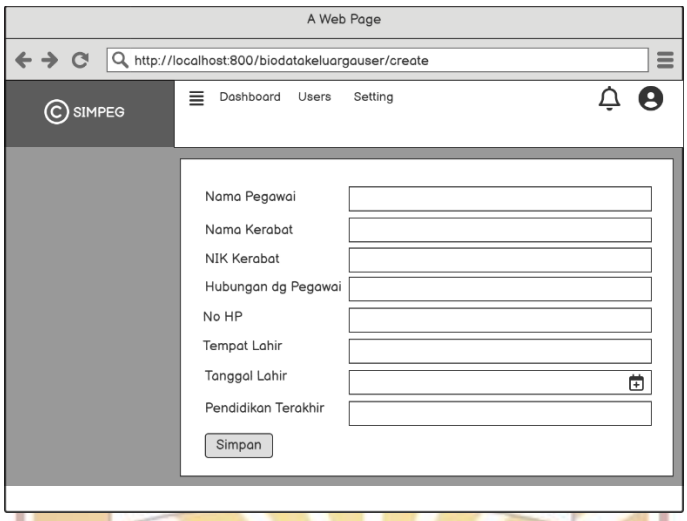
53. Perancangan antarmuka edit riwayat pangkat golongan user



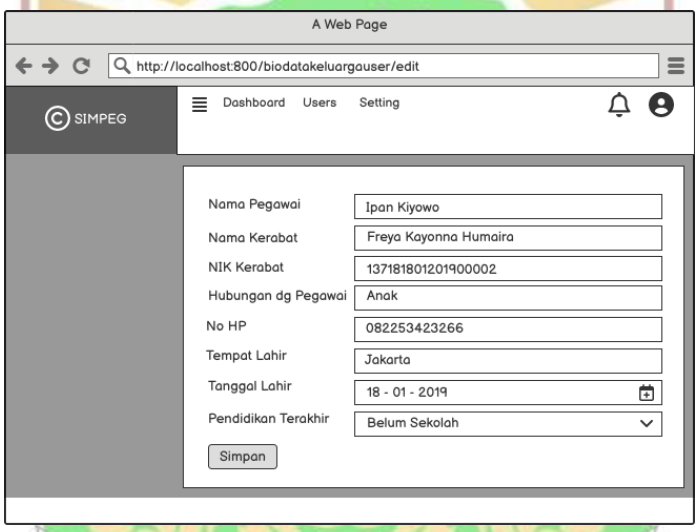
54. Perancangan antarmuka lihat riwayat pangkat golongan user



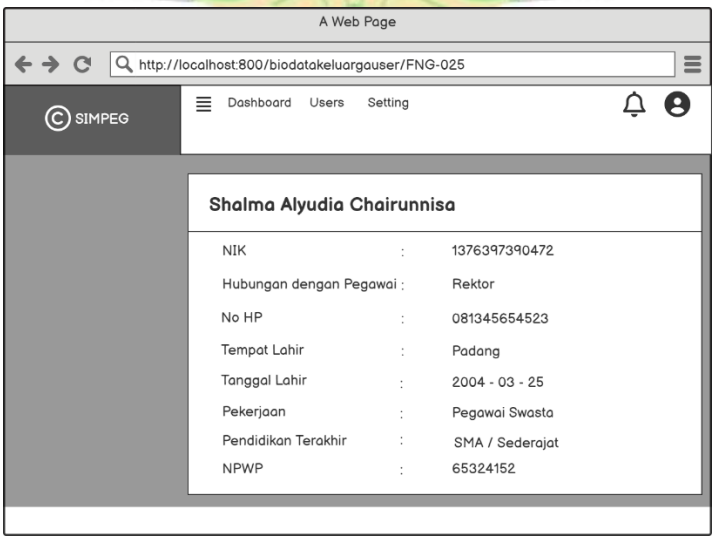
55. Perancangan antarmuka tambah biodata keluarga user



56. Perancangan antarmuka edit biodata keluarga user



57. Perancangan antarmuka lihat biodata keluarga user



LAMPIRAN F
(KODE PROGRAM)

1. Model *User.php*

```
 protected $primaryKey = 'nip_nik';
 protected $table = 'users';
 protected $fillable = ['nip_nik','nidn', 'username', 'role_id', 'password', 'ti
tle_ahead','real_name', 'back_title', 'birth_place', 'birth_date','photo','group
','blood_group','height','weight','email','id_card_number','phone_number','npwp'
,'bpjs','gender','religion','marital_status','citizen_status','retirement_age_li
mit','employee_status'];
 public $incrementing = false;
 public $timestamps = false;
 protected $keyType = 'string';
 public function role(){
    return $this->belongsTo(Role::class, 'role_id', 'id');    }
 public function employee_transfer(){
    return $this->hasMany(Employee_Transfer::class, 'nip_nik', 'nip_nik');}
 public function employee_transfer_last(){
    return $this->employee_transfer()->latest('employee_transfer_date')-
>first();}
 public function functional_details(){
    return $this->hasMany(Functional_Details::class, 'nip_nik', 'nip_nik');}
 public function structural_details(){
    return $this->hasMany(Structural_Details::class, 'nip_nik', 'nip_nik');   }
 public function training(){
    return $this->hasMany(Training::class, 'nip_nik', 'nip_nik'); }
 public function bank(){
    return $this->hasMany(Bank::class, 'nip_nik', 'nip_nik');    }
 public function education_details(){
    return $this->hasMany(Education_Details::class, 'nip_nik', 'nip_nik');}
 public function address_details(){
    return $this->hasMany(Address_Details::class, 'nip_nik', 'nip_nik'); }
 public function family(){
    return $this->hasMany(Family::class, 'nip_nik', 'nip_nik');}
 public function rank_group(){
    return $this->hasMany(Rank_Group::class, 'nip_nik', 'nip_nik'); }
 public function getAvatarUrlAttribute(){$patlink = rtrim(app()-
>basePath('public/storage'), '/');
 if($this->photo && is_dir($patlink) && Storage::disk('public')->exists($this-
>photo)){
 return url("/storage/".$this->photo);   }
return "https://uxwing.com/wp-content/themes/uxwing/download/12-people-
gesture/avatar.png";}
protected $hidden = ['password', 'remember_token', ];
protected $casts = [ 'email_verified_at' => 'datetime', ];}
```

2. Model *Work_unit.php*

```
class Work_Unit extends Model
{
 protected $primaryKey = 'work_unit_id';
 protected $table = 'work_units';
 protected $fillable = ['work_unit_id', 'name'];
 public $incrementing = false;
 public $timestamps = false;
 public function employee_transfer(){
 return $this->hasMany (Employee_Transfer::class,'work_unit_id',
'work_unit_id');}
 use AutoNumberTrait;
 public function getAutoNumberOptions(){ return [
'work_unit_id' => [
'format' => 'WU-?',
'length' => 5  ] ]; }
```

3. Model *Training.php*

```
class Training extends Model{
    protected $table = 'trainings';
    protected $primaryKey = 'training_id';
    protected $fillable = ['training_id', 'nip_nik', 'hour', 'training_name','ty
pe_of_training','place', 'year', 'certificate_file'];
    public $incrementing = false;
    public $timestamps = false;
    public function user(){
```

```
    return $this->belongsTo(Employee_Transfer::class, 'nip_nik', 'nip_nik');}
    use AutoNumberTrait;
    public function getAutoNumberOptions() {
    return [
    'training_id' => [
    'format' => 'TR-?',
    'length' => 4 ] ];} }
```

4. Model *Structural.php*

```
class Structural extends Model
{
   protected $table = 'structurals';
   protected $primaryKey = 'structural_id';
   public $incrementing = false;
   protected $fillable = ['structral_id', 'information'];
   public function structural_details (){
   return $this->hasMany(Structural_Details ::class); }
   use AutoNumberTrait;
   public function getAutoNumberOptions() {
   return [
   'structural_id' => [
   'format' => 'STR-?',
   'length' => 3 ] ]; } }
```

5. Model *Rankgroup.php*

```
class Rank_Group extends Model
{
  protected $table = 'rank_groups';
  protected $primaryKey = 'rank_group_id';
  public $incrementing = false;
  protected $fillable = ['rank_group_id', 'nip_nik', 'name', 'tmt', 'sk_no', 'sk
_date','sign_by','status','sk_file'];
  public function user(){
  return $this->belongsTo(User::class, 'nip_nik', 'nip_nik'); }
  use AutoNumberTrait;
  public function getAutoNumberOptions() {
  return [
  'rank_group_id' => [
  'format' => 'RG-?',
  'length' => 4   ] ]; }}
```

6. Model *Functional.php*

```
class Functional extends Model
{
  protected $table = 'functionals';
  protected $primaryKey = 'functional_id';
  public $incrementing = false;
  protected $fillable = ['functional_id', 'information',];
  public function functional_details(){
  return $this->hasMany(Functional_Details::class); }
  use AutoNumberTrait;
  public function getAutoNumberOptions() {
  return [
  'functional_id' => [
  'format' => 'FUNG-?',
  'length' => 3   ] ]; } }
```

7. Model *Functional_Details.php*

```
class Functional_Details extends Model{
 protected $table = 'functional_details';
 protected $primaryKey = 'functional_id';
 public $incrementing = false;
 protected $fillable = ['nip_nik', 'functional_id', 'tmt', 'sign_by','sk_no','sk
_date','status', 'sk_file', ];
 public $timestamps = false;
 public function functional(){
 return $this->belongsTo(Functional::class, 'functional_id', 'functional_id'); }
 public function user(){
 return $this->belongsTo(User::class, 'nip_nik', 'nip_nik'); } }
```

8. Model *Family.php*

```
class Family extends Model
{
  protected $primaryKey = 'id_number';
  protected $table = 'family';
  protected $fillable = ['nip_nik','name','relationship','username', 'phone_numb
er', 'birth_date',
 'birth_place','occupation','last_education','npwp_no'];
  public $incrementing = false;
  public $timestamps = false;
  public function user(){
  return $this->belongsTo(User::class, 'nip_nik', 'nip_nik');   }
  use AutoNumberTrait;
  public function getAutoNumberOptions() {
  return [
  'id_number' => [
  'format' => 'Kel-?',
  'length' => 3 ] ];} }
```

9. Model *Employee_Transfer.php*

```
class Employee_Transfer extends Model
{
  protected $primaryKey = 'employee_transfer_id';
  protected $table = 'employee_transfers';
  protected $fillable = ['employee_transfer_id', 'nip_nik','work_unit_id','emplo
yee_transfer_date','sk_no','sign_by','sk_file'];
  public $incrementing = false;
  public $timestamps = false;
  public function user(){
  return $this->belongsTo(User::class, 'nip_nik', 'nip_nik');   }
  public function work_unit(){
  return $this->belongsTo(Work_Unit::class, 'work_unit_id', 'work_unit_id');  }
  use AutoNumberTrait;
  public function getAutoNumberOptions()   {
  return [
  'employee_transfer_id' => [
  'format' => 'MUT-?',
  'length' => 4  ] ]; } }
```

10. Model *Education.php*

```
class Education extends Model
{
  protected $primaryKey = 'education_id';
  protected $table = 'education';
  protected $fillable = ['education_id', 'level'];
  public $incrementing = false;
  public $timestamps = false;
  public function education_details(){
  return $this-
>belongsTo(Education_Details::class, 'education_id', 'education_id');}
  use AutoNumberTrait;
  public function getAutoNumberOptions()  {
  return [
  'education_id' => [
  'format' => 'EDU-?',
  'length' => 3    ] ]; } }
```

11. Model *Education_Details.php*

```
class Education_Details extends Model
{
 protected $primaryKey = 'education_details_id';
 protected $table = 'education_details';
 protected $fillable = ['education_details_id','education_id', 'nip_nik', 'name'
,'major','graduation_year','country','dean_headmaster','certificate_file'];
 public $incrementing = false;
 public $timestamps = false;
 public function user(){
 return $this->belongsTo(User::class, 'nip_nik', 'nip_nik');  }
 public function education() {return $this->belongsTo
(Education::class,'education_id'); }
```

```
use AutoNumberTrait;
public function getAutoNumberOptions()    { return [
 'education_details_id' => [
 'format' => 'EDU-?',
 'length' => 4   ] ];    } }
```

12. WorkUnitController

```
class WorkUnitController extends Controller
{
  public function index()    {
  $work_units = Work_Unit::all();
  return view('unitkerja.index', compact('work_units')); }
  public function create() {
  $work_units = Work_Unit::all();
  return view('unitkerja.create'); }
  public function store(Request $request) {
  Work_Unit::create([
  'name' => $request->name,     ]);
  return redirect()->route('unitkerja.index')-
>with(['success' => 'Data Berhasil Disimpan']); }
  public function show($id)  {
  $work_unit = Work_Unit::find($id);
  return view('unitkerja.show', compact('work_unit')); }
  public function edit($id)  {
  $work_unit = Work_Unit::find($id);
  return view('unitkerja.edit', compact('work_unit')); }
  public function update(Request $request, $id)     {
  $work_unit = Work_Unit::find($id);
  $work_unit->update([
  'name' => $request->name, ]);
  return redirect()->route('unitkerja.index')-
>with(['success' => 'Data Berhasil Disimpan']);    }
  public function destroy($id) {
  $work_unit = Work_Unit::find($id);
  $work_unit->delete();
  return redirect()->route('unitkerja.index')-
>with(['error' => 'Data Berhasil Dihapus']);  }}
```

13. TrainingUserController.php

```
class TrainingUserController extends Controller
{
  public function index(){
  $trainings = Training::all();
  return view('traininguser.index', compact('trainings')); }
  public function create(Request $request)
{
  if($request->all()){
  $nipnik = $request->nip_nik;
  }else{
  $nipnik = Auth::user()->nip_nik; }
  $biodatapegawai = User::find($nipnik);
  return view('traininguser.create', compact('nipnik', 'biodatapegawai'));; }
  public function store(Request $request) {
  $extension = $request->certificate_file->extension();
  if ($extension == "pdf"){
  $fileName = $request->certificate_file->getClientOriginalName();
  $trainings= Training::create([
  'training_id' => $request->training_id,
  'nip_nik' => $request->nip_nik,
  'training_name' => $request->training_name,
  'type_of_training' => $request->type_of_training,
  'place' => $request->place,
  'hour' => $request->hour,
  'year' => $request->year,
  'certificate_file' => $request->certificate_file-
>storeAs('certificate_file', $fileName,'public'),]);  return redirect('/profildi
ri')->with(['success' => 'Data Berhasil Disimpan']); }
  else{
  echo "<script>alert('ekstensi file salah')</script>";
  $biodatapegawai = User::pluck('real_name','nip_nik');
  return view('traininguser.create', compact('biodatapegawai'));;} }
```

```
  public function show($training_id){
  $trainings = Training::find($training_id);
  return view('traininguser.show', compact('trainings')); }
  public function edit($id) {
  $trainings = Training::find($id);
  $biodatapegawai = User::find($trainings->nip_nik);
  $nip_nik = $biodatapegawai->nip_nik;
  return view('traininguser.edit', compact('trainings','biodatapegawai','nip_nik
'));}
  public function update(Request $request, $id)   {
  $extension = $request->certificate_file->extension();
  if ($extension == "pdf"){
  $trainings = Training::find($id);
    $trainings->update([
    'nip_nik' => $request->nip_nik,
    'training_name' => $request->training_name,
    'type_of_training' => $request->type_of_training,
    'place' => $request->place,
    'hour' => $request->hour,
    'year' => $request->year,
    'certificate_file' => $request->certificate_file,]); }else{
  echo "<script>alert('ekstensi file salah')</script>";
  $trainings = Training::find($id);
  $biodatapegawai = User::find($trainings->nip_nik);
  $nip_nik = $biodatapegawai->nip_nik;
  return view('traininguser.edit', compact('trainings','biodatapegawai','nip_nik
'));}
   return redirect()->route('profildiri.index')-
>with(['success' => 'Data Berhasil Disimpan']);}
    public function destroy($id){
    $trainings = training::find($id);
    $trainings->delete();
 return redirect()->route('profildiri.index')-
>with(['error' => 'Data berhasil dihapus']);}}
```

14. StructuralUserController.php

```
class StructuralUserController extends Controller{
  public function index() {
  $strukturaldetails = Structural_Details::all();
  return view('strukturaluser.index', compact('strukturaldetails')); }
  public function create(Request $request) {
  if($request->all()){
  $nipnik = $request->nip_nik;
  }else{
  $nipnik = Auth::user()->nip_nik;   }
  $biodatapegawai = User::find($nipnik);
  $struktural = Structural::pluck('information','structural_id');
  return view('strukturaluser.create', compact('biodatapegawai', 'struktural'))}
    public function store(Request $request){
  $check = Structural_Details::where('structural_id', '=', $request-
>input('structural_id'))->where( 'nip_nik', '=', $request->input('nip_nik'))-
>first(); if($check){
  return redirect()->back()->with(['warning' => 'Data Sudah Ada']);}else{
  $extension = $request->sk_file->extension();
  if ($extension == "pdf"){
  $fileName = $request->file('sk_file')->getClientOriginalName();
  $strukturaldetails= Structural_Details::create([
    'structural_id'=>$request->input('structural_id'),
    'nip_nik'=>$request->input('nip_nik'),
    'tmt'=>$request->input('tmt'),
    'sign_by'=>$request->input('sign_by'),
    'sk_no'=>$request->input('sk_no'),
    'sk_date'=>$request->input('sk_date'),
    'status'=>$request->input('status'),
    'sk_file' => $request->file('sk_file')-
>storeAs('sk_file_struktural', $fileName,'public'),]);
return redirect('/profildiri')->with(['success' => 'Data Berhasil Disimpan']);}
  else{ echo "<script>alert('ekstensi file salah')</script>";
  $biodatapegawai = User::pluck('real_name','nip_nik');
  $struktural = Structural::pluck('information','structural_id');
  return view('strukturaluser.create', compact('biodatapegawai', 'struktural'))}
  public function show($id){
```

```
  $strukturaldetails = Structural_Details::find($id);
  return view('strukturaluser.show', compact('strukturaldetails'));}
  public function edit($structural_id){
  $strukturaldetails = Structural_Details::join('structurals', 'structural_detai
ls.structural_id', '=', 'structurals.structural_id')
  ->select('structural_details.*', 'structurals.*')-
>where('structural_details.structural_id', '=', $structural_id)->first();
    $biodatapegawai = User::find($strukturaldetails->nip_nik);
    $struktural = Structural::pluck('information','structural_id');
    $nip_nik = $biodatapegawai->nip_nik;
return view('strukturaluser.edit', compact('strukturaldetails','biodatapegawai',
'nip_nik', 'struktural'));}
    public function update(Request $request, $structural_id)  {
    $extension = $request->sk_file->extension();
    if ($extension == "pdf"){
    $fileName = $request->file('sk_file')->getClientOriginalName();
    $strukturaldetails = Structural_Details::find($structural_id);
    $strukturaldetails->update([
    'nip_nik'=>$request->input('nip_nik'),
    'tmt'=>$request->input('tmt'),
    'sign_by'=>$request->input('sign_by'),
    'sk_no'=>$request->input('sk_no'),
    'sk_date'=>$request->input('sk_date'),
    'status'=>$request->input('status'),
    'sk_file' => $request->file('sk_file') >storeAs
('sk_file', $fileName,'public'),  ]);  }else{
 echo "<script>alert('ekstensi file salah')</script>";
 $strukturaldetails = Structural_Details::join
('structurals', 'structural_details.structural_id', '=', 'structurals.structural
_id')
   ->select('structural_details.*', 'structurals.*')-
>where('structural_details.structural_id', '=',
$structural_id)->first();
    $biodatapegawai = User::find($strukturaldetails->nip_nik);
    $struktural = Structural::pluck('information','structural_id');
    $nip_nik = $biodatapegawai->nip_nik;
return view('strukturaluser.edit',compact('strukturaldetails',
'biodatapegawai','nip_nik', 'struktural')); }
    return redirect('/profildiri')-
>with(['success' => 'Data Berhasil Disimpan']); }
 public function destroy($id) {
    $strukturaldetails = Structural_Details::find($id);
    $strukturaldetails->delete();
        return redirect()->route('profildiri.index')-
>with(['error' => 'Data Berhasil Dihapus']); }}
```

15. StructuralController.php

```
class StructuralController extends Controller
{
    public function index(){
    $struktural = Structural::all();
    return view('struktural.index', compact('struktural')); }
    public function create()   {
    return view('struktural.create');  }
    public function store(Request $request)   {
    $struktural = structural::create([
      'structural_id'=> $request->input('structural_id'),
      'information'=> $request->input('information'),
      return redirect()->route('struktural.index')->with
(['success' =>'Data Berhasil Disimpan']);   }
    public function show($structural_id){
    $struktural = Structural::find($structural_id);
    return view('struktural.show', compact('struktural'));}
    public function edit($structural_id)   {
    $struktural = Structural::find($structural_id);
    $strukturaldetail = Structural_Details::pluck('nip_nik','tmt','sign_by','sk_
no','sk_date','status','sk_file','structural_id');
    return view('struktural.edit', compact('struktural', 'strukturaldetail')); }
    public function update(Request $request, $id) {
    $struktural = Structural::find($id);
    $struktural->update([
    'information'=> $request->input('information'),]);
```

```
    return redirect()->route('struktural.index')-
>with(['success' => 'Data Berhasil Disimpan']); }
    public function destroy($structural_id)    {
        $struktural = Structural::find($structural_id);
        $struktural->delete();
        return redirect()->route('struktural.index')->with(['error' =>
'Data Berhasil Dihapus']); }}
```

16. RankGroupUserController.php

```
class RankGroupUserController extends Controller
{
  public function index(){
  $pangkatgolongan = Rank_Group::all();
  return view ('pangkatgolonganuser.index', compact('pangkatgolongan'));}
  public function create(request $request) {
  if($request->all()){
  $nipnik = $request->nip_nik;}else{
  $nipnik = Auth::user()->nip_nik; }
  $biodatapegawai = User::find($nipnik);
  $pangkatgolongan = Rank_Group::all();
  return view('pangkatgolonganuser.create', compact('biodatapegawai','pangkatgol
ongan')); }
  public function store(Request $request){
  $extension = $request->sk_file->extension();
  if ($extension == "pdf"){
  $fileName = $request->file('sk_file')->getClientOriginalName();
  $pangkatgolongan = Rank_Group::create([
    'nip_nik' =>$request->nip_nik,
    'name'=>$request->name,
    'tmt' => $request->tmt,
    'sk_no' => $request->sk_no,
     'sk_date'=>$request->sk_date,
     'sign_by'=>$request->sign_by,
     'status'=>$request->status,
     'sk_file' => $request->file('sk_file')-
>storeAs('sk_file_rankgroup', $fileName,'public'), ]);
    return redirect('/profildiri')-
>with(['success' => 'Data Berhasil Disimpan']);} else{
     echo "<script>alert('ekstensi file salah')</script>";
     $biodatapegawai = User::pluck('real_name','nip_nik');
     $pangkatgolongan = Rank_Group::all();
     return view('pangkatgolonganuser.create', compact('biodatapegawai','pangkat
golongan'));}}
    public function show($id)    {
    $pangkatgolongan = Rank_Group::find($id);
    return view('pangkatgolonganuser.show', compact('pangkatgolongan'));  }
    public function edit($id) {
    $pangkatgolongan = Rank_Group::find($id);
    $biodatapegawai = User::find($pangkatgolongan->nip_nik);
    return view('pangkatgolonganuser.edit', compact('pangkatgolongan', 'biodatap
egawai'));}
    public function update(Request $request,$id){
    $extension = $request->sk_file->extension();
      if ($extension == "pdf"){
      $fileName = $request->sk_file->getClientOriginalName();
      $pangkatgolongan = Rank_Group::find($id);
      $pangkatgolongan->update([
     'tmt' => $request->input('tmt'),
     'sk_no' => $request->input('sk_no'),
     'sk_date'=>$request->input('sk_date'),
     'decided_by'=>$request->input('decided_by'),
     'basic_rules'=>$request->input('basic_rules'),
     'sk_file' => $request->sk_file->storeAs('sk_file', $fileName,'public'), ]);
    }else{
     echo "<script>alert('ekstensi file salah')</script>";
     $pangkatgolongan = Rank_Group::find($id);
     $biodatapegawai = User::find($pangkatgolongan->nip_nik);
     return view('pangkatgolonganuser.edit', compact('pangkatgolongan', 'biodata
pegawai'));}
    return redirect()->route('profildiri.index')->with(['success'
=>'Data Berhasil Disimpan']);  }
    public function destroy($rank_group_id){
```

```
        $pangkatgolongan = Rank_Group::find($rank_group_id);
        $pangkatgolongan->delete();
        return redirect()->route('profildiri.index')-
>with(['error' => 'Data Berhasil Dihapus']);} }
```

17. RankGroupController.php

```
class RankGroupController extends Controller
{
    public function index()   {
    $pangkatgolongan = Rank_Group::all();
    return view ('pangkatgolongan.index', compact('pangkatgolongan'))}
    public function create(request $request){
      $nipnik = $request->nip_nik;
      $biodatapegawai = User::find($nipnik);
      $pangkatgolongan = Rank_Group::all();
      return view('pangkatgolongan.create', compact('biodatapegawai','pangkatgol
ongan'));   }
    public function store(Request $request) {
      $pangkatgolongan = Rank_Group::create([
      'nip_nik' =>$request->input('nip_nik'),
      'name'=>$request->input('name'),
      'tmt' => $request->input('tmt'),
      'sk_no' => $request->input('sk_no'),
      'sk_date'=>$request->input('sk_date'),
      'sign_by'=>$request->input('sign_by'),
      'status'=>$request->input('status'),
      'sk_file' => $request->input('sk_file'),
      return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request->input('nip_nik')])-
>with(['success' => 'Data Berhasil Disimpan']);}
    public function show($rank_group_id){
    $pangkatgolongan = Rank_Group::find($rank_group_id);
    return view('pangkatgolongan.show', compact('pangkatgolongan'));   }
    public function edit($rank_group_id)    {
      $pangkatgolongan = Rank_Group::find($rank_group_id);
      $biodatapegawai = User::find($pangkatgolongan->nip_nik);
      return view('pangkatgolongan.edit', compact('pangkatgolongan',
'biodatapegawai')); }
    public function update(Request $request,$rank_group_id){
      $fileName = $request->file('sk_file')->getClientOriginalName();
      $pangkatgolongan = Rank_Group::find($rank_group_id);
      $pangkatgolongan->update([
        'nip_nik' =>$request->input('nip_nik'),
        'name'=>$request->input('name'),
        'tmt' => $request->input('tmt'),
        'sk_no' => $request->input('sk_no'),
        'sk_date'=>$request->input('sk_date'),
        'sign_by'=>$request->input('sign_by'),
        'status'=>$request->input('status'),
        'sk_file' => $request->file('sk_file')-
>storeAs('sk_file_rankgroup', $fileName,'public'), ]);
        return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request->input('nip_nik')])-
>with(['success' => 'Data Berhasil Disimpan']); }
    public function destroy($id)  {
    $pangkatgolongan = Rank_Group::find($id);
    $pangkatgolongan->delete();
    return redirect()->back()->with(['success' => 'Data Berhasil Dihapus']);}}
```

18. PegawaiAPIController.php

```
<?php
namespace App\Http\Controllers;
use App\Role;
use App\User;
use App\Regency;
use App\Structural_Details;
use App\Functional_Details;
use App\Education_Details;
use App\Employee_transfer;
use App\Rank_group;
use Illuminate\Http\Request;
```

```
class PegawaiAPIController extends Controller
{
   public function index(Request $request){
      $biodataPegawai = User::with([
      'address_details.district.regency',
      'education_details',
      'education_details.education',
      'functional_details',
      'functional_details.functional',
      'structural_details',
      'structural_details.structural',
      'employee_transfer',
      'employee_transfer.work_unit',
      'training',
      'rank_group',
      'family'])
->when($request->has('structural'), function ($q) use ($request){
     $query=Structural_Details::select('nip_nik')-
>where('structural_id', $request['structural']);
     return $q->wherein('nip_nik', $query);})
->when($request->has('functional'), function ($q) use ($request){
     $query=Functional_Details::select('nip_nik')-
>where('functional_id', $request['functional']);
     return $q->wherein('nip_nik', $query);})
->when($request->has('education'), function ($q) use ($request){
     $query=Education_Details::select('nip_nik')-
>where('education_id', $request['education']);
     return $q->wherein('nip_nik', $query);})
->when($request->has('work_unit'), function ($q) use ($request){
      $query=Employee_transfer::select('nip_nik')-
>where('work_unit_id', $request['work_unit']);
      return $q->wherein('nip_nik', $query); })
->when($request->has('rank_group'), function ($q) use ($request){
      $query=Rank_group::select('nip_nik')-
>where('rank_group_id', $request['rank_group']);
      return $q->wherein('nip_nik', $query); })
->when($request->has('user'), function ($q) use ($request){
      $query=User::select('nip_nik')->where('nip_nik', $request['user']);
      return $q->wherein('nip_nik', $query);})
->get();
      $biodataFilter = $biodataPegawai->except('role_id');
      return $biodataFilter;}
      public function store(Request $request){}
    public function show($id){}
    public function update(Request $request, $id){}
    public function destroy($id){}}
```

19. FunctionalUserController.php

```
class FunctionalUserController extends Controller {
public function index() {
$fungsionaldetails = Functional_Details::all();
return view('fungsionaluser.index', compact('fungsionaldetails'));}
public function create(Request $request){
if($request->all()){
 $nipnik = $request->nip_nik; }else{
 $nipnik = Auth::user()->nip_nik;}
 $biodatapegawai = User::find($nipnik);
 $fungsionaldetail = Functional_Details::pluck('functional_id');
 $fungsional = Functional::pluck('information','functional_id');
 return view('fungsionaluser.create',compact('nipnik',
'biodatapegawai','fungsional','fungsionaldetail'));}
    public function store(Request $request){
    $check = Functional_Details::where('nip_nik', '=', $request-
>input('nip_nik'))->where('functional_id', '=', $request-
>input('functional_id'))->first(); if($check){
    return redirect()->back()->with(['warning' => 'Data Sudah Ada']);}else{
    $extension = $request->sk_file->extension();
    if ($extension == "pdf"){
    $fileNamee = $request->sk_file->getClientOriginalName();
    $fungsionaldetails= Functional_Details::create([
     'functional_id'=>$request->input('functional_id'),
                'nip_nik'=>$request->input('nip_nik'),
```

```
                'tmt'=>$request->input('tmt'),
                'sign_by'=>$request->input('sign_by'),
                'sk_no'=>$request->input('sk_no'),
                'sk_date'=>$request->input('sk_date'),
                'status'=>$request->input('status'),
                'sk_file' => $request->sk_file-
>storeAs('sk_file_fungsional', $fileNamee,'public'), ]);
return redirect('/profildiri')->with(['success' => 'Data Berhasil Disimpan']);}
 else{  echo "<script>alert('ekstensi file salah')</script>";
        $biodatapegawai = User::pluck('real_name','nip_nik');
        $fungsionaldetails = Functional_Details::pluck('functional_id');
        $fungsional = Functional::pluck('information','functional_id');
 return view('fungsionaluser.create', compact('biodatapegawai','fungsional','fun
gsionaldetails'));}}}
   public function show($functional_id){
   $fungsionaldetails= Functional_Details::find($functional_id);
   return view('fungsionaluser.show', compact('fungsionaldetails'));}
   public function edit($functional_id)   {
   $fungsionaldetail = Functional_Details::find($functional_id);
   $biodatapegawai = User::find($fungsionaldetail->nip_nik);
   $fungsional = Functional::pluck('information','functional_id');
   $nip_nik = $biodatapegawai->nip_nik;
   return view('fungsionaluser.edit', compact('fungsionaldetail','biodatapegawai
','nip_nik', 'fungsional')); }
   public function update(Request $request, $functional_id)    {
      $extension = $request->sk_file->extension();
      if ($extension == "pdf"){
      $fileName = $request->sk_file->getClientOriginalName();
      $fungsionaldetails = Functional_Details::find($functional_id);
      $fungsionaldetails->update([
      'functional_id'=>$request->functional_id,
      'nip_nik'=>$request->nip_nik,
      'tmt'=>$request->tmt,
      'sign_by'=>$request->sign_by,
      'sk_no'=>$request->sk_no,
      'sk_date'=>$request->sk_date,
      'status'=>$request->status,
      'sk_file' => $request->sk_file->storeAs('sk_file', $fileName,'public'),]);
    }else{
        echo "<script>alert('ekstensi file salah')</script>";
        $fungsionaldetail = Functional_Details::find($functional_id);
        $biodatapegawai = User::find($fungsionaldetail->nip_nik);
        $fungsional = Functional::pluck('information','functional_id');
        $nip_nik = $biodatapegawai->nip_nik;
   return view('fungsionaluser.edit', compact('fungsionaldetail','biodatapegawai
','nip_nik', 'fungsional'));}
return redirect()->route('profildiri.index')-
>with(['success' => 'Data Berhasil Disimpan']); }
   public function destroy($id) {
   $fungsionaldetails = Functional_Details::find($id);
   $fungsionaldetails->delete();  return redirect()->route('profildiri.index')-
>with(['error' => 'Data Berhasil Dihapus']);}}
```

20. FunctionalDetailsController.php

```
class FunctionalDetailsController extends Controller
{
 public function index()   {
 $fungsionaldetail = Functional_Details::all();
 return view('fungsionaldetail.index', compact('fungsionaldetail'));}
 public function create(Request $request) {
    $nipnik = $request->nip_nik;
    $biodatapegawai = User::find($nipnik);
    $fungsionaldetail = Functional_Details::pluck('functional_id');
    $fungsional = Functional::pluck('information','functional_id');
   return view('fungsionaldetail.create', compact('nipnik','biodatapegawai','fun
gsional','fungsionaldetail')); }
 public function store(Request $request){
       $fileNamee = $request->sk_file->getClientOriginalName();
       $fungsionaldetail= Functional_Details::create([
         'functional_id'=>$request->input('functional_id'),
         'nip_nik'=>$request->input('nip_nik'),
         'tmt'=>$request->input('tmt'),
```

```
          'sign_by'=>$request->input('sign_by'),
          'sk_no'=>$request->input('sk_no'),
          'sk_date'=>$request->input('sk_date'),
          'status'=>$request->input('status'),
          'sk_file' => $request->sk_file-
>storeAs('sk_file', $fileNamee,'public'),  ]);
      return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request->input('nip_nik')])-
>with(['success' => 'Data Berhasil Disimpan']);}
   public function show($functional_id)     {
      $fungsionaldetail = Functional_Details::find($functional_id);
      return view('fungsionaldetail.show', compact('fungsionaldetail'));  }
   public function edit($functional_id)   {
      $fungsionaldetail = Functional_Details::join('functionals', 'functional_de
tails.functional_id', '=', 'functionals.functional_id')
   ->select('functional_details.*', 'functionals.*')-
>where('functional_details.functional_id', '=', $functional_id)->first();
      $biodatapegawai = User::find($fungsionaldetail->nip_nik);
        $fungsionaldetail = Functional_Details::find($functional_id);
        $fungsional = Functional::pluck('information','functional_id');
         return view('fungsionaldetail.edit', compact('biodatapegawai','fungsion
al','fungsionaldetail'));}
    public function update(Request $request, $functional_id)   {
      $fileName = $request->sk_file->getClientOriginalName();
      $fungsionaldetail = Functional_Details::find($functional_id);
      $fungsionaldetail->update([
         'functional_id'=>$request->functional_id,
         'nip_nik'=>$request->nip_nik,
         'tmt'=>$request->tmt,
         'sign_by'=>$request->sign_by,
         'sk_no'=>$request->sk_no,
         'sk_date'=>$request->sk_date,
         'status'=>$request->status,
         'sk_file' => $request->sk_file-
>storeAs('sk_file_fungsional', $fileName,'public'), ]);
      return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request->input('nip_nik')])-
>with(['success' => 'Data Berhasil Disimpan']);}
    public function destroy($id){
      $fungsionaldetail = Functional_Details::find($id);
      $fungsionaldetail->delete();
      return redirect()->back()->with(['error' => 'Data Berhasil Dihapus']); }}
```

21. FunctionalController.php

```
<?php
namespace App\Http\Controllers;
use App\Functional;
use App\Functional_Details;
use Illuminate\Http\Request;
use GuzzleHttp\Client;
class FunctionalController extends Controller{
    public function index(){
    $url = "http://localhost/ta/public/api/fungsional?api-
key=xddHIyF6x21VyfTO4pwaP3ArUqFiGfoQRrDE64hv";
    try {
  $client = new Client();
  $res = $client->request('GET',$url);
  $json = $res->getBody();
  $fungsional = json_decode($json, true);
  $fungsional = collect($fungsional)->map(function ($s){
  return (object) $s;     });
  }catch(Exception $e){
  dd($e->getMessage());  }
  return view('fungsional.index', compact('fungsional'));}
  public function create()   {
  return view('fungsional.create');   }
    public function store(Request $request){
      $fungsional = Functional::create([
      'functional_id'=> $request->input('functional_id'),
      'information'=> $request->input('information'),     ]);
      return redirect()->route('fungsional.index')-
>with(['success' => 'Data Berhasil Disimpan']);}
```

```
    public function show($functional_id)    {
        $fungsional = Functional::find($functional_id);
        return view('fungsional.show', compact('fungsional'));}
    public function edit( $functional_id)    {
        $fungsional = Functional::find($functional_id);
        $fungsionaldetail = Functional_Details::pluck('nip_nik','tmt','sign_by',
'sk_no','sk_date','status','sk_file','functional_id');
return view('fungsional.edit', compact('fungsional','fungsionaldetail')); }
public function update(Request $request, $id){
        $fungsional = Functional::find($id);
        $fungsional->update([
        'nip_nik'=> $request->nip_nik,
        'information' => $request->information,
        'tmt' => $request->tmt,
        'sign_by'=> $request->sign_by,
        'sk_no' => $request->sk_no,
        'sk_date'=> $request->sk_date,
        'status' => $request->status,
        'sk_file' => $request->sk_file,   ]);
        return redirect()->route('fungsional.index')-
>with(['success' => 'Data Berhasil Disimpan']); }
    public function destroy( $functional_id)    {
        $fungsional = Functional::find($functional_id);
        $fungsional->delete();
        return redirect()->route('fungsional.index')-
>with(['error' => 'Data Berhasil Dihapus']);}}
```

22. FamilyUserController.php

```
class FamilyUserController extends Controller
{
   public function index(){
   $biodatakeluarga = Family::get();
   return view ('biodatakeluargauser._form', compact('biodatakeluarga'));}
   public function create(request $request) {
      if($request->all()){
      $nipnik = $request->nip_nik;
      }else{
      $nipnik = Auth::user()->nip_nik; }
   $biodatapegawai = User::find($nipnik);
   $biodatakeluarga = Family::all();
   $education = Education::pluck('level','education_id');
   return view('biodatakeluargauser.create', compact('biodatapegawai','biodatake
luarga','education')); }
   public function store(Request $request) {
   $do = Family::create([
      'id_number' => $request->id_number,
      'nip_nik' => $request->nip_nik,
      'name' => $request->name,
      'relationship'=>$request->relationship,
      'phone_number'=>$request->phone_number,
      'birth_date'=>$request->birth_date,
      'birth_place'=>$request->birth_place,
      'occupation'=>$request->occupation,
      'last_education'=>$request->last_education,
      'npwp_no'=>$request->npwp_no, ])->id_number;
 Family::where('id_number', '=', $do)->update([
      'id_number' => $request->id_number, ]);
return redirect('/profildiri') ->with(['success' => 'Data Berhasil Disimpan']);}
public function show($id_number){
      $biodatakeluarga = Family::join('education', 'family.last_education', '=',
 'education.education_id')
->where('family.id_number','=',$id_number)->first();
return view('biodatakeluargauser.show', compact('biodatakeluarga')); }
public function edit($id_number) {
    $biodatakeluarga = Family::find($id_number);
    $biodatapegawai = User::find($biodatakeluarga->nip_nik);
    $education = Education::pluck('level','education_id');
    return view('biodatakeluargauser.edit', compact('education','biodatapegawai'
,'biodatakeluarga')); }
 public function update(Request $request, $id){
    $biodatakeluarga = Family::find($id);
    $biodatakeluarga->update([
```

```
        'id_number' => $request->id_number,
        'nip_nik' => $request->nip_nik,
        'name' => $request->name,
        'relationship'=>$request->relationship,
        'phone_number'=>$request->phone_number,
        'birth_date'=>$request->birth_date,
        'birth_place'=>$request->birth_place,
        'occupation'=>$request->occupation,
        'last_education'=>$request->last_education,
         'npwp_no'=>$request->npwp_no,]);
return redirect()->route('profildiri.index') -
>with(['success' => 'Data Berhasil Disimpan']);}

    public function destroy($id) {
        $biodatakeluarga = Family::find($id);
        $biodatakeluarga->delete();
        return redirect()->route('profildiri.index') -
>with(['error' => 'Data Berhasil Dihapus']);  } }
```

23. FamilyController.php

```
class FamilyController extends Controller
{
   public function index(){
   $biodatakeluarga = Family::get();
   return view ('biodatakeluarga._form', compact('biodatakeluarga'));  }
   public function create(request $request)   {
        $nipnik = $request->nip_nik;
        $biodatapegawai = User::find($nipnik);
        $biodatakeluarga = Family::all();
        $education = Education::pluck('level','education_id');
        return view('biodatakeluarga.create', compact('biodatapegawai','biodatak
eluarga','education')); }
   public function store(Request $request){
        $do = Family::create([ 'id_number' => $request->id_number,
        'nip_nik' => $request->nip_nik, 'name' => $request->name,
        'relationship'=>$request->relationship,
        'phone_number'=>$request->phone_number,
        'birth_date'=>$request->birth_date,'birth_place'=>$request->birth_place,
        'occupation'=>$request->occupation,
        'last_education'=>$request->last_education,
        'npwp_no'=>$request->npwp_no,   ])
   ->id_number;
 Family::where('id_number', '=', $do)->update([
      'id_number' => $request->id_number,   ]);
    return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request-
>input('nip_nik')]) ->with(['success' => 'Data Berhasil Disimpan']);}
    public function show($id_number)   {
        $biodatakeluarga = Family::find($id_number);
        return view('biodatakeluarga.show', compact('biodatakeluarga')); }
    public function edit($id_number)    {
        $biodatakeluarga = Family::find($id_number);
        $biodatapegawai = User::find($biodatakeluarga->nip_nik);
        $education = Education::pluck('level','education_id');
        return view('biodatakeluarga.edit', compact('biodatakeluarga','biodatape
gawai',"education"));}
    public function update(Request $request, $id_number){
        $biodatakeluarga = Family::find($id_number);
        $biodatakeluarga->update([
            'nip/nik' => $request->nip_nik, 'name' => $request->name,
            'relationship'=>$request->relationship,
            'phone_no'=>$request->phone_no, 'birth_date'=>$request->birth_date,
            'birth_place'=>$request->birth_place,
            'occupation'=>$request->occupation,
            'last_education'=>$request->last_education,
            'npwp_no'=>$request->npwp_no,]);
        return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request-
>input('nip_nik')]) ->with(['success' => 'Data Berhasil Disimpan']);}
    public function destroy($id_number){
      $biodatakeluarga = Family::find($id_number);
      $biodatakeluarga->delete();
```

```
      return redirect()->back()->with(['error' => 'Data Berhasil Dihapus']);  }}
```

24. EmployeeTransferController.php

```
class EmployeeTransferController extends Controller
{ public function index(){
     $employee_transfer = Employee_Transfer::get();
     $work_unit = Work_Unit::get();
     $employee_transfer = Employee_Transfer::all();
     return view('mutasi.index', compact('employee_transfer','work_unit'));}
  public function create(Request $request){
     $nipnik = $request->nip_nik;
     $biodatapegawai = User::find($nipnik);
     $work_unit = Work_Unit::pluck('name','work_unit_id');
     return view('mutasi.create', compact('biodatapegawai','work_unit'));}
  public function store(Request $request) {
     $fileNamee = $request->sk_file->getClientOriginalName();
     Employee_Transfer::create([
            'employee_transfer_id' => $request->employee_transfer_id,
            'work_unit_id' => $request->work_unit_id,
            'nip_nik' => $request->nip_nik,
            'employee_transfer_date' => $request->employee_transfer_date,
            'sk_no' => $request->sk_no,
            'sign_by' => $request->sign_by,
            'sk_file' => $request->sk_file-
>storeAs('sk_file_mutasi', $fileNamee,'public'),]);
        return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request-
>input('nip_nik')]) ->with(['success' => 'Data Berhasil Disimpan']); }
    public function show($employee_transfer_id)   {
        $work_unit = Work_Unit::get();
        $employee_transfer = Employee_Transfer::find($employee_transfer_id);
        return view('mutasi.show', compact('employee_transfer','work_unit')); }
    public function edit( $employee_transfer_id) {
        $employee_transfer = Employee_Transfer::join('work_units', 'employee_tra
nsfers.work_unit_id', '=', 'work_units.work_unit_id')
        ->select('employee_transfers.*', 'work_units.*')-
>where('employee_transfers.employee_transfer_id', '=', $employee_transfer_id)-
>first();
        $biodatapegawai = User::find($employee_transfer->nip_nik);
        $employee_transfer = Employee_Transfer::find($employee_transfer_id);
        $work_unit = Work_Unit::pluck('name','work_unit_id');
return view('mutasi.edit', compact('employee_transfer','work_unit','biodatapegaw
ai'));  }
    public function update( Request $request, $employee_transfer_id){
        $fileNamee = $request->sk_file->getClientOriginalName();
        $employee_transfer= Employee_Transfer::find($employee_transfer_id);
        $employee_transfer->update([
           'work_unit_id' => $request->work_unit_id,
           'nip_nik' => $request->nip_nik,
           'employee_transfer_date' => $request->employee_transfer_date,
           'sk_no' => $request->sk_no,
           'sign_by' => $request->sign_by,
           'sk_file' => $request->sk_file-
>storeAs('sk_file_mutasi', $fileNamee,'public'),]);
        return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request->input('nip_nik')])-
>with(['success' => 'Data Berhasil Disimpan']);   }
    public function destroy($employee_transfer_id)   {
        $employee_transfer = Employee_Transfer::find($employee_transfer_id);
        $employee_transfer->delete();
        return redirect()->back()->with(['error' => 'Data Berhasil Dihapus']);}}
```

25. EducationDetailsController.php

```
class EducationDetailsController extends Controller {
public function index()   {
      $educationdetail = Education_Details::all();
      return view('educationdetail.index', compact('educationdetail')); }
public function create(Request $request){
      $nipnik = $request->nip_nik;
      $biodatapegawai = User::find($nipnik);
      $education = Education::pluck('level','education_id');
```

```
      $education_id = null;''
 return view('educationdetail.create', compact('nipnik','education','biodatapega
wai', 'education_id')); }
    public function store(Request $request){
        $fileName = $request->certificate_file->getClientOriginalName();
        $educationdetail= Education_Details::create([
            'education_details_id'=>$request->input('education_details_id'),
            'education_id'=>$request->input('educationselect'),
            'nip_nik'=>$request->input('nip_nik'),
            'name'=>$request->input('name'),
            'major'=>$request->input('major'),
            'graduation_year'=>$request->input('graduation_year'),
            'country'=>$request->input('country'),
            'dean_headmaster'=>$request->input('dean_headmaster'),
            'certificate_file' => $request->certificate_file-
>storeAs('certificate_file', $fileName,'public'), ]);
 return redirect()->route('biodatapegawai.show', ['biodatapegawai' => $request-
>input('nip_nik')]) ->with(['success' => 'Data Berhasil Disimpan']); }
    public function show($id) {
        $educationdetail = Education_Details::find($id);
        return view('educationdetail.show', compact('educationdetail')); }
    public function edit($education_details_id)    {
        $educationdetail = Education_Details::join('education', 'education_detai
ls.education_id', '=', 'education.education_id')
        ->select('education_details.*', 'education.*')-
>where('education_details.education_details_id', '=', $education_details_id)-
>first();
        $biodatapegawai = User::find($educationdetail->nip_nik);
        $education = Education::pluck('level','education_id');
        $education_id = $educationdetail->education_id;
        return view('educationdetail.edit', compact('educationdetail','biodatape
gawai','education','education_id'));  }
    public function update(Request $request,$id)    {
        $fileName = $request->certificate_file->getClientOriginalName();
        $educationdetail = Education_Details::find($id);
        $educationdetail->update([
            'education_id'=>$request->input('educationselect'),
            'name'=>$request->input('name'),
            'major'=>$request->input('major'),
            'graduation_year'=>$request->input('graduation_year'),
            'country'=>$request->input('country'),
            'dean_headmaster'=>$request->input('dean_headmaster'),
            'certificate_file' => $request->certificate_file-
>storeAs('certificate_file', $fileName,'public'), ]);
        return redirect()-
>route('biodatapegawai.show', ['biodatapegawai' => $request->input('nip_nik')])-
>with(['success' => 'Data Berhasil Disimpan']);}
    public function destroy($id) {
        $educationdetail = Education_Details::find($id);
        $educationdetail->delete();
        return redirect()->back()->with(['error' => 'Data Berhasil Dihapus']);}}
```

26. BiodatapribadiController.php

```
class BiodatapribadiController extends Controller{
 public function index(){
        $biodatapegawai = User::get();
        $role = Role::get();
        $regencies = Regency::all();
        return view ('biodatapegawai.index', compact('biodatapegawai','regencies
','role'));   }
    public function indexlist(){
        $biodatapegawai = User::all();
        $role = Role::get();
        $regencies = Regency::all();
        return view ('biodatapegawai.index', compact('biodatapegawai','regencies
','role'));}
    public function get_kecamatan(Request $request){
        $kecamatan = District::where('regency_id',$request->kabupaten_kota)-
>get();
        return response()->json([
```

```
            "kecamatan" => $kecamatan, ]);   }
    public function store(Request $request){
        $fileName = $request->upload_gambar->getClientOriginalName();
        $user= User::create([
            'nip_nik'=> $request->nip_nik,
            'nidn'=>$request->nidn,
            'title_ahead'=>$request->title_ahead,
            'real_name'=>$request->real_name,
            'back_title'=>$request->back_title,
            'birth_place'=>$request->birth_place,
            'birth_date'=>$request->birth_date,
            'blood_group'=>$request->blood_group,
            'height'=>$request->height,
            'weight'=>$request->weight,
            'phone_number'=>$request->phone_number,
            'email'=>$request->email,
            'id_card_number'=>$request->id_card_number,
            'npwp'=>$request->npwp,
            'role_id'=>$request->roleselect,
            'username'=>$request->nip_nik,
            'password'=>bcrypt($request->nip_nik),
            'bpjs'=>$request->bpjs,
            'gender'=>$request->gender,
            'religion'=>$request->religion,
            'marital_status'=>$request->marital_status,
            'citizen_status'=>$request->citizen_status,
            'retirement_age_limit'=>$request->retirement_age_limit,
            'employee_status'=>$request->employee_status,
            'photo' => $request->upload_gambar-
>storeAs('photo', $fileName,'public'),]);
        $hitung_detail = Address_Details::count('address_details_id');
        if($hitung_detail == 0){
            $count = 1;}
        else{
            $count = $hitung_detail + 1;}
        $address_detail= Address_Details::create([
            'address_details_id'=> "ADDRESS-".$count,
            'nip_nik'=> $request->nip_nik,
            'district_id'=> $request->districts,
            'address'=>$request->address, ]);
        return redirect()->route("biodatapegawai.index")-
>with(['success' => 'Data Berhasil Disimpan']); }
    public function edit($nip_nik){
        $biodatapegawai=User::join('address_details', 'users.nip_nik', '=', 'add
ress_details.nip_nik')
        -
>join('districts', 'address_details.district_id', '=', 'districts.district_id')
        ->join('regencies', 'districts.regency_id', '=', 'regencies.regency_id')
        ->join('roles', 'users.role_id', '=', 'roles.id') -
>select('users.*', 'address_details.address', 'districts.district_id', 'regencie
s.regency_id', 'roles.*')
        ->where('users.nip_nik', '=', $nip_nik)
        ->first();
        $regencies = Regency::pluck('regency_name','regency_id');
        $districts = District::pluck('district_name','district_id');
        $role = Role::pluck('nama_role','id');
        $role_id = $biodatapegawai->role_id;
        $district_id = $biodatapegawai->district_id;
        $regency_id = $biodatapegawai->regency_id;
        return view('biodatapegawai.edit', compact('biodatapegawai','regencies',
 'districts','role', 'district_id', 'regency_id', 'role_id'));}
    public function update(Request $request) {
        if($request->upload_gambar){
        $fileName = $request->upload_gambar->getClientOriginalName();
```

```
        $biodatapegawai= User::where("nip_nik",$request->nip_nik)->update([
            'nidn'=>$request->input('nidn'),
            'title_ahead'=>$request->input('title_ahead'),
            'real_name'=>$request->input('real_name'),
            'back_title'=>$request->input('back_title'),
            'birth_place'=>$request->input('birth_place'),
            'birth_date'=>$request->input('birth_date'),
            'blood_group'=>$request->input('blood_group'),
            'height'=>$request->input('height'),
            'weight'=>$request->input('weight'),
            'phone_number'=>$request->input('phone_number'),
            'email'=>$request->input('email'),
            'id_card_number'=>$request->input('id_card_number'),
            'npwp'=>$request->input('npwp'),
            'role_id'=>$request->input('roleselect'),
            'username'=>$request->input('nip_nik'),
            'bpjs'=>$request->input('bpjs'),
            'gender'=>$request->input('gender'),
            'religion'=>$request->input('religion'),
            'marital_status'=>$request->input('marital_status'),
            'citizen_status'=>$request->input('citizen_status'),
            'retirement_age_limit'=>$request->input('retirement_age_limit'),
            'employee_status'=>$request->input('employee_status'),
            'photo' => $request->upload_gambar-
>storeAs('photo', $fileName,'public'),]);
        }else{
            $biodatapegawai= User::where("nip_nik",$request->nip_nik)->update([
                'nidn'=>$request->input('nidn'),
                'title_ahead'=>$request->input('title_ahead'),
                'real_name'=>$request->input('real_name'),
                'back_title'=>$request->input('back_title'),
                'birth_place'=>$request->input('birth_place'),
                'birth_date'=>$request->input('birth_date'),
                'blood_group'=>$request->input('blood_group'),
                'height'=>$request->input('height'),
                'weight'=>$request->input('weight'),
                'phone_number'=>$request->input('phone_number'),
                'email'=>$request->input('email'),
                'id_card_number'=>$request->input('id_card_number'),
                'npwp'=>$request->input('npwp'),
                'role_id'=>$request->input('roleselect'),
                'username'=>$request->input('nip_nik'),
                'bpjs'=>$request->input('bpjs'),
                'gender'=>$request->input('gender'),
                'religion'=>$request->input('religion'),
                'marital_status'=>$request->input('marital_status'),
                'citizen_status'=>$request->input('citizen_status'),
                'retirement_age_limit'=>$request->input('retirement_age_limit'),
                'employee_status'=>$request->input('employee_status'),]);}
         $address= Address_Details::where("nip_nik",$request->nip_nik)->update([
                'district_id'=>$request->input('districts'),
                'address'=>$request->input('address'),  ]);
      if(Auth::user()->role_id == 2){
      return redirect()->route('profildiri.index')-
>with(['success' => 'Data Berhasil Disimpan']);
            }else{
      return redirect()->route("biodatapegawai.index")-
>with(['success' => 'Data Berhasil Disimpan']);       }}
    public function destroy($nip_nik) {
        $biodatapegawai = User::find($nip_nik);
        $biodatapegawai->delete();
        return redirect()->back() ->with(['error' => 'Data Berhasil Dihapus']);}
    public function create()   {
        $regencies = Regency::pluck('regency_name','regency_id');
```

```
        $districts = District::pluck('district_name','district_id');
        $role = Role::pluck('nama_role', 'id');
        $role_id = null;
        $district_id = null;
        $regency_id = null;
        $biodatapegawai = null;
        return view('biodatapegawai.create', compact('biodatapegawai', 'regencie
s','districts','role', 'district_id', 'regency_id', 'role_id'));}
    public function show($id)   {
        $biodatapegawai = User::find($id);
        $address = $biodatapegawai->address_details->first();
        $regency = $biodatapegawai->address_details->first()->district->regency;
        $district = $biodatapegawai->address_details->first()->district;
        return view('biodatapegawai.show', compact('biodatapegawai', 'address',
'district', 'regency'));    }
    public function gantipass($nip_nik){
        $nip_nik = $nip_nik;
        $biodatapegawai = User::find($nip_nik);
 return view('biodatapegawai.gantipass', compact('biodatapegawai', 'nip_nik'));}
    public function updatepass(Request $request)  {
        $biodatapegawai= User::find($request->nip_nik);
        if(Hash::check($request->old_password, bcrypt($request-
>old_password)) == Hash::check($request->old_password, $biodatapegawai-
>password)){
            $biodatapegawai = User::where("nip_nik",$request->nip_nik)->update([
                'password'=>bcrypt($request->password),        ]);
                if(Auth::user()->role_id == 2){
                    return redirect()->route('profildiri.index')-
>with(['success' => 'Password Berhasil Diubah']);
                }else{
            return redirect()->route('biodatapegawai.show', $request->nip_nik)-
>with(['success' => 'Password Berhasil Diubah']);    } }else{
       return redirect()->back()->with(['error' => 'Password Lama Salah']);} }}
```

27. ApiMasterController.php

```
<?php
namespace App\Http\Controllers;
use App\Education;
use App\Structural;
use Illuminate\Http\Request;
use App\Functional;
use App\Work_Unit;
class ApiMasterController extends Controller {
public function struktural(){
   $data=Structural::get();
   return response(json_encode($data));}
public function fungsional(){
   $data=Functional::get();
   return $data;}
public function education(){
   $data=Education::get();
   return $data;    }
public function workunit()   { $data=Work_Unit::get();
       return $data; }}
```

28. ApiController.php

```
class ApiController extends Controller
{
    public function index(){
        $api = Api::all();
        return view('api.index', compact('api')); }
    public function create()   {
       $token= Str::random(40);
       $api = null;
       return view('api.create', compact('api','token'));   }
    public function store(Request $request)  {
```

```
        $api= Api::create([
        'id' => $request->input('id'),
         'name' => $request->input ('name'),
         'token' => $request->input ('token'),  ]);
    return redirect()->route("api.index")-
>with(['success' => 'Data Berhasil Disimpan']);}
    public function show($id)    {
        $api = Api::find($id);
        return view('api.show', compact('api')); }
    public function edit($id) {
        $api = Api::find($id);
        return view('api.edit', compact('api'));    }
    public function update(Request $request, $id){
        $api = Api::find($id);
        $api->update([
            'name' => $request->input('name'),
            'token' => $request->input('token'),]);
        return redirect()->route('api.index')-
>with(['success' => 'Data Berhasil Disimpan']); }
    public function destroy($id) {
        $api = Api::find($id);
        $api->delete();
        return redirect()->route('api.index')-
>with(['error' => 'Data Berhasil Dihapus']); }}
```

29. Routes api.php

```
<?php
use App\Http\Controllers\ApiMasterController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();});
Route::apiResource('pegawai', 'PegawaiAPIController');
Route::get("struktural", [ApiMasterController::class, 'struktural']);
Route::get("fungsional", [ApiMasterController::class, 'fungsional']);
Route::get("education", [ApiMasterController::class, 'education']);
Route::get("workunit", [ApiMasterController::class, 'workunit']);
```

37.*View* biodatapegawai *show.blade*

```
@extends('layouts.myApp')
@section('content')
  @if ($message = Session::get('success'))
  <div class="alert alert-success alert-block">
  <button type="button" class="close" data-dismiss="alert">×</button>
  <strong>{{ $message }}</strong>
  </div>
  @endif
    @if ($message = Session::get('error'))
    <div class="alert alert-danger alert-block">
    <button type="button" class="close" data-dismiss="alert">×</button>
    <strong>{{ $message }}</strong>
    </div>
  @endif
  @if ($message = Session::get('warning'))
    <div class="alert alert-warning alert-block">
    <button type="button" class="close" data-dismiss="alert">×</button>
    <strong>{{ $message }}</strong>
    </div>
  @endif
  @if ($message = Session::get('info'))
    <div class="alert alert-info alert-block">
    <button type="button" class="close" data-dismiss="alert">×</button>
    <strong>{{ $message }}</strong>
    </div>
  @endif
  @if ($errors->any())
    <div class="alert alert-danger">
    <button type="button" class="close" data-dismiss="alert">×</button>
    Please check the form below for errors
    </div>
    @endif
```

```
   <div class="container-fluid">
{{-- show pegawai --}}
    <div class="fade-in">
      <div class="row">
         <div class="col-lg-12">
            <div class="card">
              <h5>
                <div class="card-header">
                 <div class="d-flex">
                 {{$biodatapegawai->real_name}} </div> </div>
              </h5>
             <div class="card-body">
            <table class="table table-responsive">
           <tr>
          <th>NIP/NIK</th>
         <td>:</td>
        <td>{{ $biodatapegawai->nip_nik }}</td>
       </tr>
      <tr>
     <th>NIDN</th>
    <td>:</td>
   <td>{{ $biodatapegawai->nidn}}</td>
  </tr>
 <tr>
 <th>Nama</th>
 <td>:</td>
@if ($biodatapegawai->title_ahead && $biodatapegawai->back_title)
    <td>{{ $biodatapegawai->title_ahead}}. {{$biodatapegawai-
>real_name }}, {{ $biodatapegawai->back_title }}  </td>
@elseif ($biodatapegawai->title_ahead)<td>{{ $biodatapegawai-
>title_ahead}}. {{$biodatapegawai->real_name }} </td>
@elseif ($biodatapegawai->back_title)<td>{{ $biodatapegawai-
>real_name }}, {{ $biodatapegawai-
>back_title }}  </td>else<td>{{ $biodatapegawai->real_name }}  </td>
@endif </tr> <tr>
   <th>tempat lahir</th>
     <td>:</td> <td>{{ $biodatapegawai->birth_place}}</td> </tr>
      <tr> <th>tanggal lahir</th>
       <td>:</td>
     <td>{{ $biodatapegawai->birth_date}}</td>
    </tr>
   <tr>
  <th>Jenis Kelamin</th>
  <td>:</td>
  <td>{{ $biodatapegawai->gender}}</td>
  </tr> <tr>
    <th>Agama</th>
      <td>:</td>
        <td>{{ $biodatapegawai->religion}}</td>
           </tr>
             <tr>
                <th>Berat Badan</th>
                <td>:</td>
                <td>{{ $biodatapegawai->weight}}</td> </tr>
          <tr>
       <th>Tinggi Badan</th>
    <td>:</td>
    <td>{{ $biodatapegawai->height}}</td> </tr>
  <tr>
    <th>Golongan Darah</th>
       <td>:</td>
         <td>{{ $biodatapegawai->blood_group}}</td> </tr>
  <tr>
      <th>Kewarganegaraan</th>
          <td>:</td>
             <td>{{ $biodatapegawai->citizen_status}}</td> </tr> <tr>
       <th>Alamat</th>
          <td>:</td>
             <td>{{ $address->address}}</td></tr> <tr>
       <th>Kabupaten</th>
       <td>:</td>
       <td>{{ $regency->regency_name}}</td></tr>   <tr>
       <th>Kecamatan</th>
       <td>:</td>
```

```
        <td>{{ $district->district_name}}</td>   </tr><tr>
        <th>Email</th>
        <td>:</td>
        <td>{{ $biodatapegawai->email}}</td></tr>
<tr> <th>No Telp</th>
    <td>:</td>
   <td>{{ $biodatapegawai->phone_number}}</td>  </tr>
        <tr>
      <th>No BPJS</th>
      <td>:</td>
      <td>{{ $biodatapegawai->bpjs}}</td>    </tr>
 <tr> <th>Status Perkawinan</th>
      <td>:</td>
      <td>{{ $biodatapegawai->marital_status}}</td></tr>
  <tr>  <th>Status Pegawai</th>
   <td>:</td>
      <td>{{ $biodatapegawai->employee_status}}</td> </tr>
  <tr> <th>Foto</th>
     <td>:</td> <td><a target="_blank" href="{{ $biodatapegawai-
>avatarurl}}">{{ $biodatapegawai->photo}}</a></td> </tr>
</table>
 <div class="d-flex">
 <a href="{{ route('gantipass', $biodatapegawai->nip_nik) }}" class="btn btn-
primary">Ganti Password</a></div> </div> </div> </div>  </div> </div>
{{-- end show pegawai --}}

{{-- education --}}
<div class="fade-in">
      <div class="row">
        <div class="col-lg-12">
            <div class="card">
                <h5>
             <div class="card-header">
           <h4>Riwayat Pendidikan</h4>
              <div class="d-flex">
        <a href="{{ route('educationdetail.create',  ['nip_nik'=> $biodatapegawa
i->nip_nik]) }}" class="btn btn-primary">Tambah</a> </div> </div></h5>
      <div class="card-body">
         <table class="table table-responsive-sm table-striped">
          <thead>
            <tr>
             <th class="text-center">No.</th>
             <th class="text-center">Tingkat</th>
             <th class="text-center">Nama Sekolah</th>
             <th class="text-center">Tahun Kelulusan</th>
             <th class="text-center">Aksi</th>
             <th class="text-center"></th> </tr> </thead>
         <tbody>
@foreach ($biodatapegawai->education_details as $educationdetails)
  <tr><td class="text-center">{{ $loop->iteration }}</td>
  <td class="text-center">{{ $educationdetails->education->level}}</td>
  <td class="text-center">{{ $educationdetails->name}}</td>
  <td class="text-center">{{ $educationdetails->graduation_year }}</td>
  <td class="text-center">
  <a href="{{ route('educationdetail.show',$educationdetails-
>education_details_id) }}"class="btn btn-info" id="editButton" data-
target="#editPegawai">
  <i class="cil-zoom-in"></i></a>
<a href="{{ route('educationdetail.edit',$educationdetails-
>education_details_id) }}"class="btn btn-warning" id="editButton" data-
target="#editPegawai">
<i class="cil-pencil"></i> </a>
  <form action="{{ route('educationdetail.destroy', $educationdetails-
>education_details_id) }}"
 method="post" onclick="return confirm('Anda yakin menghapus data ?')"
  class="d-inline">
    @csrf
    @method('DELETE')
    <button class="btn btn-youtube"> <i class="cil-trash"></i> </button>
   </form> </td> </tr>
    @endforeach
      </tbody> </table> </div> </div> </div> </div> </div>
{{-- end education --}}
```

```
{{-- jabatan fungsional --}}
    <div class="fade-in">
        <div class="row">
            <div class="col-lg-12">
                <div class="card"> <h5>
                   <div class="card-header">
                      <h4>Riwayat Jabatan Fungsional</h4>
                         <div class="d-flex">
          <a href="{{ route('fungsionaldetail.create', ['nip_nik'=> $biodatapega
wai->nip_nik]) }}" class="btn btn-primary">Tambah</a>
         </div> </div> </h5>
       <div class="card-body">
    <table class="table table-responsive-sm table-striped"> <thead>
  <tr> <th class="text-center">No</th> <th class="text-
center">Jabatan Fungsional</th>
   <th class="text-center">Status</th>
   <th class="text-center">Aksi</th> </tr> </thead> <tbody>
 @foreach ($biodatapegawai->functional_details as $fungsionaldetails)<tr>
  <td class="text-center">{{ $loop->iteration }}</td>
  <td class="text-center">{{ $fungsionaldetails->functional->information}}</td>
  <td class="text-center">{{ $fungsionaldetails->status}}</td>
  <td class="text-center">
     <a href="{{ route('fungsionaldetail.show',$fungsionaldetails-
>functional_id) }}"class="btn btn-info" id="editButton" data-
target="#editPegawai">
   <i class="cil-zoom-in"></i> </a>
   <a href="{{ route('fungsionaldetail.edit',$fungsionaldetails-
>functional_id) }}"class="btn btn-warning" id="editButton" data-
target="#editPegawai">
  <i class="cil-pencil"></i> </a>
  <form action="{{ route('fungsionaldetail.destroy', $fungsionaldetails-
>functional_id) }}"
  method="post" onclick="return confirm('Anda yakin menghapus data ?')"
     class="d-inline">
  @csrf
  @method('DELETE')
  <button class="btn btn-youtube">  <i class="cil-trash"></i> </button> </form>
 </td> </tr>
  @endforeach </tbody> </table> </div> </div>  </div></div> </div>
{{-- end jabatan fungsional --}}

{{-- Struktural --}}
<div class="fade-in">
    <div class="row">
        <div class="col-lg-12">
            <div class="card"> <h5>
               <div class="card-header">
                  <h4>Riwayat Jabatan Struktural</h4>
                    <div class="d-flex">
<a href="{{ route('strukturaldetail.create', ['nip_nik'=> $biodatapegawai-
>nip_nik]) }}" class="btn btn-primary">Tambah</a>
</div> </div> </h5> <div class="card-body">
<table class="table table-responsive-sm table-striped">
<thead>
 <tr>
   <th class="text-center">No</th>
   <th class="text-center">Jabatan Struktural</th>
   <th class="text-center">Status</th>
    <th class="text-center">Aksi</th> </tr></thead> <tbody>
  @foreach ($biodatapegawai->structural_details as $strukturaldetails)<tr>
  <td class="text-center">{{ $loop->iteration }}</td>
  <td class="text-center">{{ $strukturaldetails->structural->information}}</td>
  <td class="text-center">{{ $strukturaldetails->status}}</td>
  <td class="text-center">
  <a href="{{ route('strukturaldetail.show',$strukturaldetails-
>structural_id) }}"class="btn btn-info" id="editButton" data-
target="#editPegawai">
  <i class="cil-zoom-in"></i> </a>
 <a href="{{ route('strukturaldetail.edit',$strukturaldetails-
>structural_id) }}"class="btn btn-warning" id="editButton" data-
target="#editPegawai"> <i class="cil-pencil"></i> </a>
<form action="{{ route('strukturaldetail.destroy', $strukturaldetails-
>structural_id) }}"
  method="post" onclick="return confirm('Anda yakin menghapus data ?')"
```

```
       class="d-inline">
 @csrf
 @method('DELETE')
  <button class="btn btn-youtube">
  <i class="cil-trash"></i> </button> </form> </td> </tr>
 @endforeach
    </tbody> </table> </div> </div> </div> </div> </div>
{{-- end struktural --}}

{{-- Mutasi --}}
    <div class="fade-in">
        <div class="row">
            <div class="col-lg-12">
                <div class="card"> <h5>
                        <div class="card-header">
                            <h4>Riwayat Mutasi</h4>
                            <div class="d-flex">
<a href="{{ route('mutasi.create', ['nip_nik'=> $biodatapegawai->nip_
<div class="card-body">
    <table class="table table-responsive-sm table-striped">
         <thead>
        <tr>
         <th class="text-center">No</th>
         <th class="text-center">Unit Kerja</th>
         <th class="text-center">Tanggal Mutasi</th>
         <th class="text-center">Aksi</th>
@foreach ($biodatapegawai->employee_transfer as $employee_transfers)<tr>
 <td class="text-center">{{ $loop->iteration }}</td>
  <td class="text-center">{{ $employee_transfers->work_unit->name}}</td>
    <td class="text-center">{{ $employee_transfers-
>employee_transfer_date}}</td>
<td class="text-center">
   <a href="{{ route('mutasi.show',$employee_transfers-
>employee_transfer_id) }}"class="btn btn-info" id="editButton" data-
target="#editPegawai">
   <i class="cil-zoom-in"></i> </a>
  <a href="{{ route('mutasi.edit',$employee_transfers-
>employee_transfer_id) }}"class="btn btn-warning" id="editButton" data-
target="#editPegawai">
  <i class="cil-pencil"></i> </a>
  <form action="{{ route('mutasi.destroy',$employee_transfers-
>employee_transfer_id) }}"
 method="post" onclick="return confirm('Anda yakin menghapus data ?')"
   class="d-inline">
 @csrf
 @method('DELETE')
 <button class="btn btn-youtube">
<i class="cil-trash"></i> </button> </form> </td> </tr>
@endforeach </tbody> </table></div></div></div></div></div>
{{-- end mutasi --}}

{{-- Training --}}
    <div class="fade-in">
        <div class="row">
            <div class="col-lg-12">
                <div class="card"><h5>
                        <div class="card-header">
                            <h4>Riwayat Diklat</h4>
            <div class="d-flex">
<a href="{{ route('training.create', ['nip_nik'=> $biodatapegawai-
>nip_nik]) }}" class="btn btn-primary">Tambah</a></div> </div></h5>
   <div class="card-body">
   <table class="table table-responsive-sm table-striped"><thead>
<tr> <th class="text-center">No</th>
     <th class="text-center">Nama Diklat</th>
     <th class="text-center">Tahun</th>
     <th class="text-center">Aksi</th>
     <th class="text-center"></th>
     </tr> </thead> <tbody>
@foreach ($biodatapegawai->training as $trainings)
 <tr> <td class="text-center">{{ $loop->iteration }}</td>
      <td class="text-center">{{ $trainings->training_name }}</td>
      <td class="text-center">{{ $trainings->year }}</td>
      <td class="text-center">
```

```
 <a href="{{ route('training.show',$trainings->training_id) }}"class="btn btn-
info" id="editButton" data-target="#editPegawai">
    <i class="cil-zoom-in"></i> </a>
  <a href="{{ route('training.edit',$trainings->training_id) }}"class="btn btn-
warning" id="editButton" data-target="#editPegawai">
 <i class="cil-pencil"></i> </a>  <form
   action="{{ route('training.destroy', $trainings->training_id) }}"
   method="post" onclick="return confirm('Anda yakin menghapus data ?')"
   class="d-inline">
 @csrf
 @method('DELETE')
 <button class="btn btn-youtube"> <i class="cil-trash"></i> </button> </form>
</td> </tr>
 @endforeach</tbody> </table> </div> </div>   </div>   </div></div>
{{-- end training --}}

{{-- Pangkat Golongan --}}
    <div class="fade-in">
        <div class="row">
            <div class="col-lg-12">
                <div class="card"><h5>
                    <div class="card-header">
                        <h4>Riwayat Pangkat Golongan</h4>
                        <div class="d-flex">
<a href="{{ route('pangkatgolongan.create',  ['nip_nik'=> $biodatapegawai-
>nip_nik]) }}" class="btn btn-primary">Tambah</a></div> </div></h5>
    <div class="card-body">
    <table class="table table-responsive-sm table-striped">
 <thead> <tr>
   <th class="text-center">No</th>
   <th class="text-center">Pangkat Golongan</th>
   <th class="text-center">Status</th>
   <th class="text-center">Aksi</th> </tr> </thead> <tbody>
@foreach ($biodatapegawai->rank_group as $pangkatgolongans)<tr>
  <td class="text-center">{{ $loop->iteration}}</td>
  <td class="text-center">{{ $pangkatgolongans->name  }}</td>
  <td class="text-center">{{ $pangkatgolongans->status}}</td>
  <td class="text-center">
  <a href="{{ route('pangkatgolongan.show',$pangkatgolongans-
>rank_group_id) }}"class="btn btn-info" id="editButton" data-
target="#editPegawai">
  <i class="cil-zoom-in"></i> </a>
  <a href="{{ route('pangkatgolongan.edit',$pangkatgolongans-
>rank_group_id) }}"class="btn btn-warning" id="editButton" data-
target="#editPegawai"> <i class="cil-pencil"></i>    </a>
   <form action="{{ route('pangkatgolongan.destroy', $pangkatgolongans-
>rank_group_id) }}"
 method="post" onclick="return confirm('Anda yakin menghapus data ?')"
 class="d-inline">
@csrf
@method('DELETE')
<button class="btn btn-youtube"><i class="cil-trash"></i></button></form></td>
</tr> @endforeach </tbody> </table> </div></div></div></div></div>
{{-- end pangkat golongan --}}

{{-- data keluarga pegawai --}}
<div class="fade-in">
    <div class="row">
        <div class="col-lg-12">
            <div class="card"> <h5>
                    <div class="card-header">
                        <h4>Data Keluarga</h4>
                        <div class="d-flex">
<a href="{{ route('biodatakeluarga.create',  ['nip_nik'=> $biodatapegawai-
>nip_nik]) }}" class="btn btn-primary">Tambah</a> </div></div></h5>
    <div class="card-body">
     <table class="table table-responsive-sm table-striped"> <thead>
  <tr>  <th class="text-center">No</th>
        <th class="text-center">Nama</th>
        <th class="text-center">Hubungan dengan pegawai</th>
        <th class="text-center">Aksi</th>
        <th class="text-center"></th> </tr> </thead>  <tbody>
@foreach ($biodatapegawai->family as $biodatakeluargas)<tr>
  <td class="text-center">{{ $loop->iteration }}</td>
```

```
  <td class="text-center">{{ $biodatakeluargas->name}}</td>
  <td class="text-center">{{ $biodatakeluargas->relationship }}</td>
  <td class="text-center">
a href="{{ route('biodatakeluarga.show',$biodatakeluargas-
>id_number) }}"class="btn btn-info" id="editButton" data-target="#editPegawai">
   <i class="cil-zoom-in"></i> </a>
 <a href="{{ route('biodatakeluarga.edit',$biodatakeluargas-
>id_number) }}"class="btn btn-warning" id="editButton" data-
target="#editPegawai">
  <i class="cil-pencil"></i> </a> <form
 action="{{ route('biodatakeluarga.destroy', $biodatakeluargas->id_number) }}"
  method="post" onclick="return confirm('Anda yakin menghapus data ?')"
     class="d-inline">
   @csrf
   @method('DELETE')
    <button class="btn btn-youtube">
     <i class="cil-trash"></i> </button> </form></td></tr>
 @endforeach
    </tbody>   </table></div> </div>  </div></div></div>
{{-- end data keluarga pegawai --}}
</div>
@endsection
```

30. *View* biodatapegawai *index,blade*

```
<div class="container-fluid">
    <div class="fade-in">
        <div class="row">
            <div class="col-lg-12">
                <div class="card">  <h5>
                   <div class="card-header">
                      <div class="d-flex">
  <a href="{{ route('biodatapegawai.create') }}" class="btn btn-
primary">Tambah data pegawai</a>  </div> </div></h5>
     <div class="card-body">
       <table class="table table-responsive-sm table-striped">
         <thead>
           <tr>
          <th class="text-center">No</th>
          <th class="text-center">Nama</th>
          <th class="text-center">Unit Kerja</th>
          <th class="text-center">Aksi</th> </tr>
        </thead> <tbody>
 @foreach ($biodatapegawai as $data) <tr>
   <th class="text-center" scope="row">{{$loop->iteration}}.</th>
   <td class="text-center">{{$data->real_name}} <br>{{$data->nip_nik}} </td>
   <td class="text-center">{{ $data->employee_transfer_last() ? $data-
>employee_transfer_last()->work_unit->name : "" }}</td>
 <td class="text-center">
<a href="{{ route('biodatapegawai.show',$data->nip_nik) }}"class="btn btn-
info" id="editButton" data-target="#editPegawai">
       <i class="cil-zoom-in"></i></a>
   <a href="{{ route('biodatapegawai.edit', $data->nip_nik) }}"class="btn btn-
warning" id="editButton" data-target="#editPegawai">
   <i class="cil-pencil"></i> </a>
<form action="{{ URL::route('biodatapegawai.destroy', $data->nip_nik) }}"
   method="post" onclick="return confirm('Anda yakin menghapus data ?')"
       class="d-inline">
  @csrf
  @method('DELETE')
  <button class="btn btn-youtube"> <i class="cil-trash"></i> </button> </form>
  </td> </tr>
@endforeach
    </tbody> </table> </div> </div> </div> </div> </div></div>
@endsection</form> </div></div></div>
```

31. View biodatakeluargauser *index.blade*

```
@extends('layouts.myApp')
@section('content')
<div class="container-fluid">
    <div class="fade-in">
        <div class="row">
```

```
            <div class="col-lg-12">
                <div class="card"> <h5>
                  <div class="card-header">
                   <div class="d-flex">
 <a href="{{ route('biodatakeluarga.create') }}" class="btn btn-
primary">Tambah Biodata Keluarga</a> </div> </div></h5>
<div class="card-body">
 <table class="table table-responsive-sm table-striped"> <thead>
<tr>
      <th class="text-center">No</th>
      <th class="text-center">Nama</th>
      <th class="text-center">Hubungan dengan pegawai</th>
      <th class="text-center">No HP</th>
      <th class="text-center">Tempat Lahir</th>
      <th class="text-center">Tanggal Lahir</th>
      <th class="text-center">Pekerjaan</th>
      <th class="text-center">Pendidikan Terakhir</th>
      <th class="text-center">NPWP</th>
      <th class="text-center">Aksi</th>
      <th class="text-center"></th>  </tr>
   </thead> <tbody>
 @foreach ($biodatakeluarga as $biodatakeluargas)  <tr>
     <td class="text-center">{{ $loop->iteration }}</td>
     <td class="text-center">{{ $biodatakeluargas->name}}</td>
     <td class="text-center">{{ $biodatakeluargas->relationship }}</td>
     <td class="text-center">{{ $biodatakeluargas->phone_no }}</td>
     <td class="text-center">{{ $biodatakeluargas->birth_place }}</td>
     <td class="text-center">{{ $biodatakeluargas->birth_date }}</td>
     <td class="text-center">{{ $biodatakeluargas->occupation }}</td>
     <td class="text-center">{{ $biodatakeluargas->last_education }}</td>
     <td class="text-center">{{ $biodatakeluargas->npwp_no }}</td>
     <td class="text-center">
  <a href="{{ route('biodatakeluarga.show',$biodatakeluargas-
>id_number) }}"class="btn btn-info" id="editButton" data-target="#editPegawai">
   <i class="cil-zoom-in"></i> </a>
     <a href="{{ route('biodatakeluarga.edit',$biodatakeluargas-
>id_number) }}"class="btn btn-warning" id="editButton" data-
target="#editPegawai">
   <i class="cil-pencil"></i> </a>
     <form action="{{ route('biodatakeluarga.destroy', $biodatakeluargas-
>id_number) }}"
  method="post" onclick="return confirm('Anda yakin menghapus data ?')"
   class="d-inline">
 @csrf
 @method('DELETE')
  <button class="btn btn-youtube">
  <i class="cil-trash"></i>
     </button> </form> </td> </tr>
@endforeach </tbody> </table> </div> </div> </div> </div> </div> </div>
@endsection
```

32. View education index.blade

```
<div class="container-fluid">
    <div class="fade-in">
        <div class="row">
            <div class="col-lg-12">
                <div class="card"> <h5>
                    <div class="card-header">
                      <div class="d-flex">  </div>  </div> </h5>
                    <div class="card-body">
<table class="table table-responsive-sm table-striped"> <thead>
       <tr>
       <th class="text-center">No.</th>
       <th class="text-center">Tingkat</th>
       <th class="text-center">Nama Sekolah</th>
       <th class="text-center">Bidang</th>
       <th class="text-center">Tahun Kelulusan</th>
       <th class="text-center">Negara</th>
       <th class="text-center">Dekan/Kepala Sekolah</th>
       <th class="text-center">File Ijazah</th>
       <th class="text-center">Aksi</th>
       <th class="text-center"></th> </tr></thead>  <tbody>
```

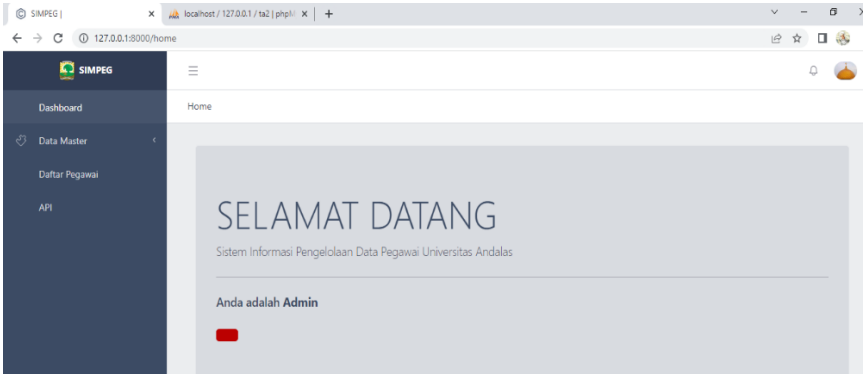
```
 @foreach ($educationdetail as $educationdetails)    <tr>
    <td class="text-center">{{ $loop->iteration }}</td>
    <td class="text-center">{{ $educationdetails->education->level}}</td>
    <td class="text-center">{{ $educationdetails->name}}</td>
    <td class="text-center">{{ $educationdetails->major}}</td>
    <td class="text-center">{{ $educationdetails->graduation_year }}</td>
    <td class="text-center">{{ $educationdetails->country}}</td>
    <td class="text-center">{{ $educationdetails->dean_headmaster}}</td>
    <td class="text-center">{{ $educationdetails->certificate_file }}</td>
    <td class="text-center">
 <a href="{{ route('educationdetail.show',$educationdetails-
>education_details_id) }}"class="btn btn-info" id="editButton" data-
target="#editPegawai">
 <i class="cil-zoom-in"></i> </a>
 <a href="{{ route('educationdetail.edit',$educationdetails-
>education_details_id) }}"class="btn btn-warning" id="editButton" data-
target="#editPegawai">
  <i class="cil-pencil"></i> </a>
 <form
     action="{{ route('educationdetail.destroy', $educationdetails-
>education_details_id) }}"
    method="post" onclick="return confirm('Anda yakin menghapus data ?')"
          class="d-inline">
   @csrf
   @method('DELETE')
   <button class="btn btn-youtube">
   <i class="cil-trash"></i>
   </button> </form> </td> </tr>
   @endforeach
   </tbody> </table> </div></div></div></div></div> </div>
@endsection
```

33. View fungsiona lcreate.blade

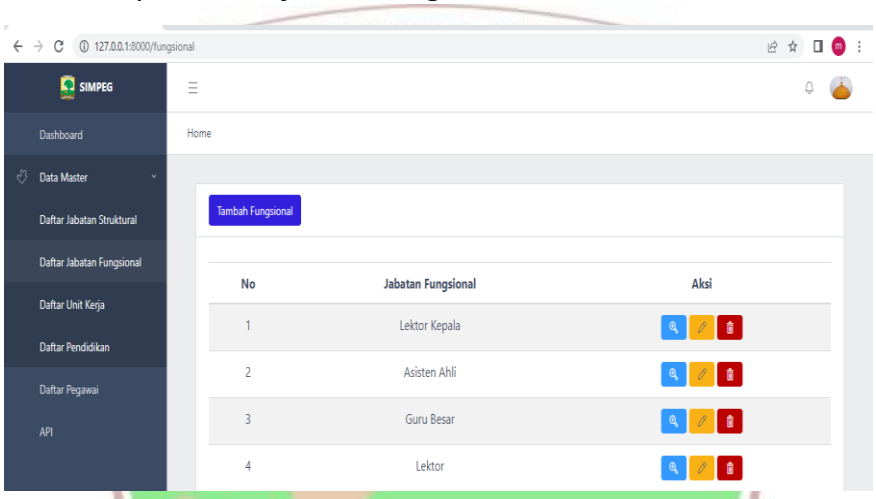
```
<div class="container-fluid">
  <div class="fade-in">
     <div class="row">
        <div class="col-lg-12">
           <div class="card">
             <div class="card-body">
   {{ Form::open(array('url' => route('fungsionaldetail.store'), 'files' => true
 ))}}
 @include('fungsionaldetail._form')
 <div>
<button type="submit" class="btn btn-primary"><span class="cil-save btn-icon mr-
2"></span>Simpan</button>        </div>
   {{ Form::close() }}
       </div>   </div>  </div>  </div>  </div> </div>
@endsection
```

LAMPIRAN G
(IMPLEMENTASI USER INTERFACE)

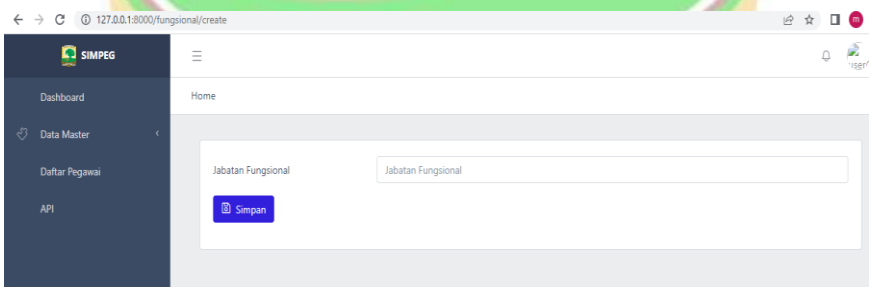
1. *User Interface Homepage* admin



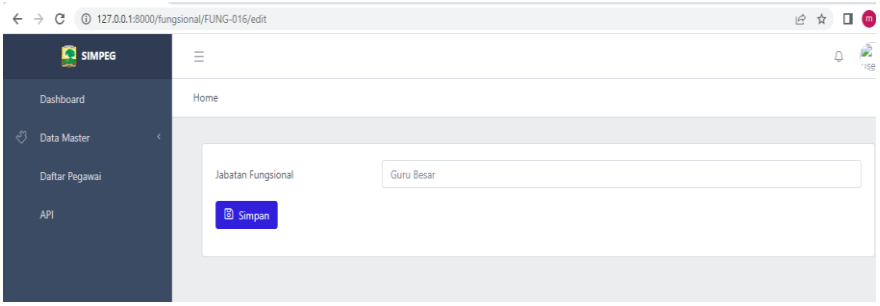
2. *User Interface* daftar jabatan fungsional admin



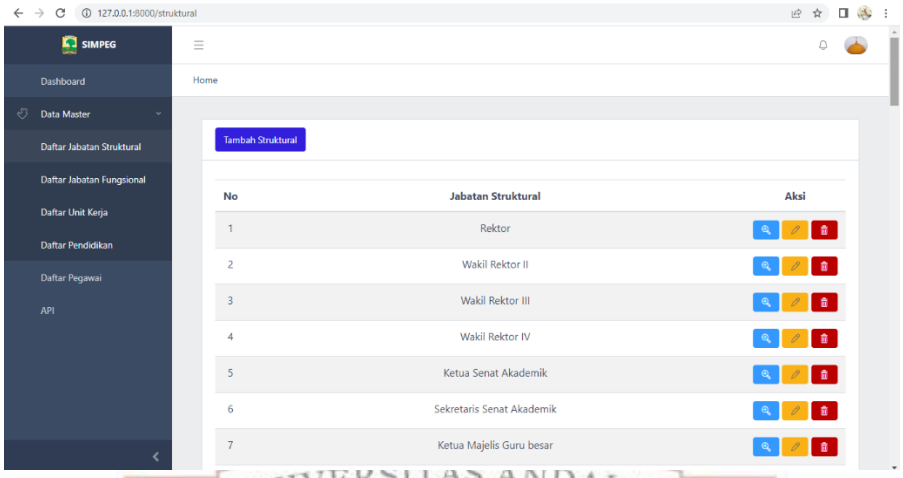
3. *User Interface* tambah jabatan fungsional admin



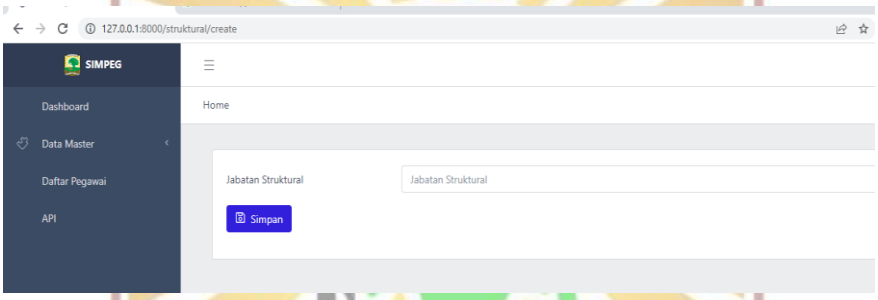
4. *User Interface* edit jabatan fungsional admin



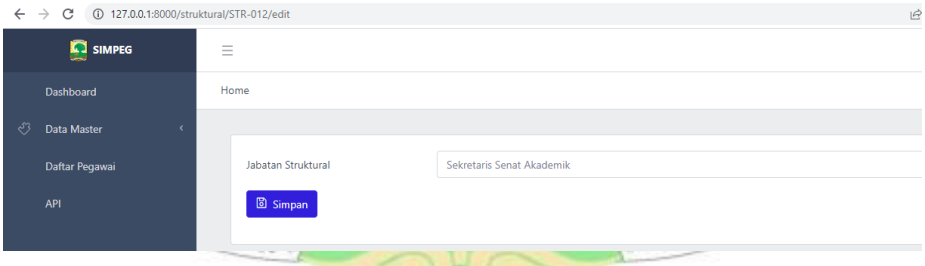
5. *User Interface* daftar jabatan struktural admin



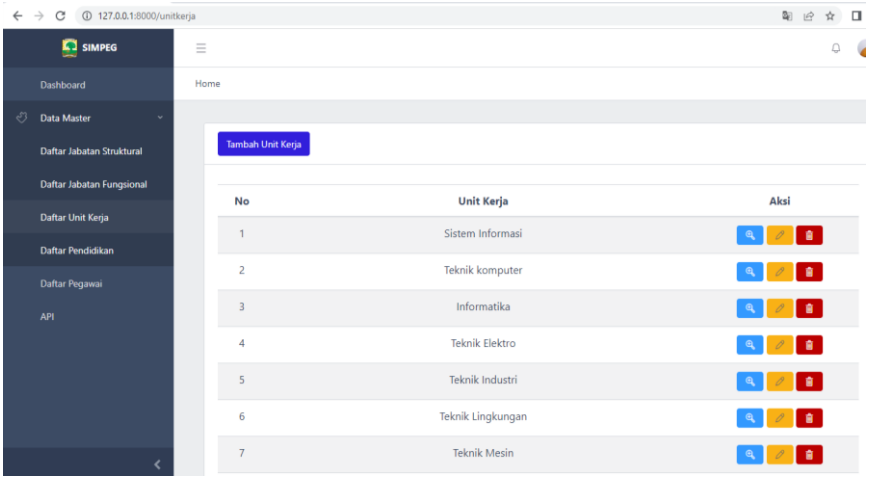
6. *User Interface* tambah jabatan struktural admin



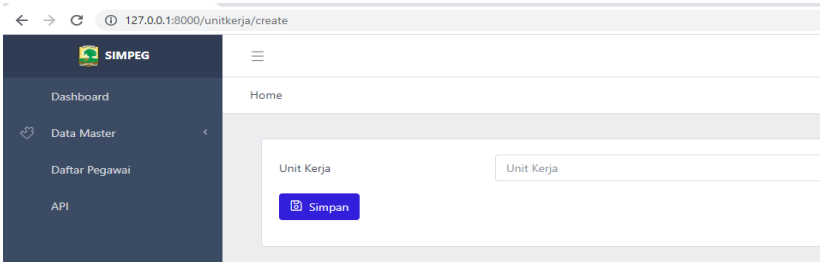
7. *User Interface* edit jabatan struktural admin



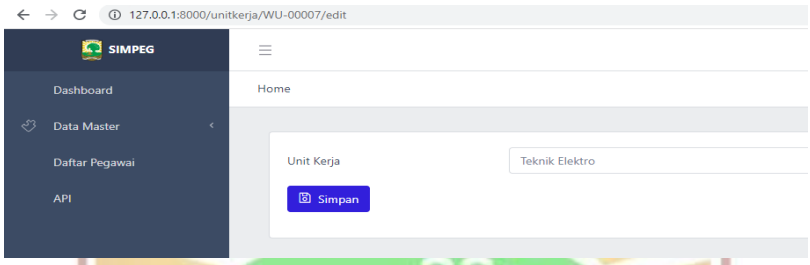
8. *User Interface* daftar unit kerja admin



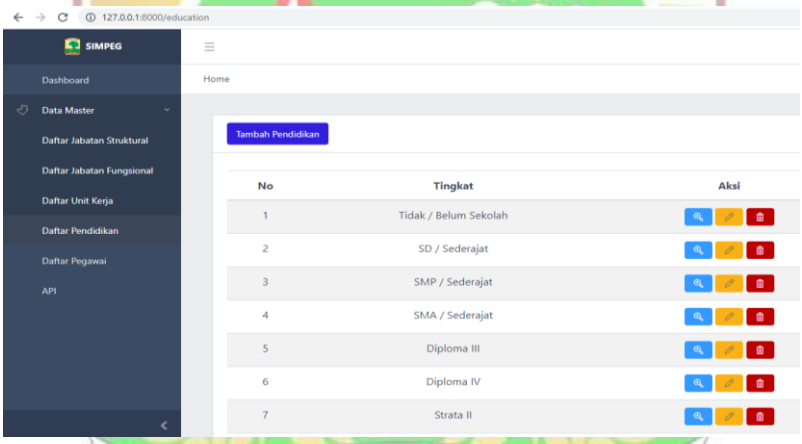
9. *User Interface* tambah unit kerja admin



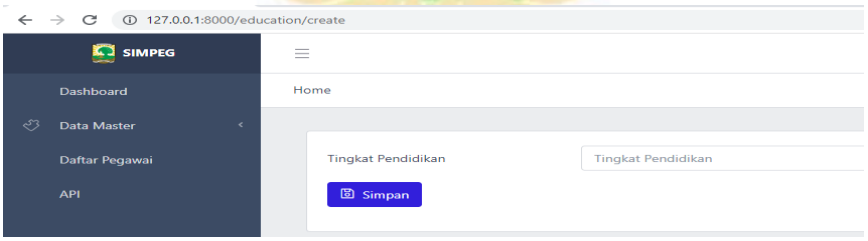
10. *User Interface* edit unit kerja admin



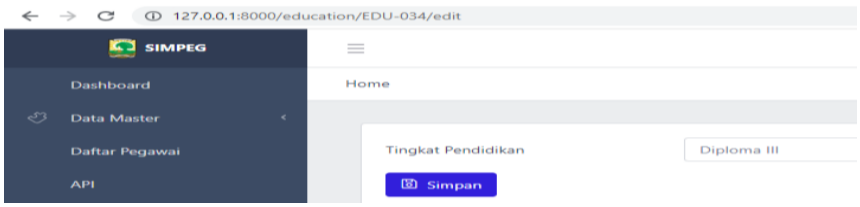
11. *User Interface* daftar pendidikan admin



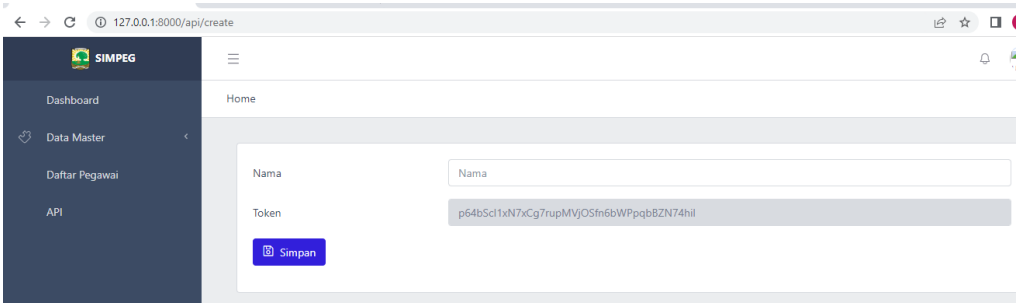
12. *User Interface* tambah pendidikan admin



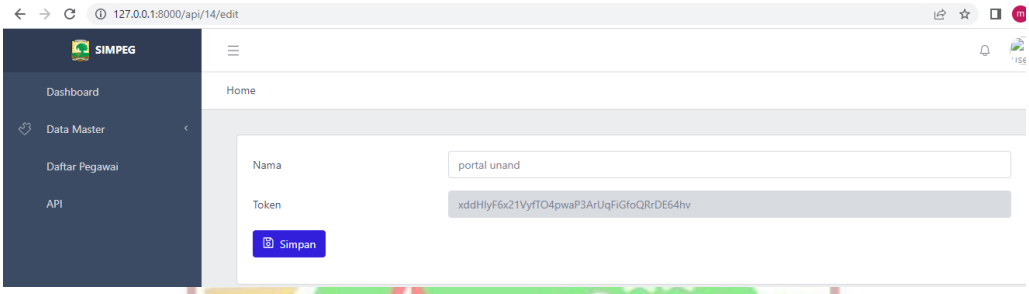
13. *User Interface* edit pendidikan admin



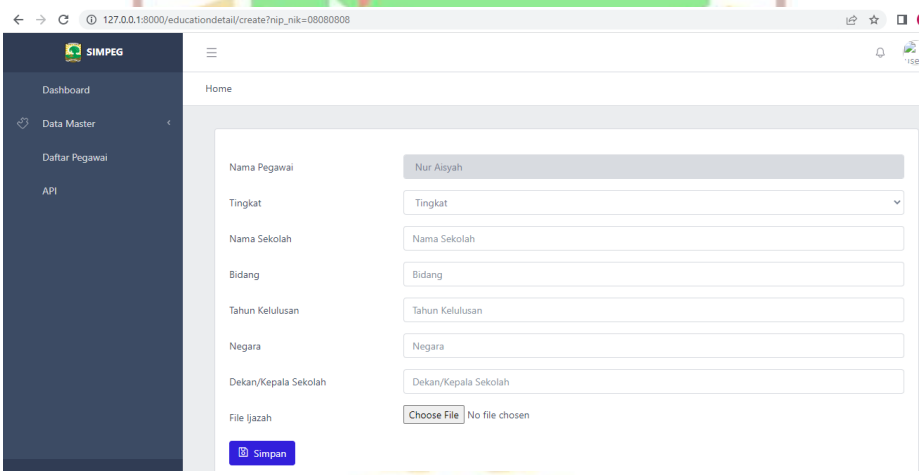
14. *User Interface* tambah API admin



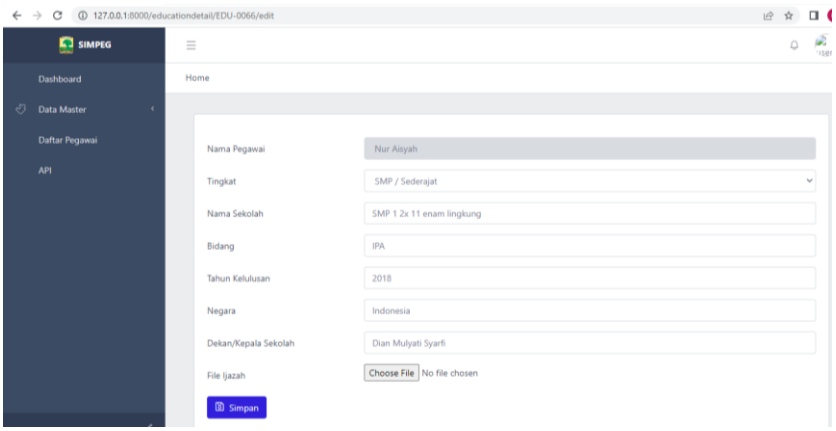
15. *User Interface* edit API admin



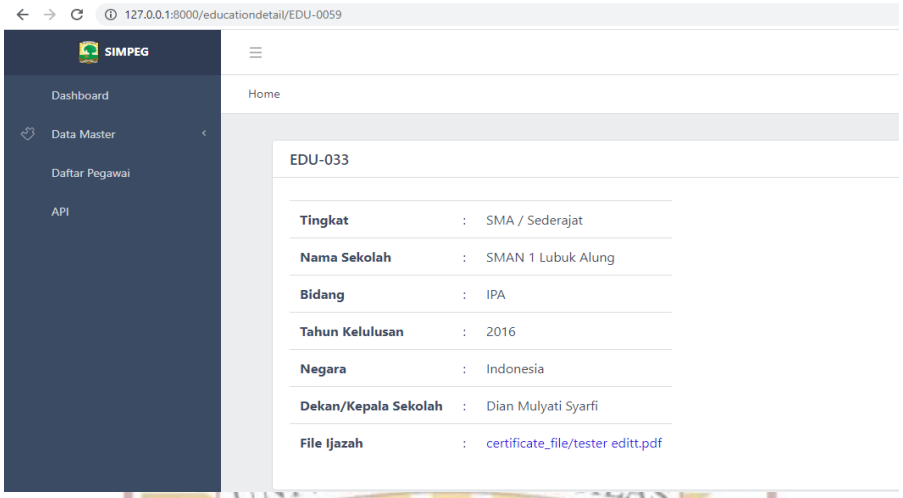
16. *User Interface* tambah riwayat pendidikan admin



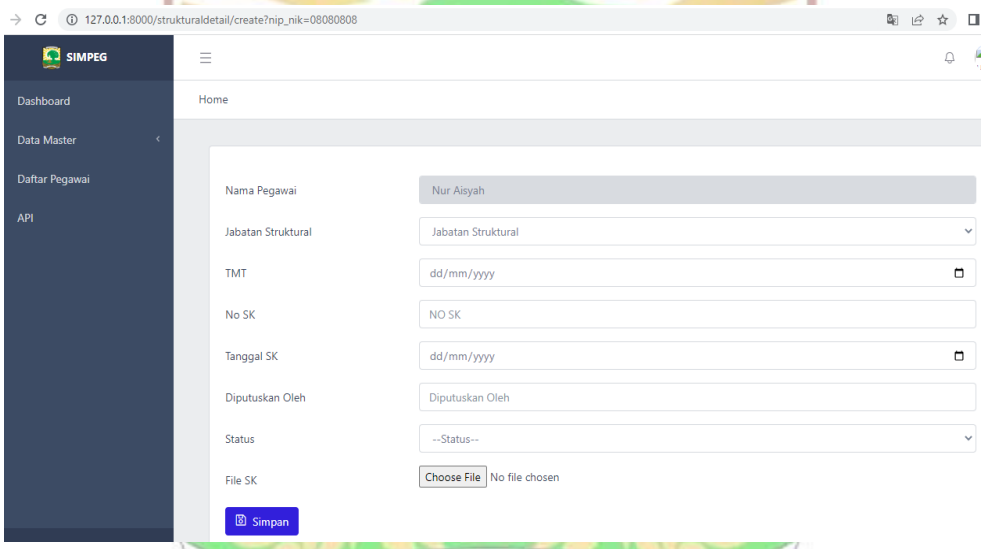
17. *User Interface* edit riwayat pendidikan admin



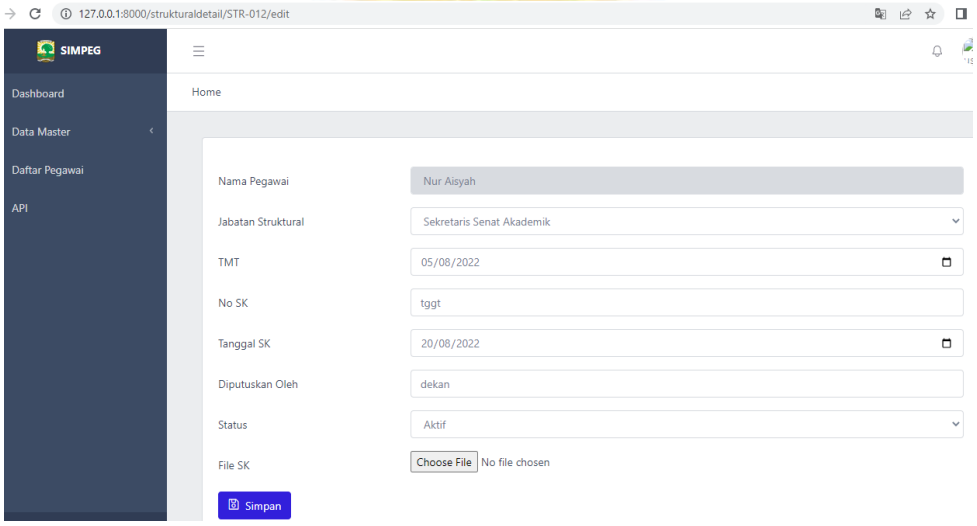
18. *User Interface* lihat riwayat pendidikan admin



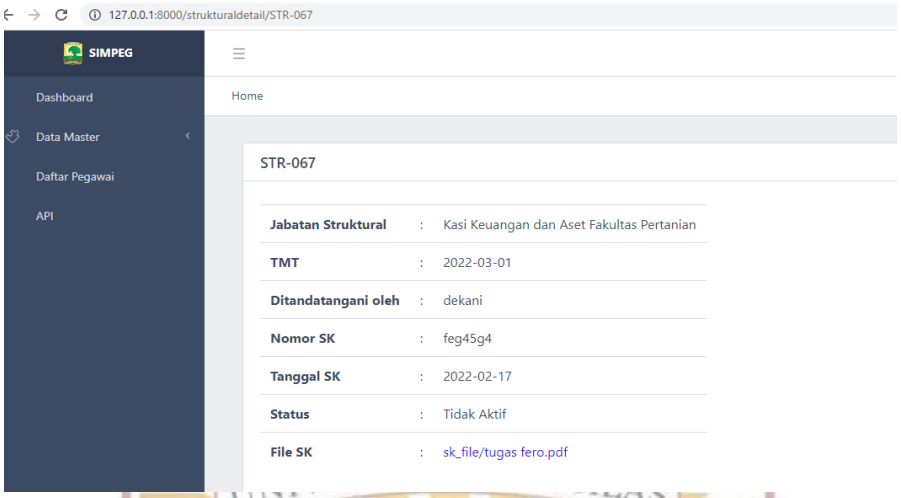
19. *User interface* tambah riwayat jabatan struktural admin



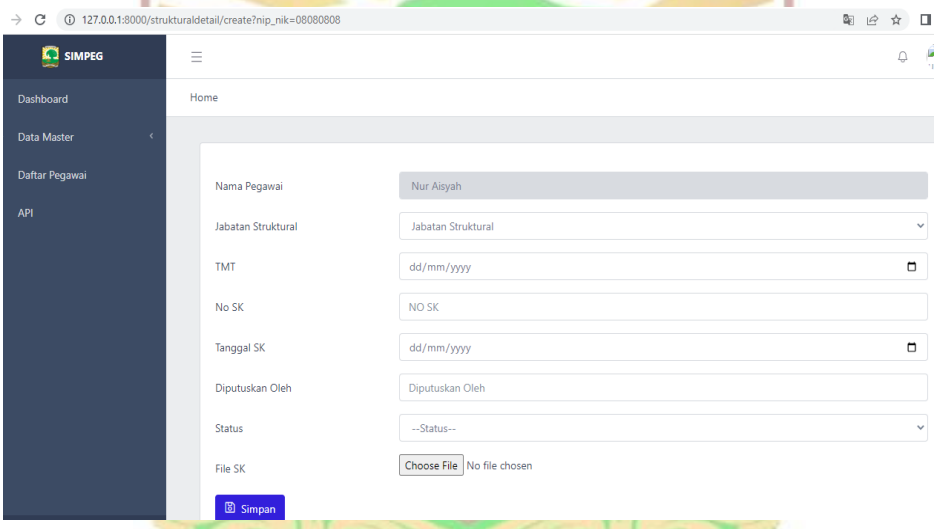
20. User Interface edit riwayat jabatan struktural admin



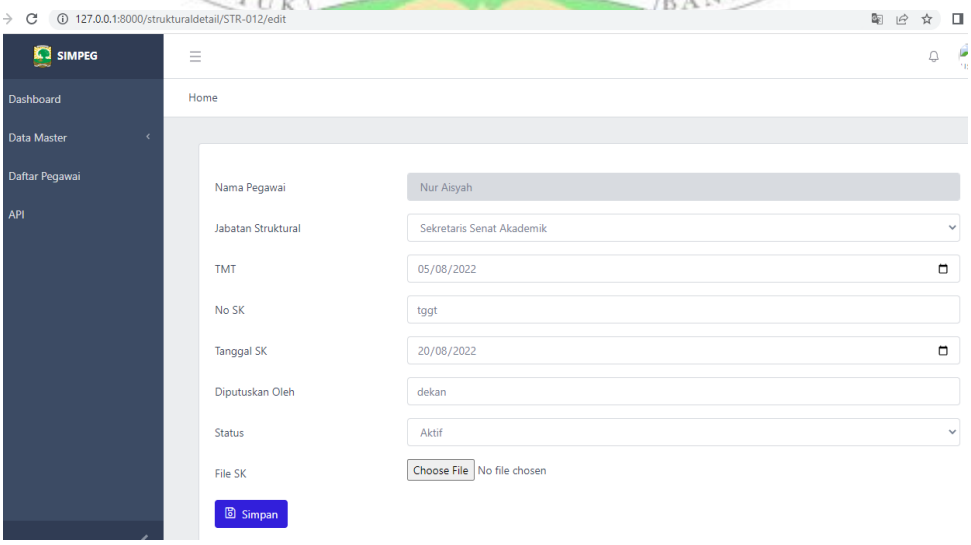
21. *User Interface* lihat riwayat jabatan struktural admin



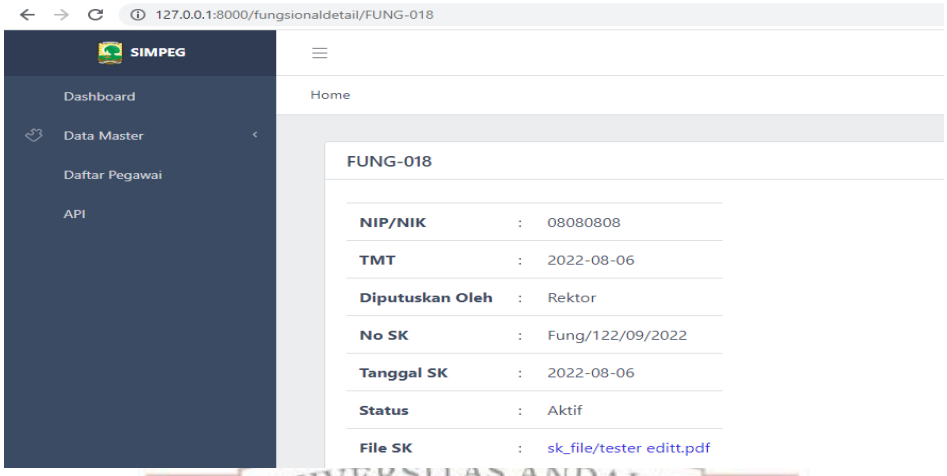
22. *User Interface* tambah riwayat jabatan fungsional admin



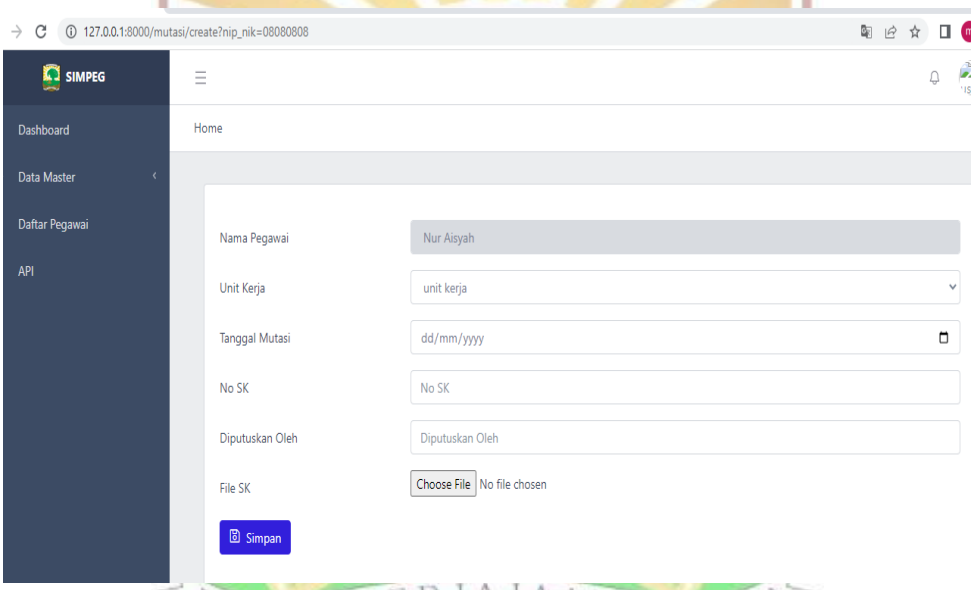
23. *User Interface* edit riwayat jabatan fungsional admin



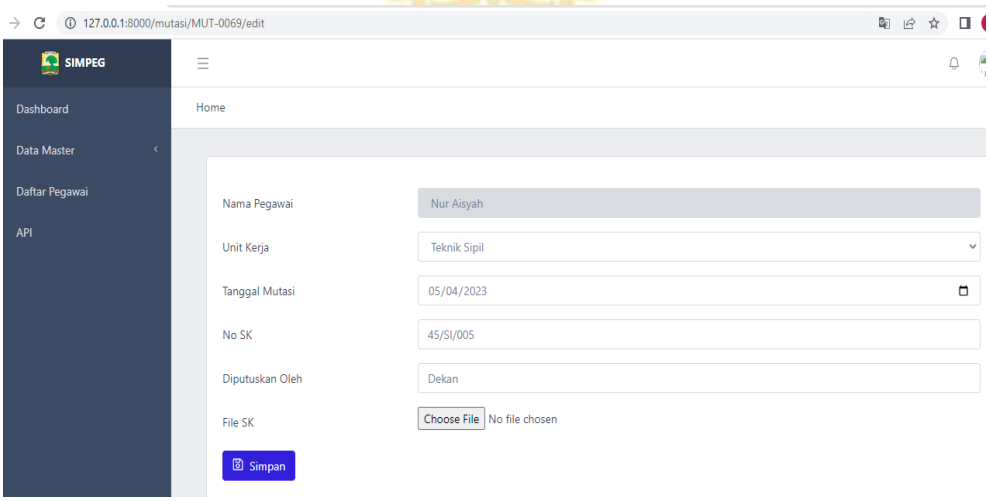
24. *User Interface* lihat jabatan fungsional admin



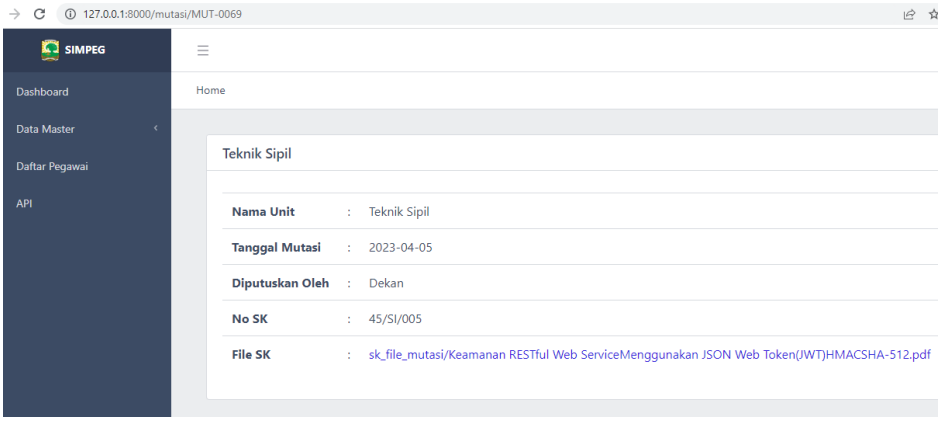
25. *User Interface* tambah riwayat mutasi admin



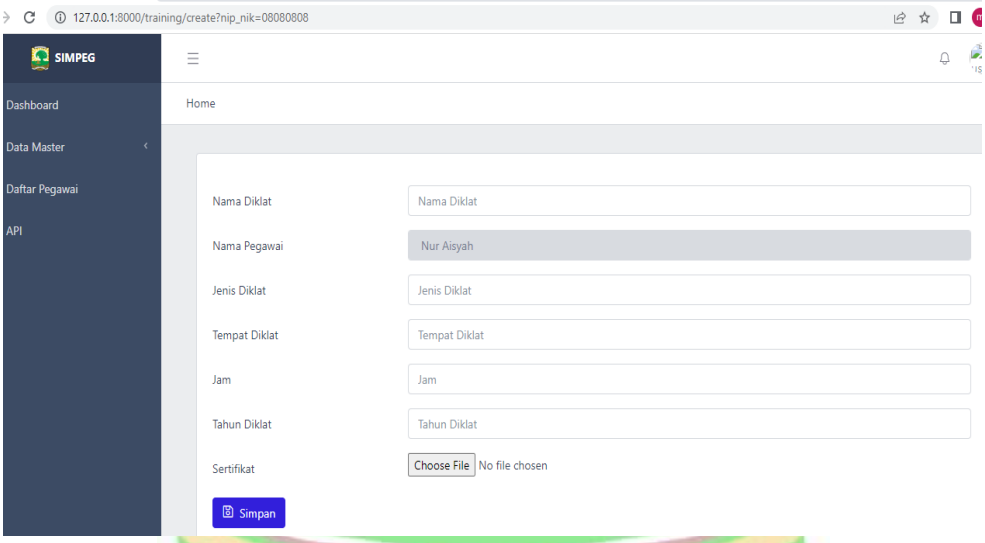
26. *User Interface* edit riwayat mutasi admin



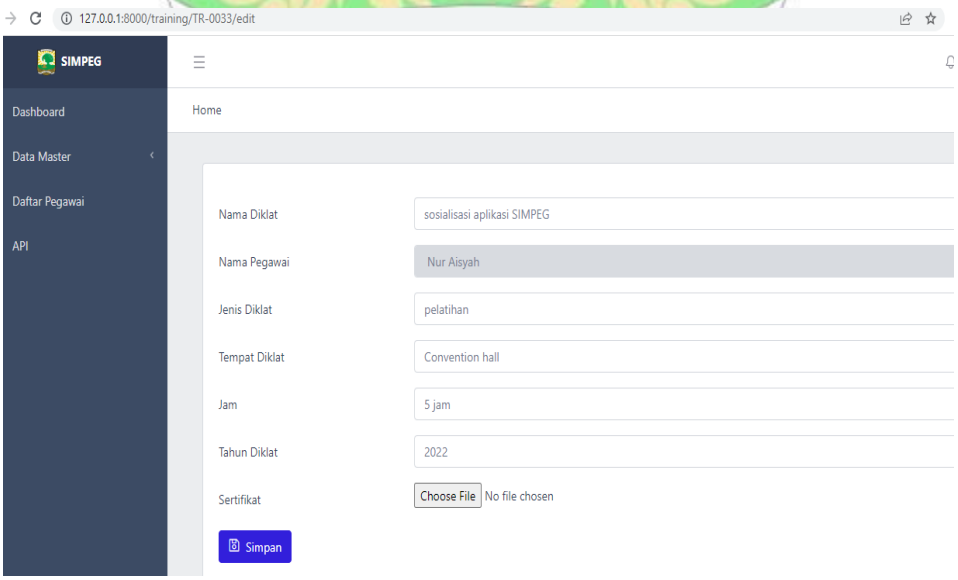
27. *User Interface* lihat riwayat mutasi admin



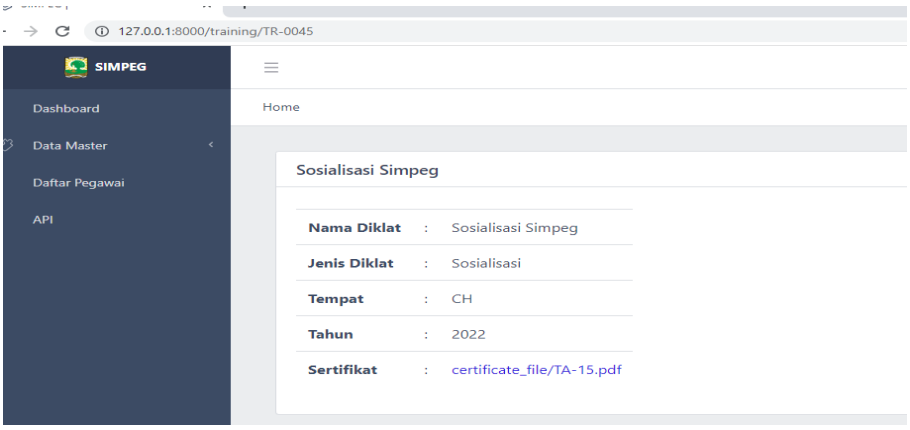
28. *User Interface* tambah riwayat diklat admin



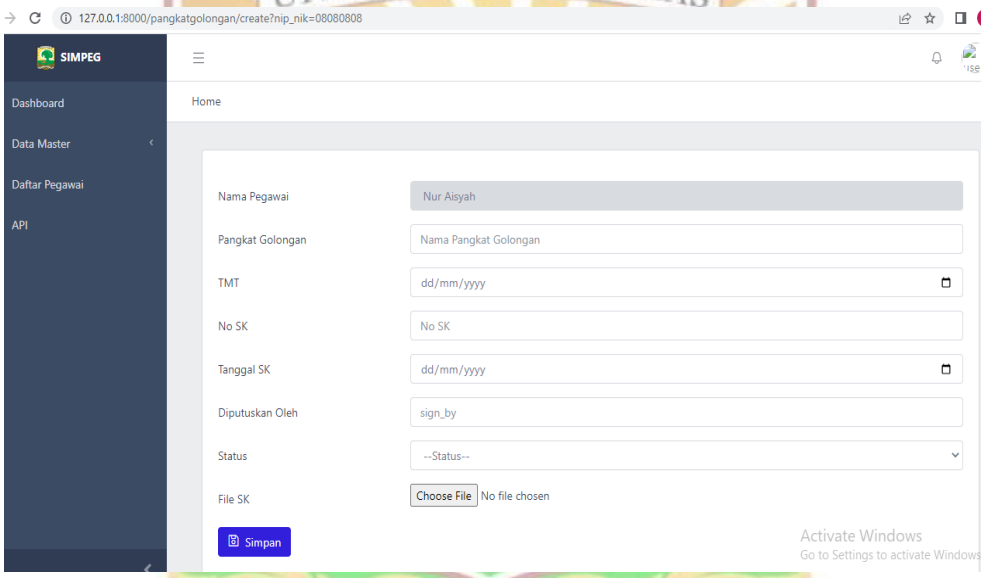
29. *User Interface* edit riwayat diklat admin



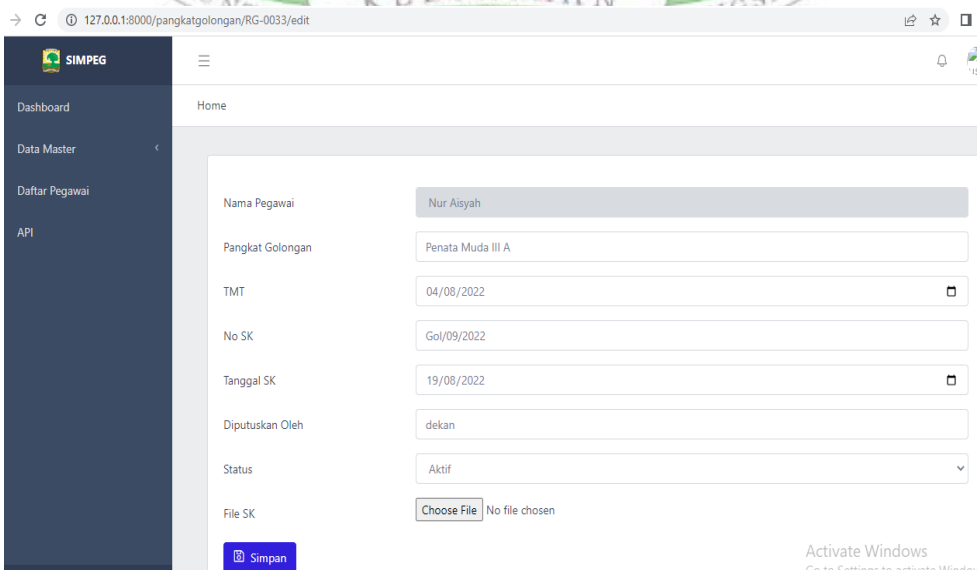
30. *User Interface* lihat riwayat diklat admin



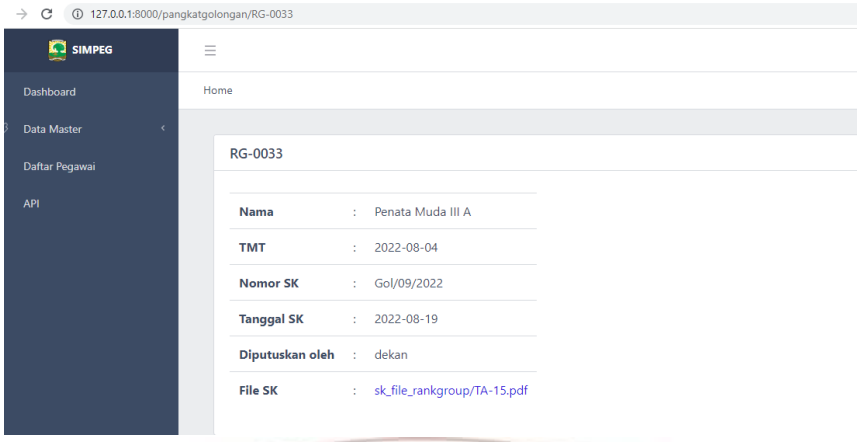
31. *User Interface* tambah riwayat pangkat golongan admin



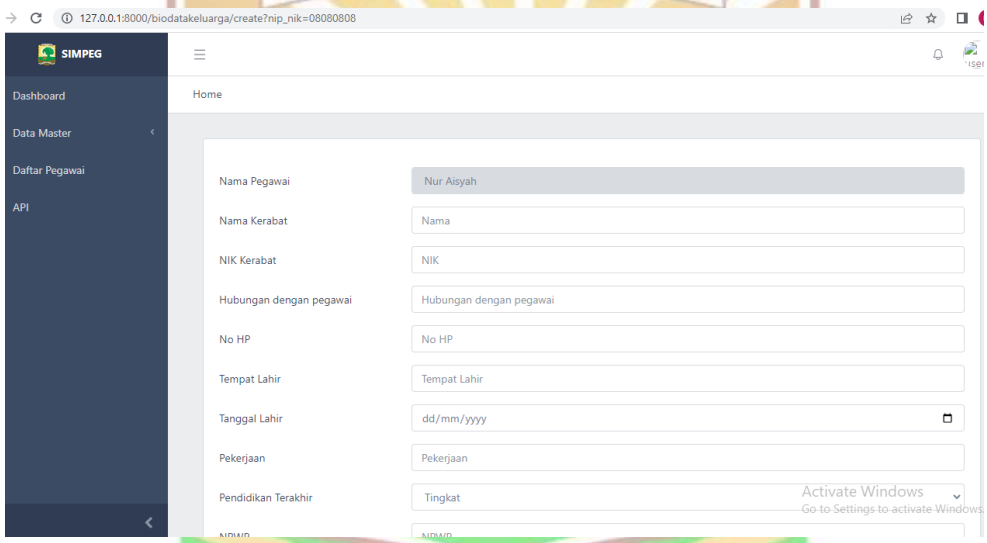
32. *User Interface* edit riwayat pangkat golongan admin



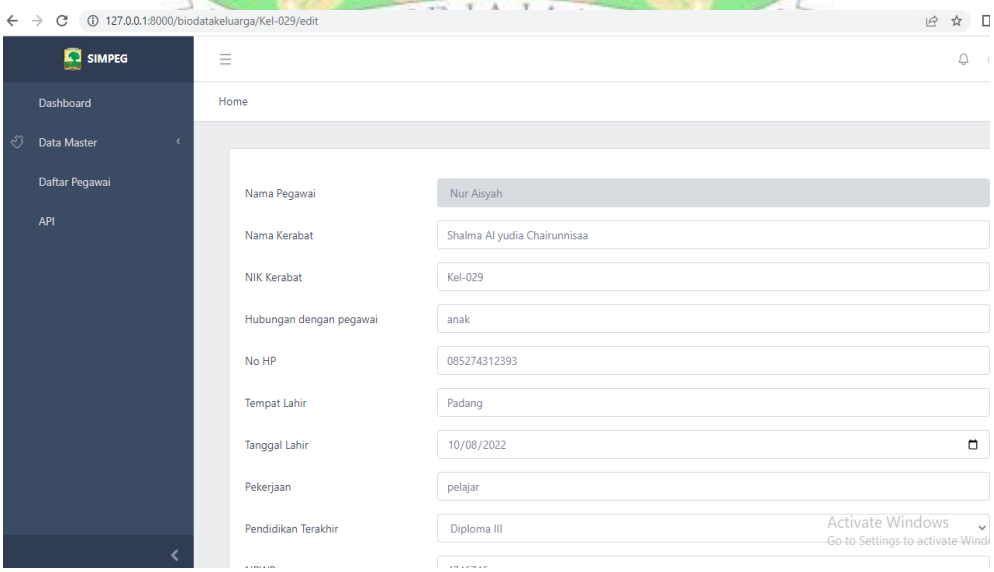
33. *User Interface* lihat riwayat pangkat golongan admin



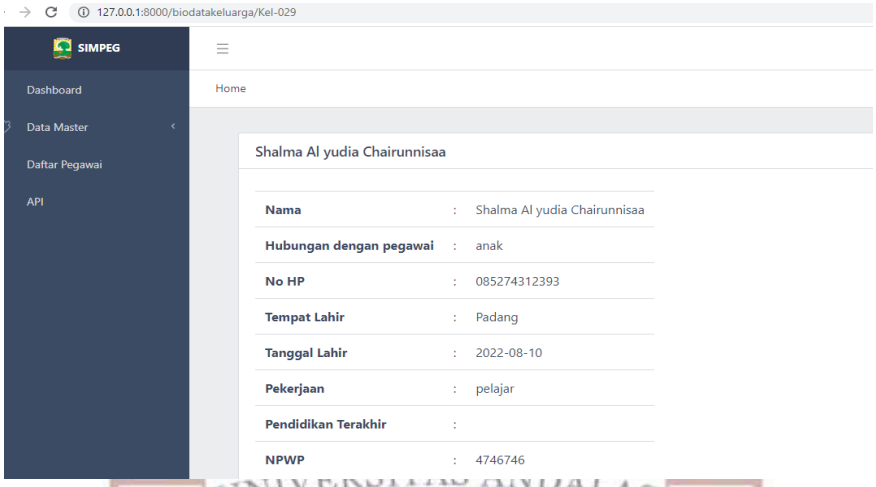
34. *User Interface* tambah biodata keluarga admin



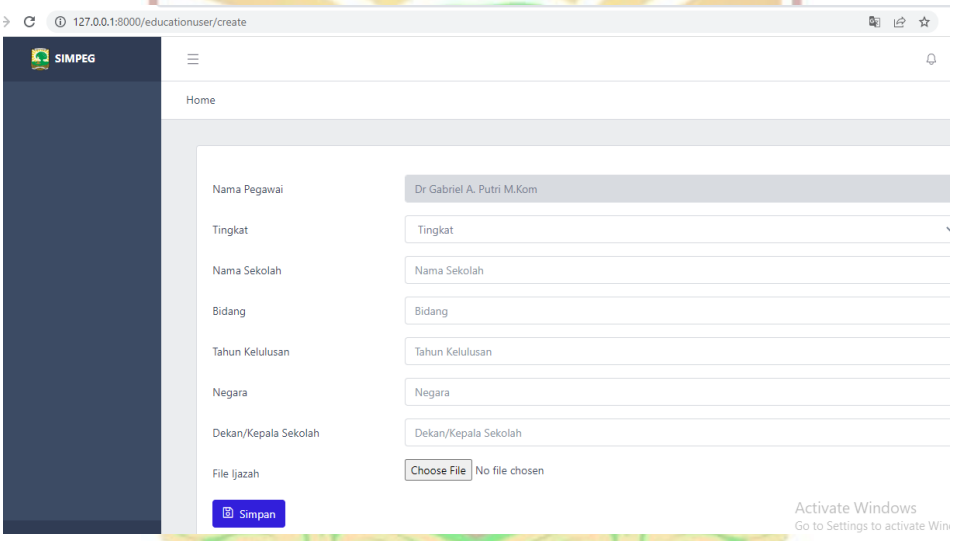
35. *User Interface* edit biodata keluarga admin



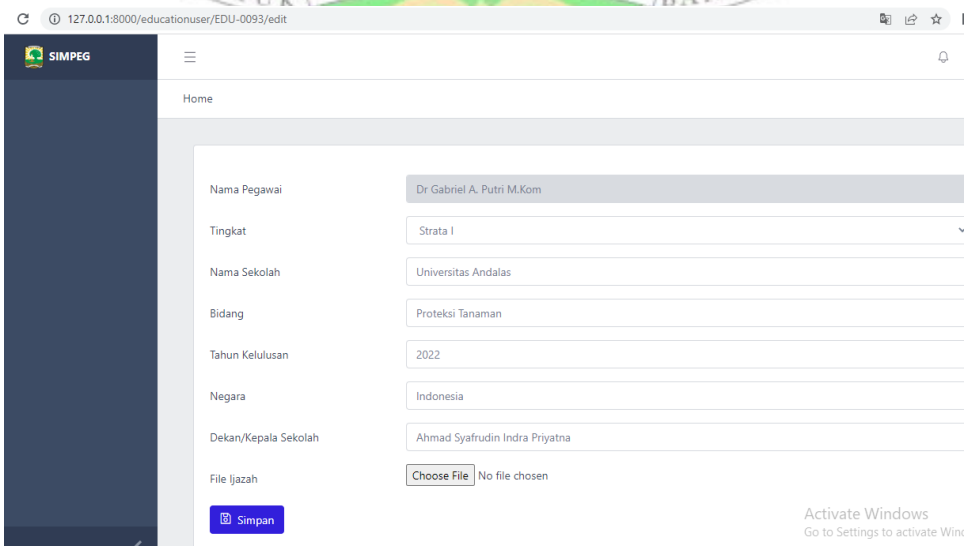
36. *User Interface* lihat biodata keluarga admin



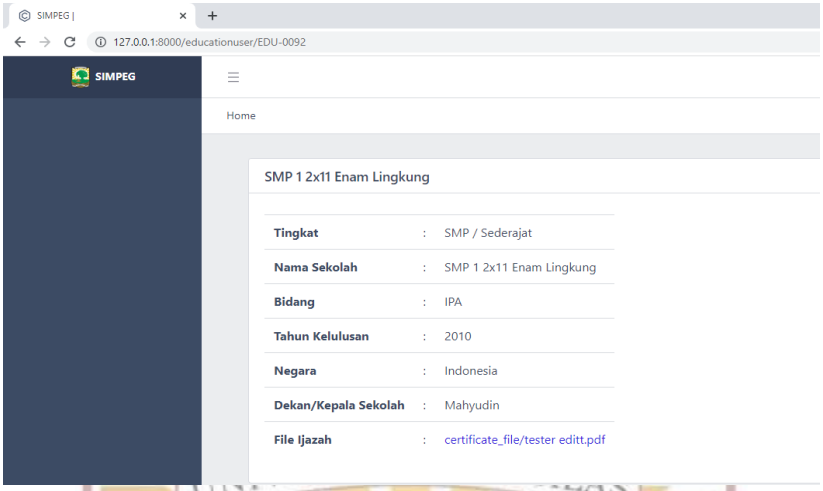
37. *User Interface* tambah riwayat pendidikan user



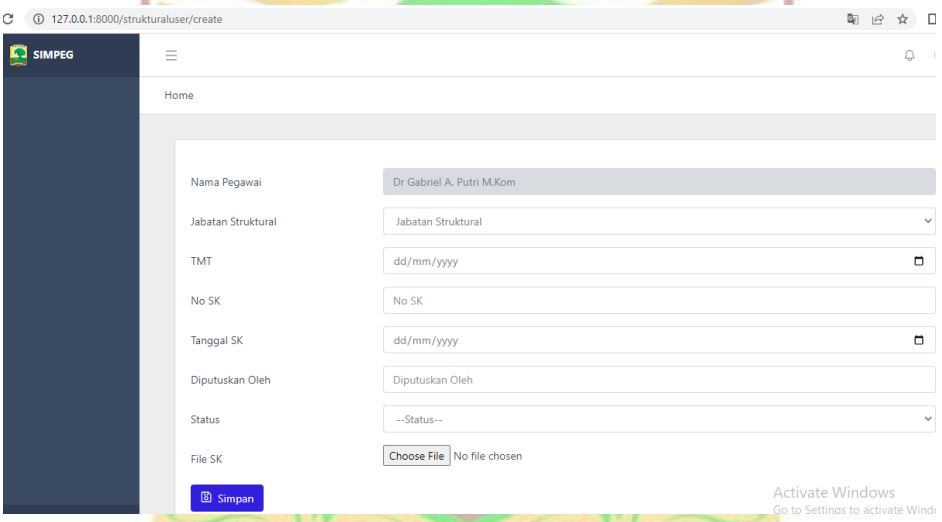
38. *User Interface* edit riwayat pendidikan user



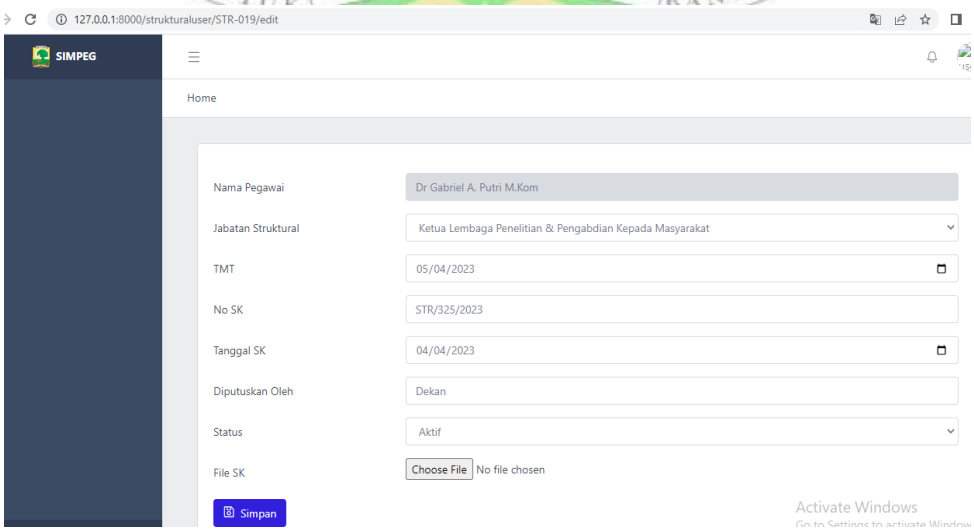
39. *User Interface* lihat riwayat pendidikan user



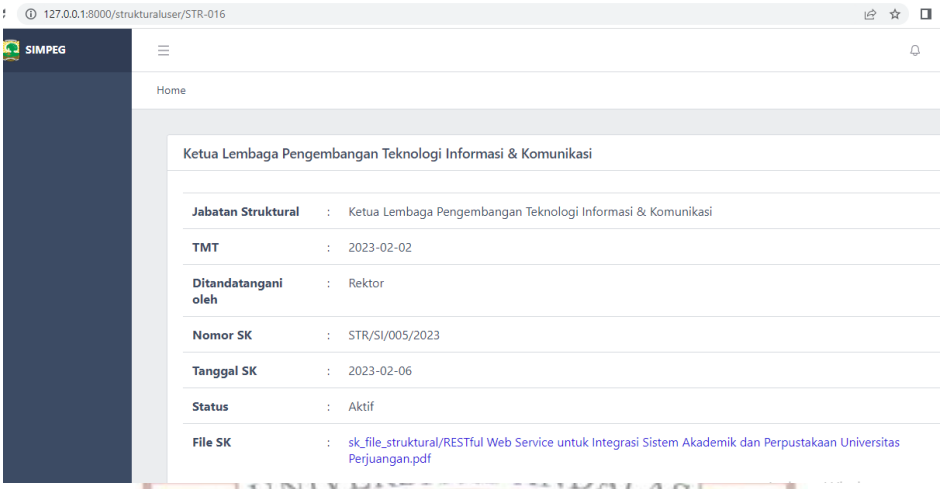
40. *User Interface* tambah riwayat jabatan struktural user



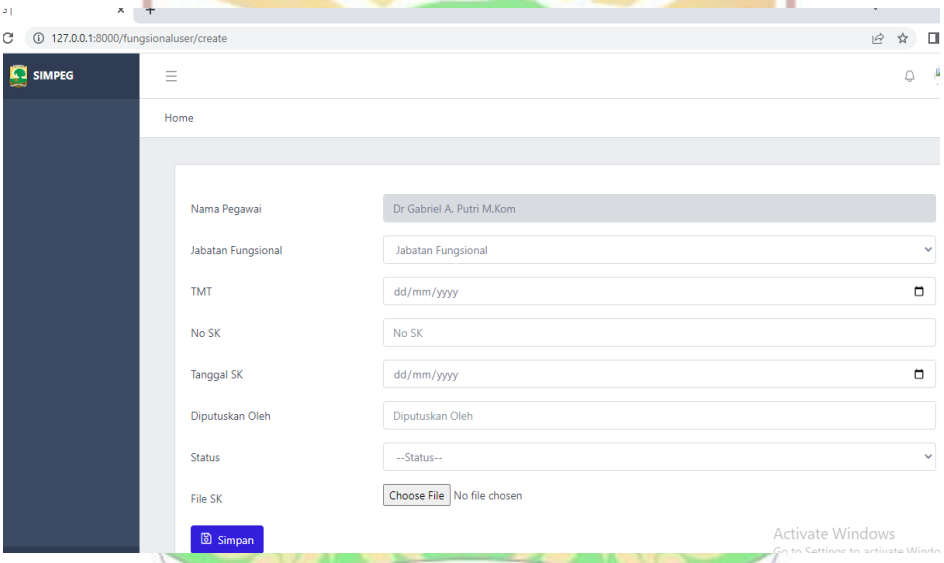
41. *User Interface* edit riwayat jabatan strukural user



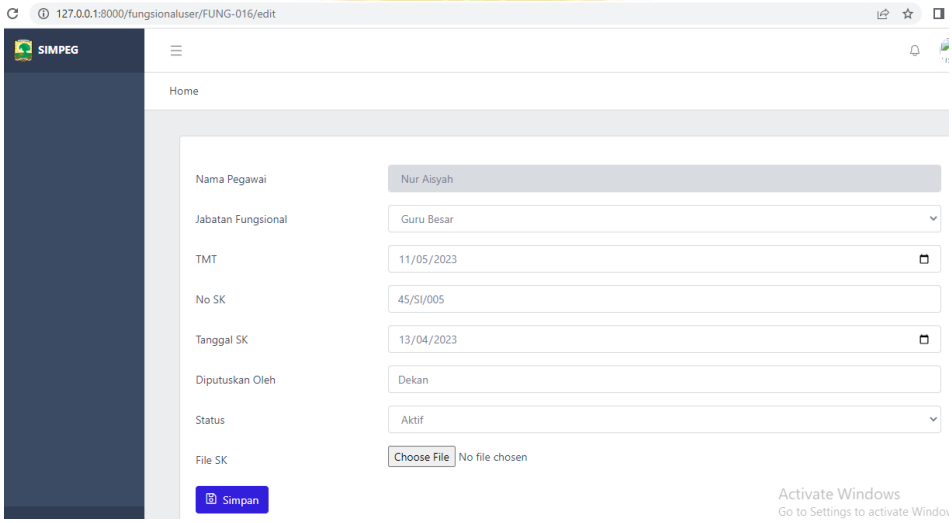
42. *User Interface* lihat riwayat jabatan struktural user



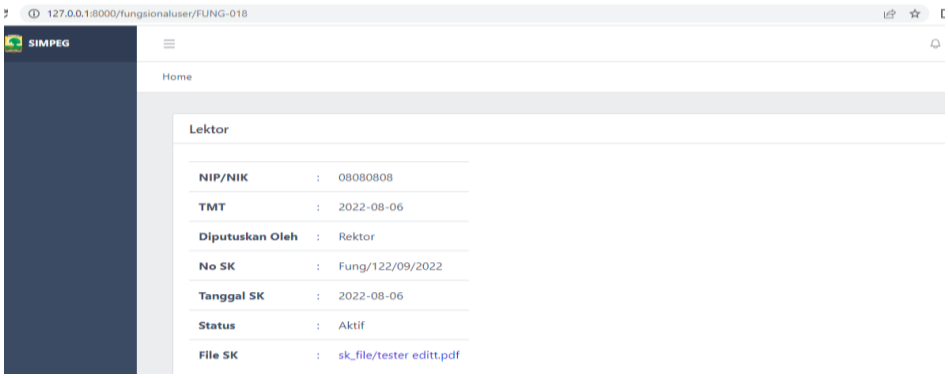
43. *User Interface* tambah riwayat jabatan fungsional user



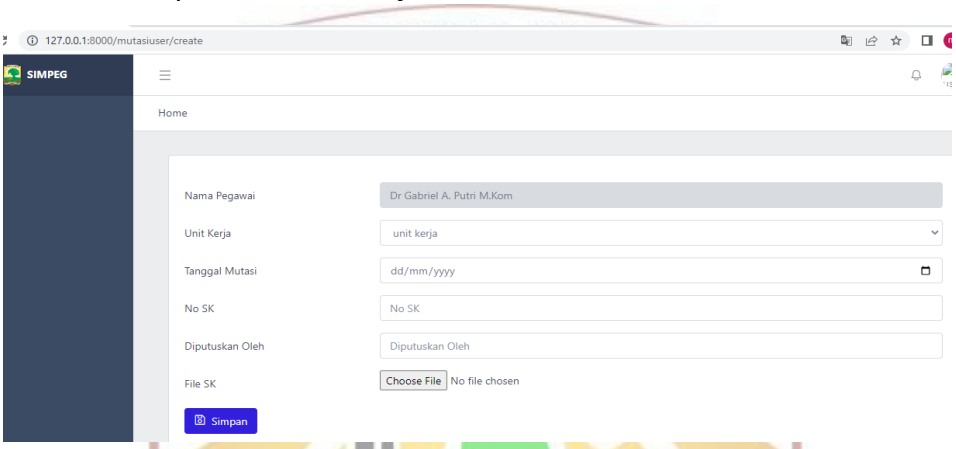
44. *User Interface* edit riwayat jabatan fungsional user



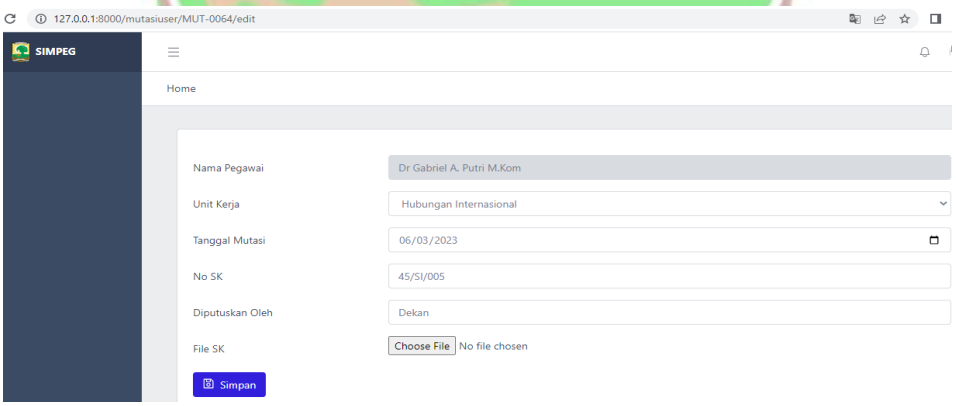
45. *User Interface* lihat jabatan fungsional user



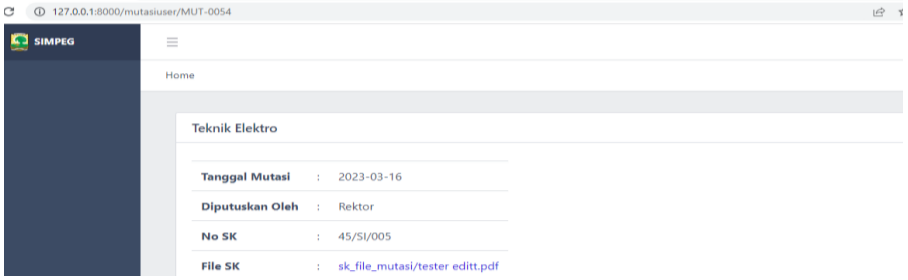
46. *User Interface* tambah riwayat mutasi user



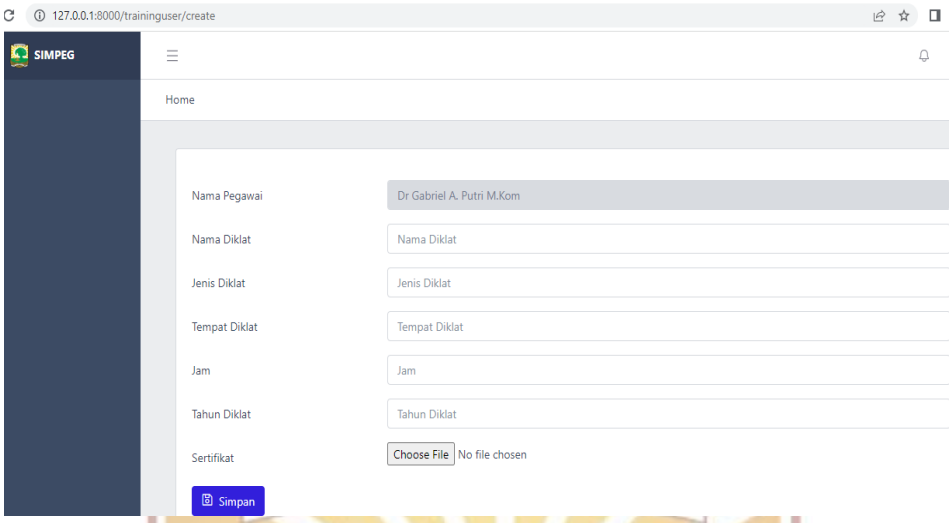
47. *User Interface* edit riwayat mutasi user



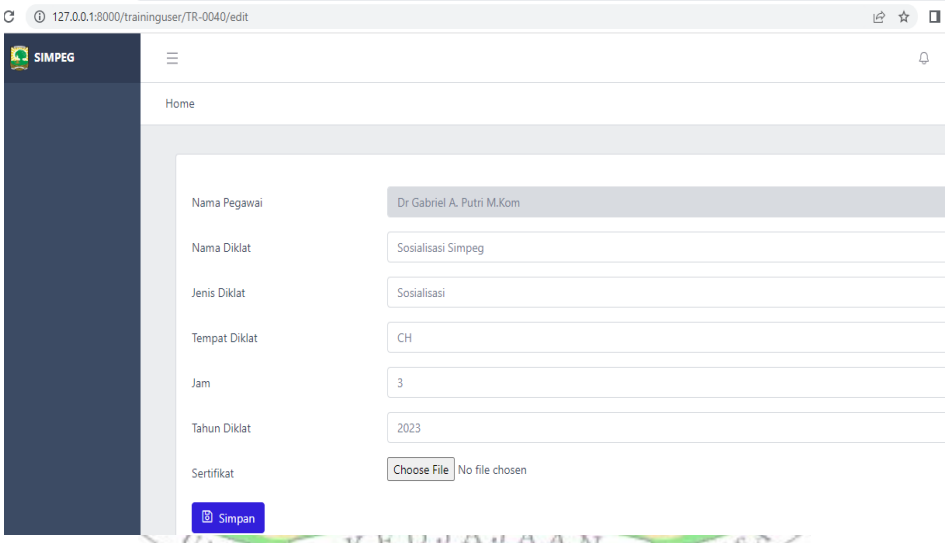
48. *User Interface* lihat riwayat mutasi user



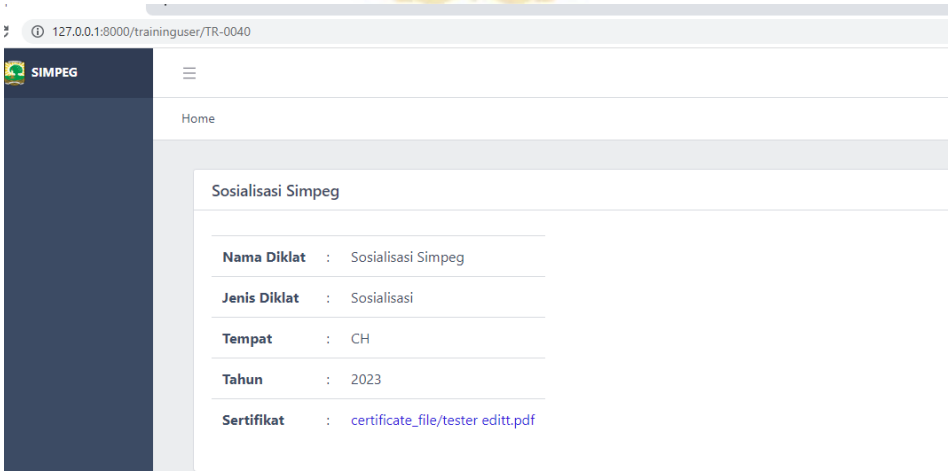
49. *User Interface* tambah riwayat diklat user



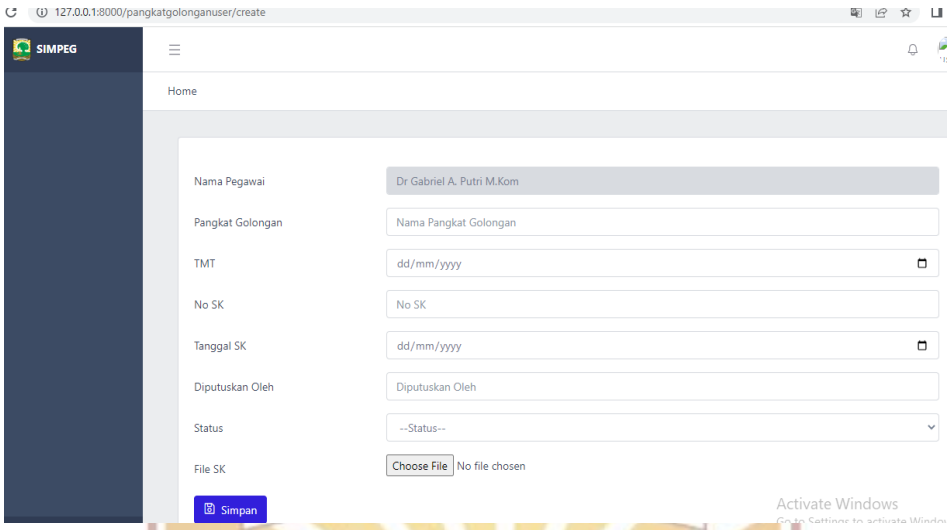
50. *User Interface* edit riwayat diklat user



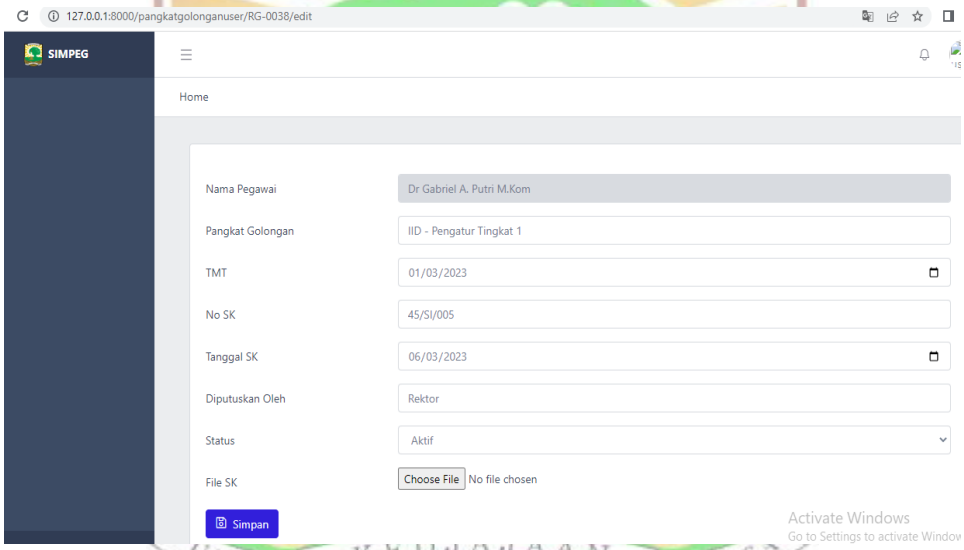
51. *User Interface* lihat riwayat diklat user



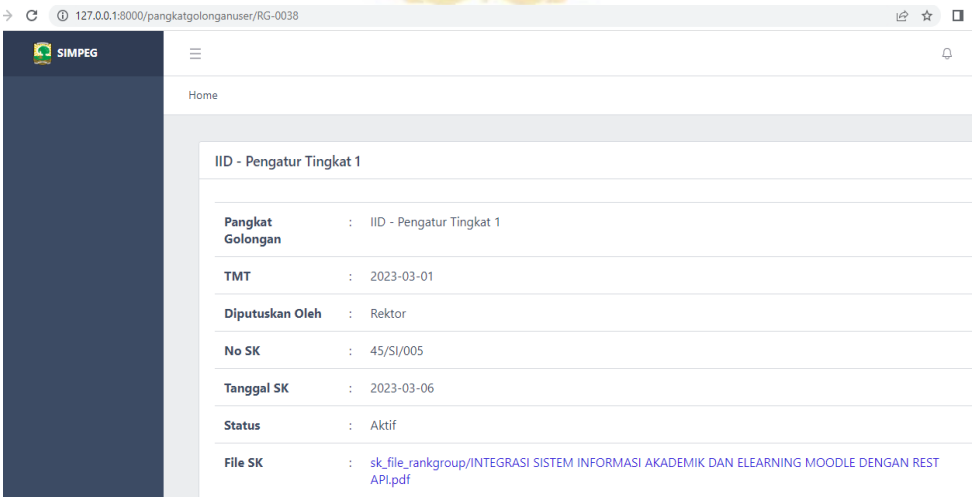
52. *User Interface* tambah riwayat pangkat golongan user



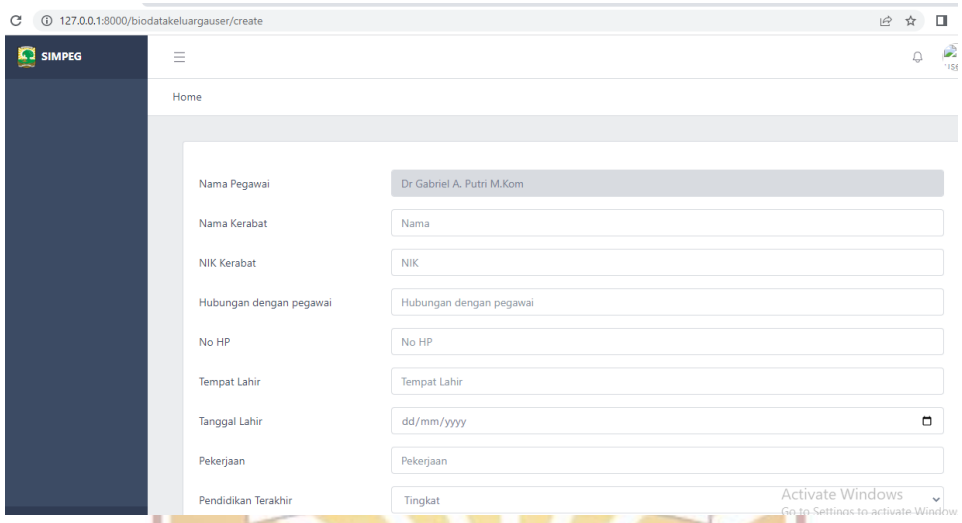
53. *User Interface* edit riwayat pangkat golongan user



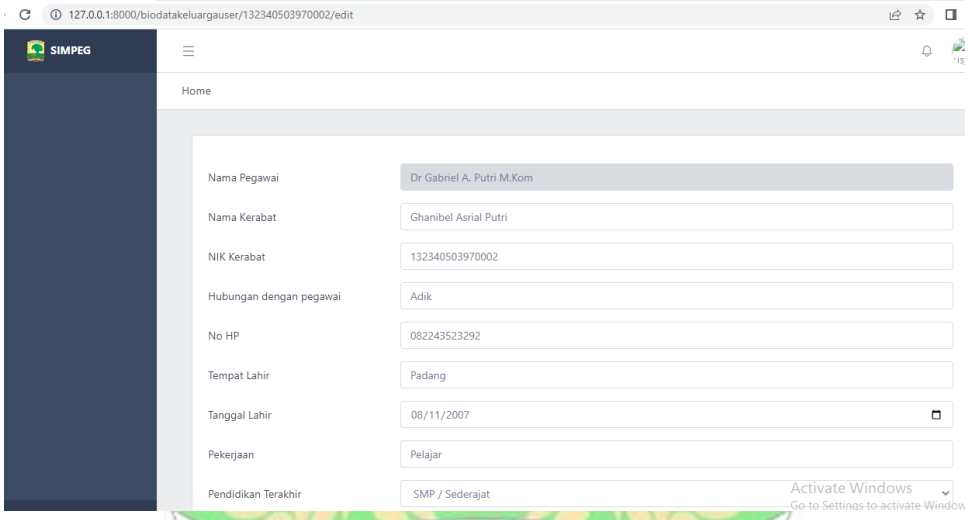
54. *User Interface* lihat riwayat pangkat golongan user



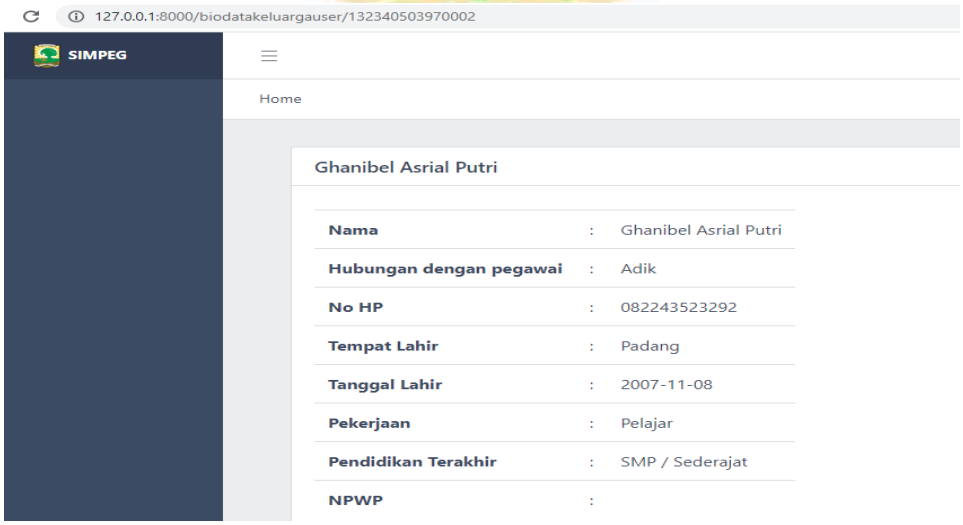
55. *User Interface* tambah biodata keluarga user



56. *User Interface* edit biodata keluarga user



57. *User Interface* lihat biodata keluarga user



LAMPIRAN H
(DOKUMEN PENDUKUNG)

Pertanyaan Terkait Sistem Manajemen Pengelolaan Data Pegawai Unand

Narasumber: Bapak Irzon, M.Kom selaku staff kepegawaian di rektorat

Tanggal: 16 Juni 2021

1. Apakah Sistem Manajemen Pengelolaan Data Pegawai saat ini aktif berjalan?
   Jawab: Saat ini aplikasi tidak berjalan optimal karena kebutuhannya belum memadai.
2. Apa saja kendala dari aplikasi saat ini?
   Jawab: Aplikasi ini tidak berjalan karena data yang ada tidak terbaru, dan masing masing aplikasi di unand menggunakan database pegawai yang berbeda sehingga menyebbkan data tersebut daouble dan tidak sinkron.
3. *User* Simpeg siapa saja?
   Jawab: Admin perfakultas untuk daftarkan user pegawai, dan user itu sendiri yaitu pegawai Unand.
4. Kenapa Simpeg harus menyediakan data untuk aplikasi lain?
   Jawab: Karena masing-masing aplikasi di Unand itu memiliki *database* sendiri, sehingga menimbulkan redudansi data
5. Ketika aplikasi Simpeg tidak berjalan bagaimana cara pengelolaan data pegawai?
6. Jawab: Data pegawai dikumpulkan melalui admin perfakultas dengan file excel yang dikumpulkan ke staff kepegawaian.

FORM PENGUJIAN SISTEM

Nama Aplikasi : Sistem Pengelolaan Data Pegawai Universitas Andalas Berbasis Web Dilengkapi Fasilitas *Web Service*
Nama Penguji : Meysa Putri, S.Kom
Jabatan : User
Pengujian aplikasi (user)

No	Fungsional	Skenario Pengujian	Hasil yang diharapkan	Sesuai	Tidak Sesuai
1	Autentikasi				
	Login	Mengisi NIP/NIK dan Password	Menampilkan halaman *dashboard*	✓	
	Change password	Mengisi *form* ubah *password*	Menampilkan halaman profil	✓	
	Logout	Mengklik *icon* pengguna dan memilih menu *logout*	Menampilkan halaman *login*	✓	
2	Mengelola data pribadi				
	Edit data pribadi	Mengklik *button* "edit" pada data pribadi	Sistem menampilkan *form* edit data pegawai dan menampilkan notifikasi data berhasil disimpan	✓	
	Lihat data pegawai	Mengklik *button login*	Sistem menampilkan halaman data detail pribadi pegawai	✓	
3	Mengelola jabatan fungsional pribadi				
	Tambah jabatan fungsional pribadi	Mengklik *button* "simpan" pada *form* tambah jabatan fungsional pribadi	Sistem menampilkan halaman daftar jabatan fungsional pribadi dan notifikasi data berhasil ditambahkan	✓	

	Edit jabatan fungsional pribadi	Mengklik *button* "edit" pada salah satu data jabatan fungsional pribadi	Sistem menampilkan *form* edit data jabatan fungsional pribadi dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus jabatan fungsional pribadi	Mengklik *button* "hapus" pada salah satu jabatan Fungsional pribadi	Sistem menampilkan halaman daftar jabatan fungsional pribadi dan notifikasi data berhasil dihapus	✓	
	Lihat jabatan fungsional pribadi	Mengklik *button* "*show*" pada salah satu jabatan fungsional pribadi	Sistem menampilkan halaman data detail jabatan fungsional pribadi	✓	
4	Mengelola jabatan struktural pribadi				
	Tambah jabatan struktural pribadi	Mengklik *button* "simpan" pada *form* tambah jabatan struktural pribadi	Sistem menampilkan halaman daftar jabatan struktural pribadi dan notifikasi data berhasil ditambahkan	✓	
	Edit jabatan struktural pribadi	Mengklik *button* "edit" pada salah satu data jabatan struktural pribadi	Sistem menampilkan *form* edit data jabatan struktural pribadi dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus jabatan struktural pribadi	Mengklik *button* "hapus" pada salah satu jabatan struktural pribadi	Sistem menampilkan halaman daftar jabatan struktural pribadi dan notifikasi data berhasil dihapus	✓	
	Lihat jabatan struktural	Mengklik *button* "*show*" pada pribadi	Sistem menampilkan halaman data detail	✓	
5	Mengelola data riwayat pendidikan pribadi				
	Tambah riwayat pendidikan pribadi	Mengklik *button* "simpan" pada *form* tambah riwayat Pendidikan pribadi	Sistem menampilkan halaman daftar riwayat pendidikan pribadi dan notifikasi data berhasil ditambahkan	✓	

	Edit riwayat pendidikan pribadi	Mengklik *button* "edit" pada salah satu data riwayat pendidikan pribadi	Sistem menampilkan *form* edit data riwayat pendidikan pribadidan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus riwayat pendidikan pribadi	Mengklik *button* "hapus" pada salah satu riwayat pendidikan pribadi	Sistem menampilkan halaman daftar riwayat pendidikan pribadi dan notifikasi data berhasil dihapus	✓	
	Lihat riwayat pendidikan pribadi	Mengklik *button* "*show*" pada salah satu riwayat Pendidikan pribadi	Sistem menampilkan halaman data detail riwayat pendidikan pribadi	✓	
6	Mengelola diklat				
	Tambah diklat	Mengklik *button* "simpan" pada *form* tambah diklat	Sistem menampilkan halaman daftar diklat dan notifikasi data berhasil ditambahkan	✓	
	Edit diklat	Mengklik *button* "edit" pada salah satu data diklat	Sistem menampilkan *form* edit data diklat dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus diklat	Mengklik *button* "hapus" pada salah satu diklat	Sistem menampilkan halaman daftar diklat dan notifikasi data berhasil dihapus	✓	
	Lihat diklat	Mengklik *button* "*show*" pada salah satu diklat	Sistem menampilkan halaman data detail diklat	✓	
7	Mengelola pangkat golongan				
	Tambah pangkat golongan	Mengklik *button* "simpan" pada *form* tambah pangkat golongan	Sistem menampilkan halaman daftar pangkat golongan dan notifikasi data berhasil ditambahkan	✓	
		satu data pangkat golongan	pangkat golongan dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus pangkat golongan	Mengklik *button* "hapus" pada salah satu pangkat golongan	Sistem menampilkan halaman daftar pangkat golongan dan notifikasi data berhasil dihapus	✓	

	Lihat pangkat golongan	Mengklik *button* "*show*" pada salah satu pangkat golongan	Sistem menampilkan halaman data detail pangkat golongan	✓	
8	Mengelola mutasi pegawai				
	Tambah mutasi pegawai	Mengklik *button* "simpan" pada *form* tambah mutasi pegawai	Sistem menampilkan halaman daftar mutasi pegawai dan notifikasi data berhasil ditambahkan	✓	
	Edit mutasi pegawai	Mengklik *button* "edit" pada salah satu data mutasi pegawai	Sistem menampilkan *form* edit data mutasi pegawai dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus mutasi pegawai	Mengklik *button* "hapus" pada salah satu mutasi pegawai	Sistem menampilkan halaman daftar mutasi pegawai dan notifikasi data berhasil dihapus	✓	
	Lihat mutasi pegawai	Mengklik *button* "*show*" pada salah satu mutasi pegawai	Sistem menampilkan halaman data detail mutasi pegawai	✓	
9	Mengelola data keluarga pegawai				
	Tambah data keluarga pegawai	Mengklik *button* "simpan" pada *form* tambah data keluarga pegawai	Sistem menampilkan halaman daftar data keluarga pegawai dan notifikasi data berhasil ditambahkan	✓	
	Edit data keluarga pegawai	Mengklik *button* "edit" pada salah satu data data keluarga pegawai	Sistem menampilkan *form* edit data keluarga pegawai dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus data keluarga pegawai	Mengklik *button* "hapus" pada salah satu data	Sistem menampilkan halaman daftar data keluarga pegawai	✓	
	Lihat data keluarga pegawai	Mengklik *button* "*show*" pada salah satu data keluarga pegawai	Sistem menampilkan halaman data detail data keluarga pegawai		
10	Mengelola unit kerja				
	Tambah unit kerja	Mengklik *button* "simpan" pada *form* tambah unit kerja	Sistem menampilkan halaman daftar unit kerja dan notifikasi data berhasil	✓	

	Edit unit kerja	Mengklik *button* "edit" pada salah satu data unit kerja	Sistem menampilkan *form* edit data unit kerja dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus unit kerja	Mengklik *button* "hapus" pada salah satu unit kerja	Sistem menampilkan halaman daftar unit kerja dan notifikasi data berhasil dihapus	✓	
	Lihat unit kerja	Mengklik *button* "*show*" pada salah satu unit kerja	Sistem menampilkan halaman data detail unit kerja	✓	

Padang, 5 April 2023

Meysa Putri, S.Kom

FORM PENGUJIAN SISTEM

Nama Aplikasi : Sistem Pengelolaan Data Pegawai Universitas Andalas Berbasis Web Dilengkapi Fasilitas *Web Service*
Nama Penguji : Nindy Malisha, SE
Jabatan : Admin
Pengujian aplikasi (admin)

No	Fungsional	Skenario Pengujian	Hasil yang diharapkan	Sesuai	Tidak Sesuai
1	Autentikasi				
	Login	Mengisi NIP/NIK dan Password	Menampilkan halaman *dashboard*	✓	
	Change password	Mengisi *form* ubah *password*	Menampilkan halaman profil	✓	
	Logout	Mengklik *icon* pengguna dan memilih menu *logout*	Menampilkan halaman *login*	✓	
2	Mengelola data pegawai				
	Tambah data pegawai	Mengklik *button* "simpan" pada *form* tambah pegawai	Sistem menampilkan halaman daftar pegawai dan notifikasi data berhasil ditambahkan	✓	
	Edit data pegawai	Mengklik *button* "edit" pada salah satu data pegawai	Sistem menampilkan *form* edit data pegawai dan menampilkan berhasil disimpan	✓	
	Hapus data pegawai	Mengklik *button* "hapus" pada salah satu pegawai	Sistem menampilkan halaman daftar pegawai dan notifikasi data berhasil dihapus	✓	

	Lihat data pegawai	Mengklik *button* "*show*" pada salah satu pegawai	Sistem menampilkan halaman data detail pegawai	✓	
3	Mengelola jabatan fungsional				
	Tambah jabatan fungsional	Mengklik *button* "simpan" pada *form* tambah jabatan fungsional	Sistem menampilkan halaman daftar jabatan fungsional dan notifikasi data berhasil ditambahkan	✓	
	Edit jabatan fungsional	Mengklik *button* "edit" pada salah satu data jabatan fungsional	Sistem menampilkan *form* edit data jabatan fungsional dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus jabatan fungsional	Mengklik *button* "hapus" pada salah satu jabatan fungsional	Sistem menampilkan halaman daftar jabatan fungsional dan notifikasi data berhasil dihapus	✓	
	Lihat jabatan fungsional	Mengklik *button* "*show*" pada salah satu jabatan fungsional	Sistem menampilkan halaman data detail jabatan fungsional	✓	
4	Mengelola jabatan struktural				
	struktural	*form* tambah jabatan struktural	jabatan struktural dan notifikasi data berhasil ditambahkan	✓	

	Edit jabatan struktural	Mengklik *button* "edit" pada salah satu data jabatan struktural	Sistem menampilkan *form* edit data jabatan struktural dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus jabatan struktural	Mengklik *button* "hapus" pada salah satu jabatan struktural	Sistem menampilkan halaman daftar jabatan struktural dan notifikasi data berhasil dihapus	✓	
	Lihat jabatan struktural	Mengklik *button* "*show*" pada salah satu jabatan struktural	Sistem menampilkan halaman data detail jabatan struktural	✓	
5	Mengelola data riwayat pendidikan				
	Tambah riwayat pendidikan	Mengklik *button* "simpan" pada *form* tambah riwayat pendidikan	Sistem menampilkan halaman daftar riwayat pendidikan dan notifikasi data berhasil ditambahkan	✓	
	Edit riwayat pendidikan	Mengklik *button* "edit" pada salah satu data riwayat pendidikan	Sistem menampilkan *form* edit data riwayat pendidikan dan menampilkan notifikasi data berhasil disimpan	✓	
	pendidikan	salah satu riwayat pendidikan	riwayat pendidikan dan notifikasi data berhasil dihapus	✓	
	Lihat riwayat pendidikan	Mengklik *button* "*show*" pada salah satu	Sistem menampilkan halaman data detail riwayat pendidikan	✓	

		pangkat golongan	dan notifikasi data berhasil dihapus		
	Lihat pangkat golongan	Mengklik *button* "*show*" pada salah satu pangkat golongan	Sistem menampilkan halaman data detail pangkat golongan	✓	
8	Mengelola mutasi pegawai				
	Tambah mutasi pegawai	Mengklik *button* "simpan" pada *form* tambah mutasi pegawai	Sistem menampilkan halaman daftar mutasi pegawai dan notifikasi data berhasil ditambahkan	✓	
	Edit mutasi pegawai	Mengklik *button* "edit" pada salah satu data mutasi pegawai	Sistem menampilkan *form* edit data mutasi pegawai dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus mutasi pegawai	Mengklik *button* "hapus" pada salah satu mutasi pegawai	Sistem menampilkan halaman daftar mutasi pegawai dan notifikasi data berhasil dihapus	✓	
	Lihat mutasi pegawai	Mengklik *button* "*show*" pada pegawai	Sistem menampilkan halaman data detail	✓	
9	Mengelola data keluarga pegawai				
	Tambah data keluarga pegawai	Mengklik *button* "simpan" pada *form* tambah data keluarga pegawai	Sistem menampilkan halaman daftar data keluarga pegawai dan notifikasi data berhasil ditambahkan	✓	

	Edit data keluarga pegawai	Mengklik *button* "edit" pada salah satu data data keluarga pegawai	Sistem menampilkan *form* edit data keluarga pegawai dan menampilkan notifikasi data berhasil disimpan	✓	
	Hapus data keluarga pegawai	Mengklik *button* "hapus" pada salah satu data keluarga pegawai	Sistem menampilkan halaman daftar data keluarga pegawai dan notifikasi data berhasil dihapus	✓	
	Lihat data keluarga pegawai	Mengklik *button* "*show*" pada salah satu data keluarga pegawai	Sistem menampilkan halaman data detail data keluarga pegawai	✓	
10	Mengelola unit kerja				
	Tambah unit kerja	Mengklik *button* "simpan" pada *form* tambah unit kerja	Sistem menampilkan halaman daftar unit kerja dan notifikasi data berhasil ditambahkan	✓	
	Edit unit kerja	Mengklik *button* "edit" pada salah satu data unit kerja	Sistem menampilkan *form* edit data unit kerja dan menampilkan notifikasi data berhasil disimpan	✓	
	[illegible]	salah satu unit kerja	kerja dan notifikasi data berhasil dihapus	✓	
	Lihat unit kerja	Mengklik *button* "*show*" pada salah satu unit kerja	Sistem menampilkan halaman data detail unit kerja	✓	

11	Menampilkan data pegawai di *tools* Postman	Mengklik *button* "*send*" pada postman	Postman menampilkan data pegawai berupa JSON	✓	
12	Melihat data pegawai berdasarkan NIP/NIK tertentu di Postman	Mengklik *button* "*send*" pada postman	Postman menampilkan data pegawai berdasarkan NIP/NIK berupa JSON	✓	
13	Melihat data master jabatan struktural di Postman	Mengklik *button* "*send*" pada postman	Postman menampilkan data master jabatan struktural berupa JSON	✓	
14	Melihat data master jabatan fungsional di Postman	Mengklik *button* "*send*" pada postman	Postman menampilkan data master jabatan fungsional berupa JSON	✓	
15	Melihat data master unit kerja di Postman	Mengklik *button* "*send*" pada postman	Postman menampilkan data master unit kerja berupa JSON	✓	

Padang, 5 April 2023

Nindy Malisha, S.E

PEMBANGUNAN SISTEM INFORMASI PENGELOLAAN DATA PEGAWAI UNIVERSITAS ANDALAS BERBASIS WEB DENGAN FASILITAS WEB SERVICE

ORIGINALITY REPORT

17%	16%	2%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	scholar.unand.ac.id Internet Source	5%
2	www.scribd.com Internet Source	2%
3	jurnal.atmaluhur.ac.id Internet Source	1%
4	Submitted to Universitas Brawijaya Student Paper	1%
5	repo.unand.ac.id Internet Source	1%
6	123dok.com Internet Source	1%
7	jtk.kodepena.org Internet Source	1%
8	pdfs.semanticscholar.org Internet Source	1%

tunasbangsa.ac.id

9	Internet Source	1 %
10	pdfcoffee.com Internet Source	1 %
11	Submitted to Universitas Andalas Student Paper	1 %
12	ejournal.nusamandiri.ac.id Internet Source	1 %
13	repositori.uin-alauddin.ac.id Internet Source	1 %
14	jhonshonteknik.blogspot.com Internet Source	1 %

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	Off