PENGEMBANGAN TEKNOLOGI *SINGLE SIGN ON* PADA SISTEM INFORMASI DOSEN DAN SISTEM INFORMASI KURIKULUM DI UNIVERSITAS NEGERI JAKARTA

Skripsi

MUHAMAD INSAN RIZKY

5235134409

Skripsi Ini Disusun Sebagai Salah Satu Persyaratan Untuk Memperoleh
Gelar Sarjana Pendidikan

PENDIDIKAN TEKNIK INFORMATIKA DAN KOMPUTER
FAKULTAS TEKNIK
UNIVERSITAS NEGERI JAKARTA
2017

HALAMAN PENGESAHAN

NAMA DOSEN	TANDA TANGAN	TANGGAL
Hamidillah Ajie, S.Si., M.T. (Dosen Pembimbing I)	.....	07 - 08 - 2017
M. Ficky Duskarnaen, M.Sc. (Dosen Pembimbing II)	.....	07 - 08 - 2017

PENGESAHAN PANITIA UJIAN SKRIPSI

NAMA DOSEN	TANDA TANGAN	TANGGAL
Widodo, S.Kom., M.Kom. (Ketua Penguji)	.....	03 - 08 - 2017
Bambang P. Adhi, S.Pd., M.Kom. (Sekretaris Penguji)	.....	03 - 08 - 2017
Vina Oktaviani, S.Pd., M.T (Dosen Penguji Ahli)	.....	03 - 08 - 2017

Tanggal Lulus: 27 - 07 - 2017

HALAMAN PERNYATAAN

Dengan ini peneliti menyatakan bahwa:

1. Karya tulis skripsi peneliti yang berjudul Pengembangan Teknologi *Single Sign-On* Pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum Di Universitas Negeri Jakarta adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik sarjana, baik di Universitas Negeri Jakarta maupun di perguruan tinggi lain;
2. Karya tulis yang berjudul Pengembangan Teknologi *Single Sign-On* Pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum Di Universitas Negeri Jakarta adalah murni gagasan, rumusan, dan penelitian peneliti sendiri dengan arahan dosen pembimbing;
3. Dalam karya tulis, tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka;
4. Pernyataan peneliti dibuat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka peneliti bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini, serta sanksi lainnya sesuai dengan norman yang berlaku di Universitas Negeri Jakarta.

Jakarta, 20 Juli 2017
Yang Membuat Pernyataan

Muhamad Insan Rizky
5235134409

KATA PENGANTAR

Puji dan syukur peneliti panjatkan kehadirat Allah SWT yang telah memberikan rahmat, karunia, dan hidayah-Nya, sehingga peneliti dapat menyelesaikan skripsi dengan judul "Pengembangan Teknologi *Single Sign-On* Pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum Di Universitas Negeri Jakarta", yang merupakan persyaratan untuk meraih gelar Sarjana Pendidikan Teknik Informatika dan Komputer di Fakultas Teknik, Universitas Negeri Jakarta.

Skripsi tidak dapat terwujud dengan baik tanpa adanya bimbingan, dorongan, saran-saran, dan bantuan dari berbagai pihak. Pada kesempatan ini peneleiti ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Ibu Dr. Yuliatri Sastrawijaya, M.Pd. selaku ketua Program Studi Pendidikan Teknik Informatika dan Komputer, Fakultas Teknik, Universitas Negeri Jakarta;
2. Bapak Hamidillah Ajie, S.Si., M.T dan M. Ficky Duskarnaen, M.Sc. selaku dosen pembimbing yang telah penuh kesabaran selalu membimbing dan memberi semangat kepada peneliti hingga skripsi ini selesai;
3. Kedua orang tua, yaitu Bapak Noor Sardono dan Ibu Tati Suryati serta kakak tercinta Minanuryati Akbarina yang senantiasa mendo'akan dan memberikan dorongan selama proses pembuatan skripsi ini;
4. Staff UPT TIK UNJ (Pak Med Irzal, Bang Irfansyah, Mas Arya Adipurwa, Fajar Maulana, M. Nurmansyah Santosa, Hanifa Fissalma, Hendrik Praditya, Nugroho Saputra, Dwi Ramadhian, Yanuar Dwi P., Aditya Nugroho, Septian Ricky P. dan Rendra Andika P.) yang mendukung baik secara teknis maupun non-teknis;
5. Firdaus Ibnu, S.Pd. yang selalu memberikan arahan, wawasan, pengalaman, dan nasihat yang sangat membangun pola pikir yang dewasa dan kritis tidak hanya selama proses pembuatan skripsi tetapi dalam kehidupan sehari-hari;
6. Teman-teman PTIK yang senantiasa membagikan informasi, pengalaman, dan motivasi dalam proses pembuatan skripsi ini;
7. Seluruh pihak yang telah terlibat secara langsung maupun tidak langsung yang tidak bisa disebutkan satu per satu.

Do'a dan harapan semoga kebaikan pihak yang telah disebutkan mendapatkan balasan yang lebih baik dari Allah SWT. Peneliti menyadari bahwa skripsi masih jauh dari kesempurnaan, karenanya peneliti mohon maaf apabila terdapat kekurangan dan kesalahan baik dari isi maupun tulisan. Akhir kata, peneliti berharap semoga skripsi ini dapat bermanfaat bagi yang membacanya.

Jakarta, 27 Juli 2017

Muhamad Insan Rizky

PENGEMBANGAN TEKNOLOGI *SINGLE SIGN-ON* PADA SISTEM INFORMASI DOSEN DAN SISTEM INFORMASI KURIKULUM DI UNIVERSITAS NEGERI JAKARTA

MUHAMAD INSAN RIZKY

ABSTRAK

Fitur otentikasi dan otorisasi merupakan sebuah fitur yang umum diterapkan dalam sistem informasi. Otentikasi merupakan suatu proses untuk mengidentifikasi pengguna yang masuk ke dalam sistem sedangkan otorisasi merupakan proses pemeriksaan *privileges* atau hak istimewa bagi pengguna yang mengakses sistem tersebut sehingga fitur-fitur yang ada dapat disediakan sesuai dengan kebutuhan pengguna. Pada umumnya fitur otentikasi dan otorisasi diterapkan dengan menggunakan proses *login.* Perkembangan teknologi saat ini telah mengantarkan pengguna kepada solusi permasalahan ini yakni dengan menerapkan teknologi yang disebut *Single-Sign-On* (SSO). Dengan teknologi ini, layanan yang terpisah dapat terintegrasi dengan baik dalam sebuah sistem. Saat ini banyak bermunculan *framework* aplikasi yang siap pakai untuk mempermudah pengembang aplikasi dalam mengembangkan aplikasi lainnya. *Framework* Laravel merupakan salah satu *framework* aplikasi yang dikembangkan dengan bahasa PHP. Adapun dalam mengembangkan SSO, Laravel menyediakan fitur bernama Laravel Passport. Aplikasi yang diintegrasikan dengan sistem otentikasi ini adalah Sistem Informasi Dosen dan Sistem Informasi Kurikulum di lingkungan Universitas Negeri Jakarta. Penelitian dilakukan sejak bulan Oktober 2016 hingga Juni 2017. Metode yang digunakan pada pengembangan SSO ini adalah metode *prototyping*. Penelitian bermula dari menganalisa kebutuhan sistem, lalu merancang *prototype* diantaranya merancang *database* dan tampilannya. Setelah itu membuat *prototype* dan mengujinya. Setelah pembuatan *prototype* selesai, *prototype* diterapkan ke dalam sistem yang diintegrasikan dan dilakukan pengujian kembali.

Kata kunci : *Single Sign-On*, Sistem Informasi, *Framework* dan *Prototyping*

SINGLE SIGN-ON TECHNOLOGY DEVELOPMENT IN LECTURERS INFORMATION SYSTEM AND CURRICULUM INFORMATION SYSTEM IN STATE UNIVERSITY OF JAKARTA

MUHAMAD INSAN RIZKY

ABSTRACT

Authentication and authorization feature is a feature that commonly applied in information system. Authentication is a process for identifying users who enter the system while authorization is a privileges checking process or privileges for users accessing the system so that existing features can be provided according to user needs. Generally the authentication and authorization features are applied by using the login process. Current technological developments have led users to this solution by implementing a technology called Single-Sign-On (SSO). With this technology, separate services can integrate well in a system. Currently a lot of emerging framework ready-made applications to simplify application developers in developing other applications. The Laravel framework is one of the application frameworks developed with PHP language. As for developing SSO, Laravel provides a feature called Laravel Passport. Applications that are integrated with this authentication system are Lecturers Information System and Curriculum Information System within the State University of Jakarta. This research was conducted from October 2016 until June 2017. The method used in SSO development is prototyping method. Research starts from analyzing system requirements, then designing prototype such as designing database and appearance. After that create a prototype and test it. After the prototype is complete, the prototype is applied to the integrated system and re-tested.

Keywords : Single Sign-On, Information System, Framework and Prototyping

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Penerapan teknologi informasi di berbagai bidang di sebuah perguruan tinggi menghasilkan berbagai aplikasi yang dikembangkan secara terpisah berdasarkan layanan yang dibutuhkan. Hal ini bertujuan untuk memudahkan pengembang aplikasi dalam mengatasi masalah yang terjadi pada masing-masing aplikasi. Pengembangan aplikasi secara terpisah juga menyebabkan perubahan yang terjadi pada sebuah sistem tidak akan mengganggu proses pada sistem yang lainnya. Jika suatu masalah terjadi pada salah satu aplikasi, maka aplikasi yang lainnya tidak akan mengalami gangguan akibat perubahan pada aplikasi yang bermasalah.

Universitas Negeri Jakarta (UNJ) merupakan salah satu perguruan tinggi yang telah menghasilkan beberapa aplikasi yang dikembangkan secara terpisah. Adapun beberapa sistem informasi di UNJ, antara lain: Sistem Penerimaan Mahasiswa Baru (SIPENMABA), Sistem Informasi Uang Kuliah Tunggal (SIUKAT), Sistem Informasi Akademik (SIAKAD), Sistem Informasi Dosen (SIDOS) dan Sistem Informasi Kurikulum (SIKUR).

SIPENMABA merupakan sistem pendaftaran secara daring untuk penerimaan mahasiswa baru jalur mandiri. Sistem ini mengelola pendaftaran mahasiswa baru mulai dari pengisian data diri, pilihan program studi, pembayaran, hingga penentuan kursi ujian.

Biasanya bagi seluruh mahasiswa baru yang telah dinyatakan lolos seleksi baik melalui jalur SNMPTN, SBMPTN, dan PENMABA Mandiri diwajibkan untuk mengisi data-data yang dibutuhkan oleh SIUKAT untuk menentukan besar Uang Kuliah Tunggal (UKT) yang akan menjadi biaya kuliah bagi mahasiswa tersebut. SIUKAT melakukan penentuan UKT berdasarkan data yang dimasukkan oleh mahasiswa baru berdasarkan rumusan tertentu yang telah disepakati oleh pimpinan UNJ.

SIAKAD merupakan sistem informasi yang mengelola kegiatan akademik seperti pengisian rencana studi, pengisian nilai oleh dosen, informasi jadwal kuliah, dan lain-lain. Mahasiswa dan dosen terlibat aktif dalam sistem ini selama kegiatan akademik berlangsung.

SIDOS merupakan sistem yang mengelola informasi seputar dosen mulai dari data diri dosen, riwayat pendidikan, riwayat mengajar, publikasi ilmiah dan lain-lain. Tujuan dari SIDOS ialah untuk memberikan informasi dosen UNJ secara lebih luas dan lengkap baik kepada masyarakat internal maupun eksternal UNJ.

SIKUR merupakan sistem yang mengelola informasi kurikulum yang sudah, sedang, dan akan digunakan di UNJ berdasarkan masing-masing program studi. Pengguna SIDOS dan SIKUR tidak hanya dosen saja melainkan seluruh civitas akademika UNJ dengan otorisasi yang berbeda untuk setiap jenis penggunanya.

Fitur otentikasi dan otorisasi merupakan sebuah fitur yang umum diterapkan dalam sistem informasi. Otentikasi merupakan suatu proses untuk mengidentifikasi pengguna yang masuk ke dalam sistem sedangkan otorisasi merupakan proses pemeriksaan *privileges* atau hak istimewa bagi pengguna yang mengakses sistem tersebut sehingga fitur-fitur yang ada dapat disediakan sesuai dengan kebutuhan

pengguna. Pada umumnya fitur otentikasi dan otorisasi diterapkan dengan menggunakan proses *login*. Seseorang memasukkan *username* dan *password* (*credential*) kemudian dilakukan otentikasi apakah *credential* tersebut valid atau tidak, jika valid maka seseorang tersebut boleh mengakses sistem tersebut namun jika tidak valid maka dia tidak boleh mengaksesnya. Sebagian besar aplikasi *web* saat ini menggunakan cara tersebut, dengan berbagai tambahan keamanan.

Dalam pelaksanaannya, proses otentikasi dan otorisasi pengguna pada sistem informasi di UNJ dilakukan secara terpisah oleh masing-masing sistem yang tersedia. Setiap sistem menyimpan *credential* dari masing-masing pengguna dengan adanya kemungkinan informasi *credential* tersebut berbeda-beda pada setiap sistem dengan pengguna yang sama. Hal ini dianggap cukup menyulitkan pengguna karena pengguna harus mengirimkan *credential* setiap kali ingin masuk ke dalam sebuah sistem dalam waktu dekat.

Perkembangan teknologi saat ini telah mengantarkan pengguna kepada solusi permasalahan ini yakni dengan menerapkan teknologi yang disebut *Single-Sign-On* (SSO). Menurut Doni Djayusman, SSO adalah sebuah teknologi yang mengizinkan pengguna jaringan agar dapat mengakses beberapa layanan dalam sebuah jaringan hanya dengan menggunakan satu akun pengguna saja. Dengan teknologi ini, layanan yang terpisah dapat terintegrasi dengan baik dalam sebuah sistem. Adapun teknologi yang mendukung penerapan SSO, antara lain: OAuth2, CAS (*Central Authentication Service*), OpenAM (*Open Access Manager*), dan JOSSO (Java *Open Single Sign-On*).

Saat ini banyak bermunculan *framework* aplikasi yang siap pakai untuk mempermudah pengembang aplikasi dalam mengembangkan aplikasi lainnya.

Framework merupakan sebuah *software* untuk memudahkan para pengembang membuat aplikasi atau *web* yang isinya adalah berbagai fungsi, *plugin*, dan kerangka konsep sehingga membentuk suatu sistem tertentu. Dengan menggunakan *framework*, sebuah aplikasi akan tersusun dan terstruktur dengan rapi. Pemilihan sebuah *framework* tentu menjadi kebijakan dari masing-masing pengembang tergantung dari kebutuhan dan kemampuan pengembang tersebut. *Framework* Laravel merupakan salah satu *framework* aplikasi yang dikembangkan dengan bahasa PHP. Adapun dalam mengembangkan SSO, Laravel menyediakan fitur bernama Laravel Passport pada versi 5.3.

Berdasarkan masalah tersebut, maka penelitian tentang Pengembangan Teknologi *Single-Sign-On* pada Sistem Informasi di Universitas Negeri Jakarta perlu dilakukan agar pengelolaan akun pengguna bisa lebih baik dan masing-masing aplikasi dapat terintegrasi dalam sebuah sistem yang sama. Namun penulis hanya berfokus pada SIDOS dan SIKUR sebagai sistem informasi yang akan diintegrasikan dengan teknologi SSO.

1.2. Identifikasi Masalah

Berdasarkan latar belakang, maka dapat diidentifikasi berbagai masalah sebagai berikut:

1. Pengelolaan data akun pengguna dilakukan pada masing-masing sistem informasi sehingga terdapat *redundancy* data;
2. Data akun pengguna yang disimpan secara terpisah mengakibatkan pengguna harus mengingat banyak data akun dan harus memasukkan *credential* setiap kali ingin mengakses suatu sistem.

1.3. Batasan Masalah

Melihat luasnya lingkup permasalahan yang telah diidentifikasi, maka penelitian dibatasi pada:

1. SSO yang dikembangkan dirancang untuk dapat diterapkan pada beberapa aplikasi namun baru akan diuji pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum;
2. SSO dikembangkan dalam bentuk *prototype* karena belum diterapkan secara menyeluruh.

1.4. Rumusan Masalah

Berdasarkan proses latar belakang, identifikasi, dan pembatasan masalah, maka perumusan masalah yang akan dibahas pada penelitian adalah "Bagaimana Mengembangkan Teknologi *Single-Sign-On* pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum di Universitas Negeri Jakarta?"

1.5. Tujuan Penelitian

Berdasarkan perumusan masalah yang telah dirumuskan sebelumnya maka tujuan dari penelitian adalah:

1. Mengintegrasikan pengelolaan akun pengguna pada sistem informasi yang tersedia di UNJ;
2. Meningkatkan kemudahan pengguna dalam mengakses sistem informasi yang tersedia di UNJ.

1.6. Manfaat Penelitian

Kegunaan dari penelitian adalah untuk memudahkan dosen dan mahasiswa UNJ dalam menggunakan berbagai sistem informasi yang tersedia hanya dengan satu akun, serta memudahkan pihak UNJ dalam mengelola peran untuk masing-masing akun sehingga otorisasi menjadi lebih teratur.

BAB II
TINJAUAN PUSTAKA

2.1. Teori

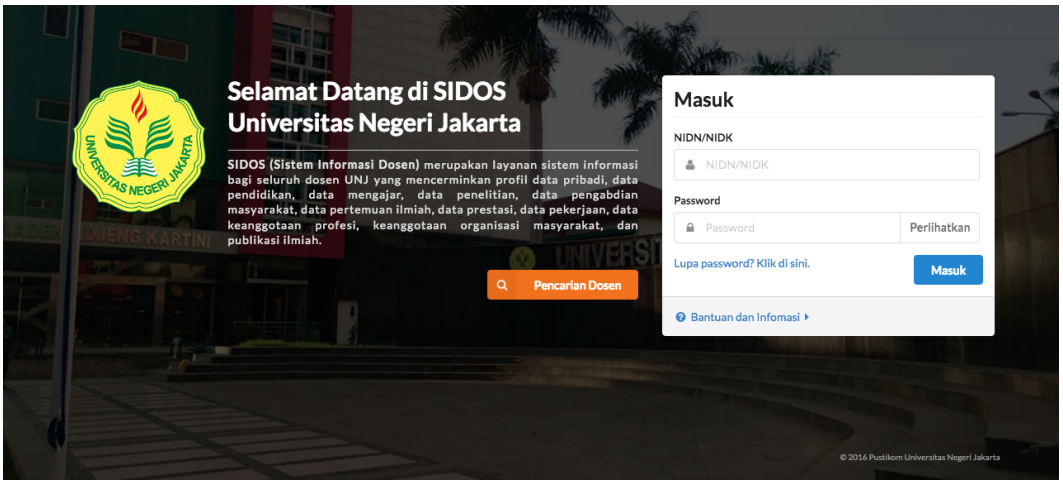
2.1.1. Sistem Informasi di Universitas Negeri Jakarta

Universitas Negeri Jakarta (UNJ) telah menghasilkan beberapa sistem informasi untuk memenuhi kebutuhan yang ada. Sistem informasi tersebut dikembangkan secara terpisah untuk memudahkan proses pengembangannya. Adapun beberapa sistem informasi di UNJ, antara lain: Sistem Penerimaan Mahasiswa Baru (SIPENMABA), Sistem Informasi Uang Kuliah Tunggal (SIUKAT), Sistem Informasi Akademik (SIAKAD), Sistem Informasi Dosen (SIDOS) dan Sistem Informasi Kurikulum (SIKUR).

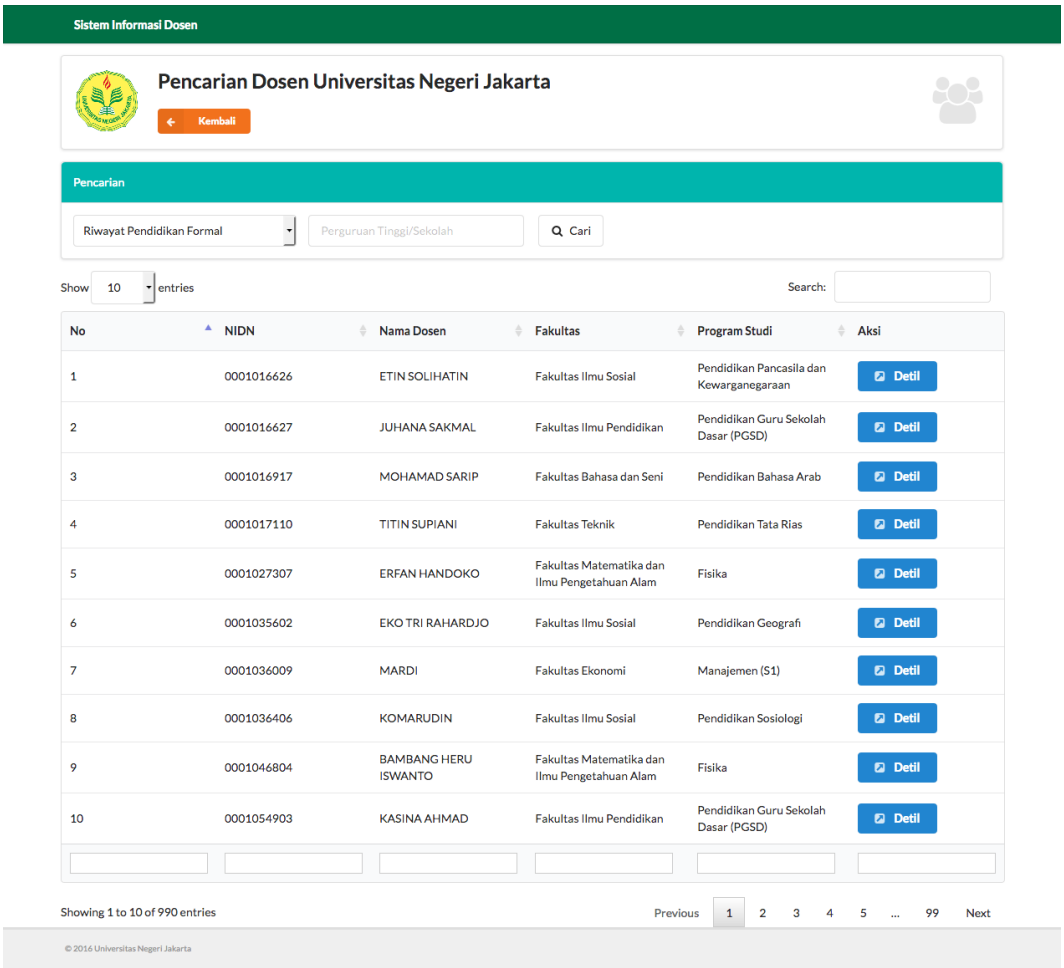
2.1.1.1. Sistem Informasi Dosen

SIDOS merupakan layanan sistem informasi bagi seluruh dosen UNJ yang mencerminkan profil data pribadi, data pendidikan, data mengajar, data penelitian, data pengabdian masyarakat, data pertemuan ilmiah, data prestasi, data pekerjaan, data keanggotaan profesi, keanggotaan organisasi masyarakat, dan publikasi ilmiah. Tujuan dari SIDOS ialah untuk memberikan informasi dosen UNJ secara lebih luas dan lengkap baik kepada masyarakat internal maupun eksternal UNJ. Layanan SIDOS dapat diakses pada http://sidos.unj.ac.id (Sistem Informasi Dosen, 2017).

Adapun tampilan dari SIDOS dapat dilihat pada Gambar 2.1 dan Gambar 2.2:



Gambar 2.1 Halaman *Login* Sistem Informasi Dosen

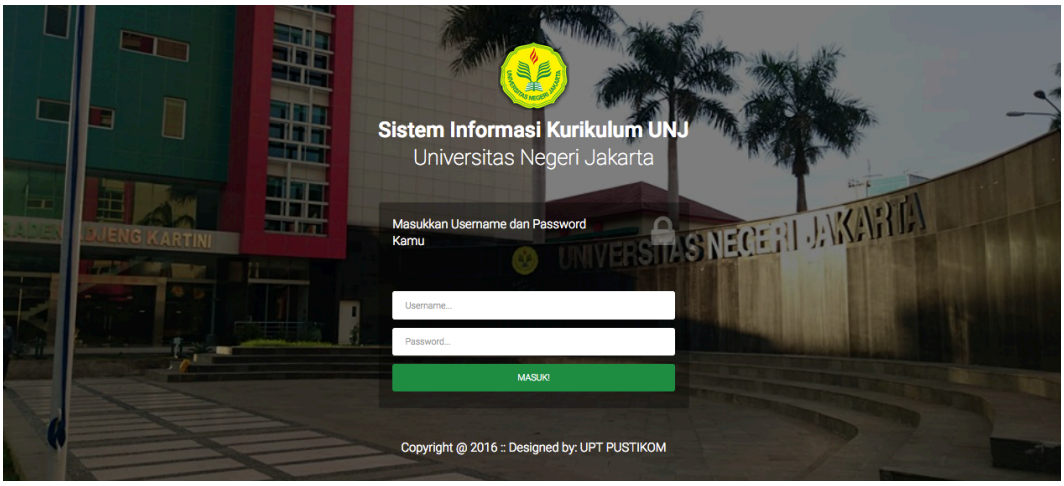


Gambar 2.2 Halaman Pencarian Sistem Informasi Dosen

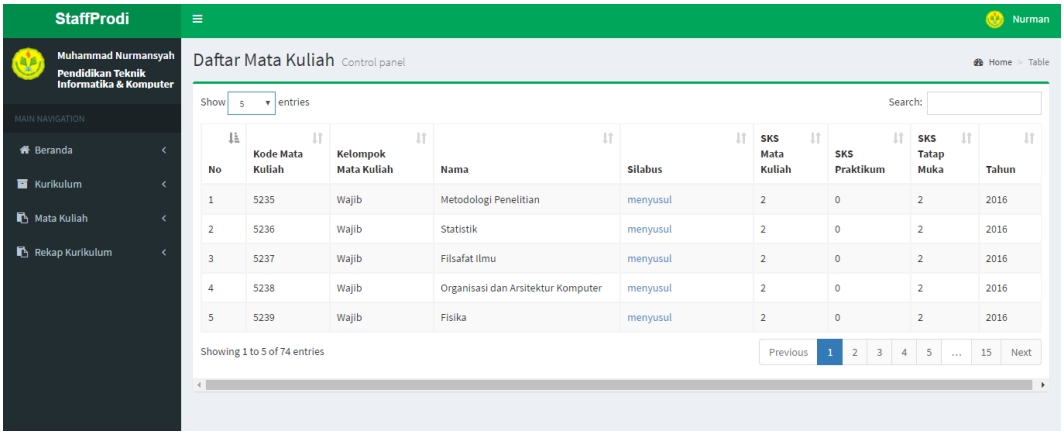
2.1.1.2. Sistem Informasi Kurikulum

SIKUR merupakan layanan sistem informasi yang mengelola informasi kurikulum yang sudah, sedang, dan akan digunakan di UNJ berdasarkan masing-masing program studi. Pengguna SIKUR tidak hanya dosen saja melainkan seluruh civitas akademika UNJ dengan *privileges* yang berbeda untuk setiap jenis penggunanya (UPT TIK, 2017).

Adapun tampilan dari SIKUR dapat dilihat pada Gambar 2.3 dan Gambar 2.4:



Gambar 2.3 Halaman *Login* Sistem Informasi Kurikulum



Gambar 2.4 Halaman Daftar Mata Kuliah Sistem Informasi Kurikulum

2.1.2. *Single Sign On*

Teknologi *Single Sign On* (SSO) adalah sistem yang mengizinkan pengguna agar dapat mengakses seluruh sumber daya dalam jaringan hanya dengan menggunakan satu *credential* saja. Sistem ini tidak memerlukan interaksi yang manual, sehingga memungkinkan pengguna melakukan proses sekali *login* untuk mengakses seluruh layanan aplikasi tanpa berulang kali memasukkan *password* setiap kali memasuki masing-masing aplikasi. Teknologi ini sangat diminati dalam jaringan yang sangat besar dan bersifat heterogen, dimana sistem operasi serta aplikasi yang digunakan berasal dari banyak *vendor*, dan pengguna diminta untuk mengisi informasi dirinya ke dalam setiap *multi-platform* yang hendak diakses (Doni Djayusman, 2013:29-30).

Single Sign On adalah sebuah mekanisme yang membuat *user* hanya perlu mengingat satu *username* dan *password* yang autentik untuk membuka beberapa layanan sekaligus. Sistem *Single Sign On* menghindari *login* ganda dengan cara mengidentifikasi subjek secara ketat dan memperkenankan informasi otentikasi untuk digunakan dalam sistem atau kelompok sistem yang terpercaya. Sistem SSO dapat meningkatkan kegunaan jaringan secara keseluruhan dan pada saat yang sama dapat memusatkan pengelolaan dari parameter sistem yang relevan (Aminudin, 2008:109).

Single Sign On, secara luas dapat dikelompokkan menjadi tiga jenis utama, yaitu *Web Single Sign On, Federated Single Sign On*, dan *Desktop Single Sign On,* tergantung pada jenis aplikasi dan kasus penggunaan yang lazim dalam organisasi. *Web Single Sign On* memberi *Single Sign On* di antara *server* keamanan *web*, dan aplikasi *web* akhir (Buecker, 2012:3).

Web Single Sign On dapat menghilangkan kebutuhan untuk masuk dua kali saat klien mencoba mengakses sumber *web* di *server* yang memerlukan autentikasi dari registrinya sendiri (Buecker, 2012:3). *Federated Single Sign On* menyediakan *Single Sign On* antara aplikasi *web* yang bisa dipercaya dengan menggunakan registri pengguna yang terpisah, biasanya terjadi pada organisasi dengan teman bisnisnya. *Desktop Single Sign On*, biasa disebut *Enterprise Single Sign On* Memungkinkan untuk mulus. *Single Sign On* yang transparan pada aplikasi *desktop* lebih banyak daripada sekedar aplikasi web (Buecker, 2012:3). Tanda tunggal perusahaan memberi pengguna kemampuan untuk masuk dengan satu *password* ke aplikasi klien yang ada (Buecker, 2012:3).

Pengguna layanan dapat lebih menyukai sistem *Single Sign On* dari pada sistem *sign-on* biasa, namun pengelola layanan jaringan memiliki banyak tugas tambahan yang harus dilakukan, seperti perlunya perhatian ekstra untuk menjamin bukti-bukti otentikasi agar tidak tersebar dan tidak disadap pihak lain ketika melintasi jaringan (Aminudin, 2008:110).

Beberapa arsitektur dari sistem SSO telah muncul, masing-masing dengan berbagai keunggulan dan infrastruktur yang berbeda. Pada umumnya sistem SSO (Doni Djayusman, 2013:30) memiliki beberapa keuntungan, antara lain:

1. Pengguna tidak perlu mengingat banyak *username* dan *password*. Cukup dengan satu credential, sehingga pengguna cukup melakukan proses otentikasi sekali saja untuk mendapatkan izin akses terhadap semua layanan aplikasi yang tersedia di dalam jaringan;
2. Kemudahan pemrosesan data. Jika setiap layanan aplikasi memiliki data pengguna masing-masing, maka pemrosesan data pengguna (penambahan,

pengurangan, perubahan) harus dilakukan pada setiap aplikasi yang ada. Sedangkan dengan menggunakan sistem SSO, cukup hanya melakukan sekali pemrosesan pada *server database backend*-nya. Hal ini menyatakan bahwa penggunaan sistem SSO meningkatkan efisiensi waktu dan kepraktisan dalam memproses data;

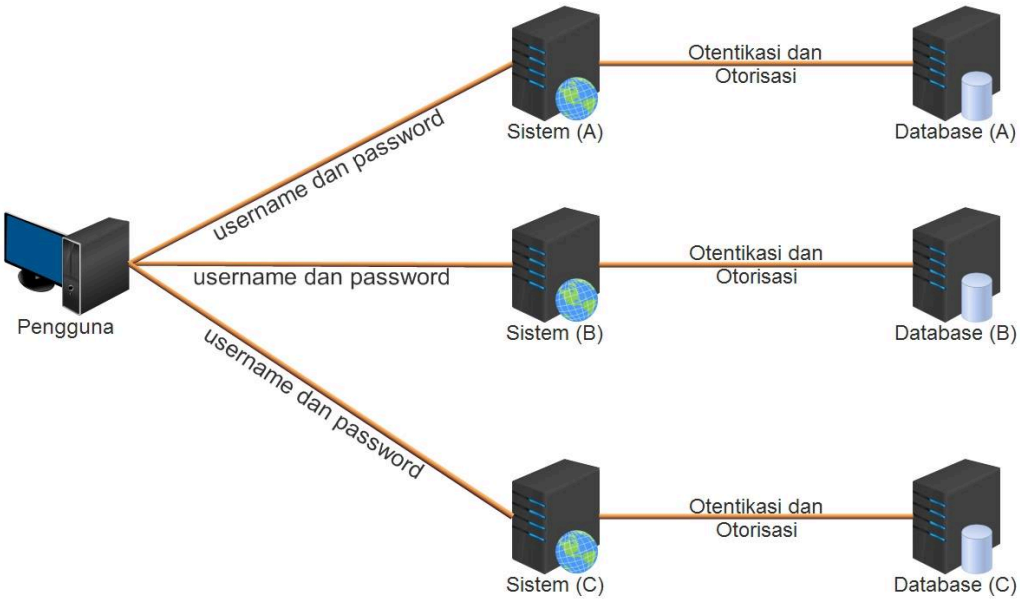
3. Tidak perlu membuat data pengguna yang sama di setiap aplikasi karena setiap layanan aplikasi dalam jaringan dapat terhubung langsung dengan *server database backend* ini, maka hanya dengan sekali saja menginput data kedalam *database*, *credential* pengguna akan valid di seluruh layanan aplikasi;
4. Menghemat biaya untuk pemeliharaan *password*. Ketika harus me-*reset password* karena pengguna lupa pada *password*-nya, pengelola layanan tidak perlu menghabiskan waktu dan bandwith untuk menemukan data *credential* pengguna.

Selain mendatangkan manfaat, sistem SSO (Doni Djayusman, 2013:30-31) juga dapat mendatangkan kerugian, antara lain:

1. Pentingnya kesadaran pengguna untuk merahasiakan data *credential* dan menjaga keadaan *login*-nya. Bila masih dalam keadaan login, pengguna yang tidak sah dapat memakai mesin yang ditinggalkan pengguna sahnya;
2. Kerumitan mengimplementasikan sistem SSO kedalam sebuah jaringan yang heterogen dan *multiplatform*, sehingga banyak pengelola layanan jaringan kurang begitu giat dalam mengimplementasikannya;

3. Kelemahan dalam hal keamanan. Jika *password* sistem pengelola layanan jaringan diketahui oleh orang yang tidak berhak, maka orang tersebut dapat melakukan perubahan terhadap semua data yang ada di dalam sistem.

Terdapat perbedaan proses pada teknologi *single sign on* dan *sign on* pada umumnya. Alur proses *sign on* (Ramadhan,Gilang, 2012:16) dapat dilihat pada Gambar 2.5:

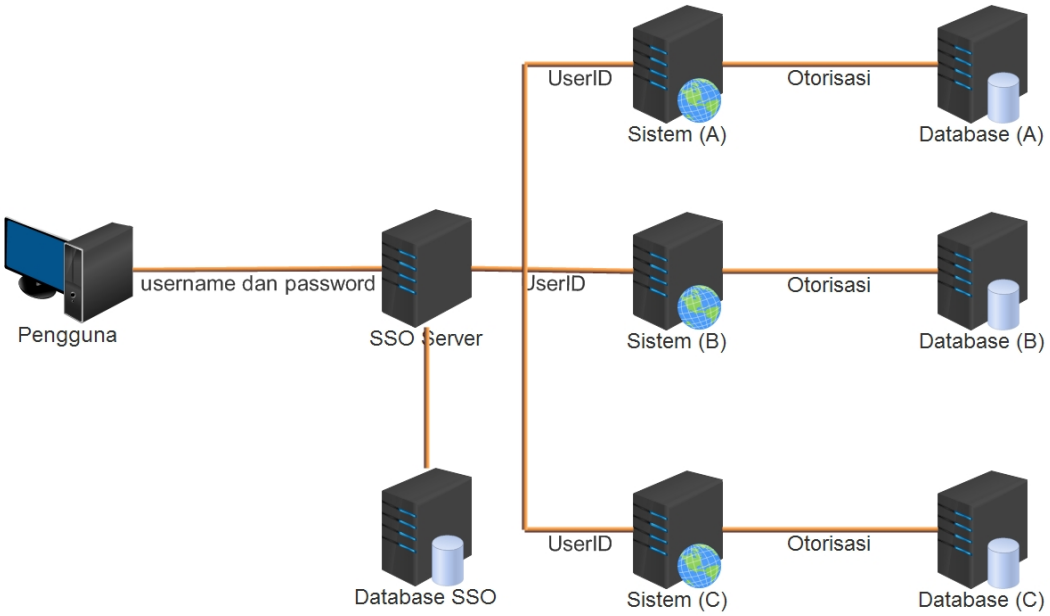


Gambar 2.5 Alur Sign On Pada Umumnya

Pada Gambar 2.5 merupakan alur dari *sign on* yang diterapkan pada sistem informasi pada umumnya. *User* akan memasukkan *username* dan *password* setiap kali masuk ke sistem yang akan diakses kemudian sistem tersebut akan melakukan otentikasi dan otorisasi pada *database* yang terhubung dengan sistem tersebut. Setiap sistem hanya memerlukan satu kali otentikasi untuk masing-masing *user* namun proses otentikasi ini dilakukan berkali-kali pada sistem yang berbeda karena

data pengguna disimpan dan dikelola oleh masing-masing sistem. Dalam hal ini, data akun pengguna pada sistem A mungkin berbeda dengan data akun pengguna pada sistem B meskipun data akun tersebut dimiliki oleh orang yang sama.

Adapun alur proses *Single Sign On* (Ramadhan,Gilang,2012:17) dapat dilihat pada Gambar 2.6:



Gambar 2.6 Alur Single Sign On

Pada Gambar 2.6 merupakan alur dari *Single Sign On* dengan pendekatan Federasi. Seluruh identitas *user* disimpan pada *database* SSO namun untuk *role user* pada masing-masing sistem disimpan pada masing-masing *database* sistem. *User* hanya akan memasukkan *username* dan *password* sebanyak satu kali, kemudian SSO *Server* akan melakukan otentikasi pada *database* SSO. Setelah lolos otentikasi, SSO akan meneruskan sejenis identitas *user* yang dalam hal ini disebut sebagai *User ID* atau *token* sebagai tiket ke masing-masing sistem yang diakses oleh *user* tersebut. Selanjutnya sistem akan melakukan otorisasi untuk mengetahui

role dari *User ID* yang telah diberikan oleh SSO *Server* sehingga sistem tersebut dapat menyediakan fitur yang sesuai dengan kebutuhan *user* tersebut.

2.1.2.1. Jenis Pendekatan SSO

Secara umum SSO diimplementasikan sebagai sebuah model otentikasi yang independen yang mana seluruh aplikasinya menggunakan modul otentikasi berbasis SSO untuk mengesahkan pengguna. Ketika pengelola layanan memilih untuk mengaplikasikan sistem SSO, maka dapat dipilih salah satu dari tiga pendekatan SSO (Abdurrahman,Luthfi, 2012:9-10) berikut:

1. Pendekatan Terpusat (*Centralized Approaches*): Pendekatan ini mempunyai sebuah lokasi yang terpusat dimana seluruh identifikasi disimpan. *Server* SSO bertindak sebagai perantara untuk mendistribusikan identifikasi ketika dibutuhkan dan sampai pada otentikasi dan otorisasi pengguna jika diperlukan. Biasanya hal ini membutuhkan pergantian aplikasi untuk mengintegrasikan dengan *server* SSO. Teknologi pada Microsoft's .NET *Passport* menggunakan pendekatan ini;
2. Pendekatan Distribusi (*Distributed Approaches*): Pendekatan ini mengizinkan kumpulan pernyataan identifikasi yang dilokalisasikan dari tiap-tiap aplikasi dengan diteruskan menggunakan komponen *client-based* (berbasis klien). Oleh karena itu pengguna memiliki kendali penuh terhadap komponen klien dan profil/*password* disinkronkan ke pusat *server*. Satu dari keuntungan pendekatan ini adalah bahwa jika seorang penyerang (*attacker*) mendapatkan akses informasi dari sistem, ini hanya akan menemukan akses ke informasi dari *database* yang sedang berjalan dari sistem tersebut;

3. Pendekatan Federasi (*Federation Approches*): Pendekatan ini menyediakan identifikasi terpusat dan layanan manajemen otentikasi yang sejalan dengan kumpulan pernyataan identifikasi yang dilokalisasikan. Hampir seluruh dari produk dan arsitektur SSO yang sekarang berdasarkan pada model ini. Pendekatan ini menyediakan manfaat yang paling besar dari kedua pendekatan terpusat dan distribusi.

2.1.2.2. Arsitektur SSO

Solusi sistem SSO didasarkan pada salah satu dari dua tingkat pendekatan, yaitu pendekatan *script* dan pendekatan *agent*. *Agent* merupakan sebuah program kecil yang berjalan pada tiap-tiap web server. Agent ini membantu mengkoordinir aliran kerja dari sistem SSO dalam hal otentikasi pengguna dan penanganan sesi (Abdurrahman,Luthfi ,2012:11-12).

Arsitektur *Sistem* SSO (Abdurrahman,Luthfi ,2012:13) memiliki dua bagian utama, yaitu agent yang berada di web server/layanan aplikasi dan sebuah server SSO berdedikasi yang mana akan dijelaskan berikut ini:

1. *Agent*: Sebuah agent akan menterjemahkan setiap permintaan HTTP yang masuk ke *web server*. Hanya ada satu *agent* di tiap-tiap *web server*, yang mana *host* bagi layanan aplikasi. *Agent* tersebut akan berinteraksi dengan *web browser* pada sisi pengguna, dan dengan *server* SSO pada sisi layanan aplikasi.
2. SSO *server*: *Server* SSO menggunakan *cookies* temporer (sementara) untuk menyediakan fungsi manajemen sesi. Sebuah *cookies* terdiri dari informasi seperti *user-id*, *session-id*, *session creation time*, *session expiration time* dan lain-lain.

Produk-produk sistem SSO yang berbasis *open source* yang umum digunakan saat ini seperti OAuth2, CAS (*Central Authentication Service*), OpenAM (*Open Access Manager*), dan JOSSO (*Java Open Single Sign-On)*.

2.1.3. Framework

Framework dalam bahasa Indonesia memiliki arti yaitu kerangka kerja. Secara istilah, *framework* dapat diartikan sebagai kumpulan dari *library* (*class*) yang dapat diturunkan atau dapat langsung dipakai fungsinya oleh modul-modul atau fungsi yang akan kita kembangkan (Septian,2011:7).

Adapun beberapa keuntungan (Octafian,Tri,2015:2) dari menggunakan *framework,* antara lain sebagai berikut:

1. Waktu pembuatan *website* jauh lebih singkat. Kode aplikasi *website* menjadi lebih mudah dibaca, karena sedikit dan sifatnya pokok;
2. *Website* menjadi lebih mudah diperbaiki karena tidak perlu fokus ke semua komponen *website*, terutama kode *system framework;*
3. Tidak perlu lagi membuat kode penunjang aplikasi *website* seperti koneksi *database*, validasi *form*, GUI dan keamanan;
4. Pikiran menjadi lebih terfokus terhadap kode alur permasalahan *website*, apa yang ditampilkan dan layanan apa saja yang diberikan dari aplikasi tersebut;
5. Jika dikerjakan secara tim, maka akan lebih terarah karena system framework mengharuskan pengguna menggunakan keteraturan peletakan kode.

Sedangkan kekurangan (Octafian,Tri, 2015:2) dari menggunakan *framework*, antara lain:

1. Beberapa *programmer* mungkin akan kesulitan saat merancang apilikasi dengan keterbatasan yang dimiliki suatu *framework;*
2. Adanya tambahan biaya dalam membangun aplikasi yang digunakan oleh *programmer* karena *framework* tersebut tidak *support*.

Beberapa contoh fungsi-fungsi standar yang telah tersedia dalam suatu *framework* adalah fungsi *paging*, enkripsi, *email*, SEO, *session*, *security*, kalender, bahasa, manipulasi gambar, grafik, tabel bergaya zebra, validasi borang, *upload*, *captcha*, proteksi terhadap XSS(XSSfiltering), template, kompresi, XML dan lain-lain. Adapun beberapa *framework* yang cukup populer saat ini, antara lain: Laravel, CodeIgniter, Zend, Spring, Express Js, dan lain-lain.

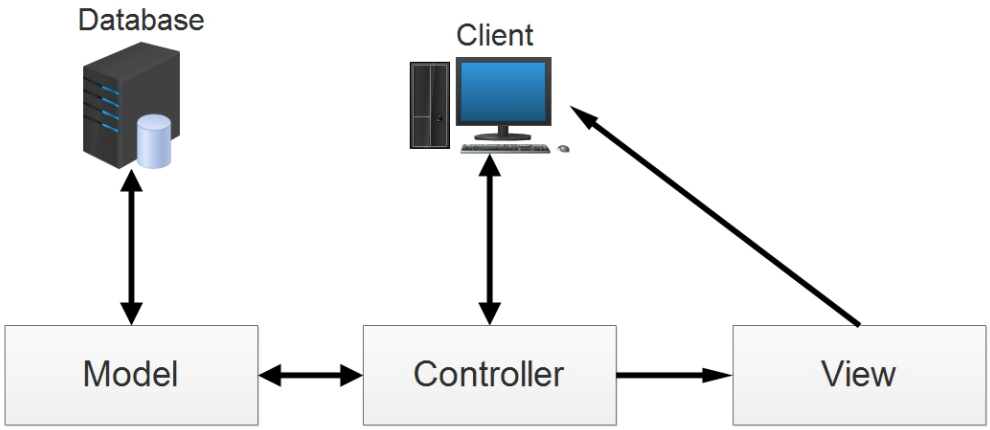
2.1.3.1. Laravel

Laravel adalah *framework* PHP dengan kode terbuka (*open source*) dengan desain MVC (*Model-View-Controller*) yang digunakan untuk membangun aplikasi *website*. *Framework* ini pertama kali dibangun oleh Taylor Otwell pada tanggal 22 Pebruari 2012 (Tim Air Putih, 2014:1).

Laravel adalah sebuah *framework* PHP yang dirilis dibawah lisensi MIT, dibangun dengan konsep MVC (*Model View Controller*). Laravel adalah pengembangan website berbasis MVC yang ditulis dalam PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan awal dan biaya pemeliharaan, dan untuk meningkatkan pengalaman

bekerja dengan aplikasi dengan menyediakan sintaks yang ekspresif, jelas dan menghemat waktu (IdCloudHost, 2016).

MVC adalah sebuah pendekatan perangkat lunak yang memisahkan aplikasi logika dari presentasi. MVC memisahkan aplikasi berdasarkan komponen-komponen aplikasi, seperti: manipulasi data, *controller*, dan *user interface*. Model merupakan penghubung aplikasi dengan *database* yang merepresentasikan struktur data dan berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain. *View* merupakan bagian yang mengatur tampilan aplikasi yang berhadapan langsung dengan pengguna (IdCloudHost, 2016). Sedangkan *Controller* merupakan bagian yang menjembatani *model* dan *view*. Alur MVC dapat dilihat lebih jelas pada Gambar 2.7:



Gambar 2.7 ***Model-View-Controller Mapping***

Penjelasan Gambar 2.7 adalah ketika *user* melakukan *request website* ke *web server*, maka pertama kali yang dijalankan adalah *file controller*, kemudian *file controller* ini akan mengecek apakah memerlukan *database* atau tidak, jika iya maka rute selanjutnya adalah *controller* memanggil *model*. M*odel* melakukan pengolahan *database* lalu mengembalikan hasilnya ke dalam *controller*.

Selanjutnya *controller* akan memecah hasil dari *model* ke dalam *view* dan ditampilkan ke *user*.

Terdapat beberapa fitur (IdCloudHost,2016) yang disediakan oleh *framework* Laravel, antara lain:

1. *Bundles*, yaitu sebuah fitur dengan sistem pengemasan modular dan tersedia beragam di aplikasi;
2. *Eloquent* ORM, merupakan penerapan PHP lanjutan menyediakan metode internal dari pola "*active record*" yang menagatasi masalah pada hubungan objek *database*;
3. *Application Logic*, merupakan bagian dari aplikasi, menggunakan *controller* atau bagian *Route*;
4. *Reverse Routing*, mendefinisikan relasi atau hubungan antara *Link* dan *Route*;
5. *Restful controllers*, memisahkan logika dalam melayani *method* HTTP, seperti: GET, POST, DELETE, dan PUT;
6. *Class Auto Loading*, menyediakan *loading* otomatis untuk *class* PHP;
7. *View Composer*, adalah kode unit logikal yang dapat dieksekusi ketika *view* sedang *loading*;
8. *IoC Container*, memungkin obyek baru dihasilkan dengan pembalikan *controller*;
9. *Migration*, menyediakan sistem kontrol untuk skema *database*;
10. *Unit Testing*, banyak tes untuk mendeteksi dan mencegah regresi;
11. *Automatic Pagination*, menyederhanakan tugas dari penerapan halaman.

Sama seperti *framework* lainnya, Laravel pun memiliki beberapa kelebihan, antara lain sebagai berikut:

1. Menggunakan PHP versi terbaru: dalam PHP versi 5.6 ke atas memiliki cukup banyak fitur baru yang membuat PHP lebih full dan modern;
2. Ekspresif. Ketika melihat suatu *syntax* Laravel, seorang *programmer* diharapkan akan langsung tahu kegunaan dari *syntax* tersebut meskipun belum pernah mempelajarnya apalagi menggunakannya;
3. *Simple*. Yang membuat Laravel *simple* adalah adanya fitur *Elequent* ORM. Fitur ini merupakan fitur yang akan menyederhanakan proses *query* ke *database*;
4. *Security*. Aplikasi yang akan dibangun benar-benar aman dari masalah *security* dasar, seperti: pengamanan csrf, autentikasi, sanitasi data, validasi data dan lain-lain. Dan juga bisa didapatkan secara gratis.

Di samping kelebihan yang sudah dijelaskan, Laravel pun memiliki beberapa kekurangan, antara lain:

1. Ukuran *framework* cukup besar dibandingakan dengan php *native* dikarenakan ada beberapa berkas sebagai *third-party* yang mendukung fitur-fitur Laravel;
2. Memiliki jumlah *through put* yang lebih besar daripada *framework* lainnya.

2.1.3.2. Laravel Passport

Seiring dengan berkembangnya kebutuhan para pengembang aplikasi di dunia industri menyebabkan Laravel terus meningkatkan fitur-fitur pada *framework*-nya. Banyak pengembang yang membuat *module* yang bersifat *open*

source yang kompatibel dengan *framework* Laravel. Salah satu fitur terbaru dari Laravel versi 5.3 adalah Laravel Passport. Fitur ini diadopsi dari *module* yang bernama *League Oauth2 Server* yang dikembangkan oleh Alex Bilbie pada akun Github-nya. Laravel Passport menyediakan implementasi OAuth2 *Server* secara penuh untuk aplikasi Laravel (Laravel *Documentation*).

OAuth 2.0 adalah protokol standar industri untuk otorisasi. OAuth 2.0 menggantikan kerja yang dilakukan pada protokol OAuth asli yang dibuat pada tahun 2006. OAuth 2.0 berfokus pada klien pengembang kesederhanaan sambil memberikan mengalir otorisasi khusus untuk aplikasi web, aplikasi desktop, ponsel, dan perangkat ruang tamu. Spesifikasi ini sedang dikembangkan dalam IETF OAuth WG.

a. **Laravel Passport *Database***

Module Laravel Passport sudah mengakomodasi struktur *database* yang dibutuhkan untuk membuat sistem otentikasi SSO. Pengembang hanya perlu menghubungkan Laravel dengan *database* yang akan digunakan kemudian melakukan migrasi *database* dengan memasukkan perintah pada *command line* seperti yang telah dijelaskan pada dokumentasi resminya (Laravel *Documentation*). Namun, Laravel Passport hanya mengatasi struktur *database* untuk SSO secara umum. Hal ini tentu sangat mendukung para pengembang untuk melakukan penyesuaian tergantung kebutuhan dari sistem yang akan dibuat.

2.1.4. *Cookie*

Cookie adalah serangkaian teks yang disimpan pada komputer oleh situs *web* yang dikunjungi. Pada umumnya *cookie* menyimpan pengaturan atau preferensi

Anda untuk suatu situs *web* tertentu, misalnya bahasa yang dipilih, atau lokasi (negara). Ketika kembali ke situs *web* tersebut, *web browser* akan mengirimkan *cookie* yang bersesuaian kepada situs *web* yang bersangkutan. Dengan cara ini, situs dapat menampilkan informasi yang sesuai dengan pengaturan atau preferensi pengguna (Mozilla).

Cookie dapat menyimpan berbagai jenis informasi, termasuk di antaranya informasi pribadi seperti nama, alamat rumah, alamat *email*, atau nomor telepon. Akan tetapi informasi ini hanya akan disimpan jika pernah memberikan informasi ini kepada situs tersebut. Situs *web* tidak dapat mengakses informasi yang tidak pernah Anda berikan kepada situs *web* tersebut, dan situs *web* juga tidak dapat mengakses berkas lainnya pada komputer (Mozilla).

Secara bawaan (*default*), aktivitas menyimpan dan mengirim *cookie* tidak pernah terlihat oleh pengguna. Akan tetapi, Anda dapat mengubah pengaturan *web browser* sehingga dapat mengizinkan atau menolak permintaan penyimpanan *cookie*, menghapus *cookie* yang tersimpan saat *web browser* ditutup, dan lain sebagainya (Mozilla).

2.1.5. HTTP dan HTTPS

HTTP singkatan dari *Hypertext Transfer Protocol*. Bila memasukkan *HTTP://* di *address bar* di depan *domain*, ia memberitahu *browser* untuk menghubungkan melalui HTTP. HTTP menggunakan TCP (*Transmission Control Protocol*), umumnya di port 80, untuk mengirim dan menerima paket data melalui *web*. Sederhananya ini adalah protokol yang digunakan oleh klien dan *server* yang memungkinkan Anda untuk berkomunikasi dengan situs-situs lain. Klien

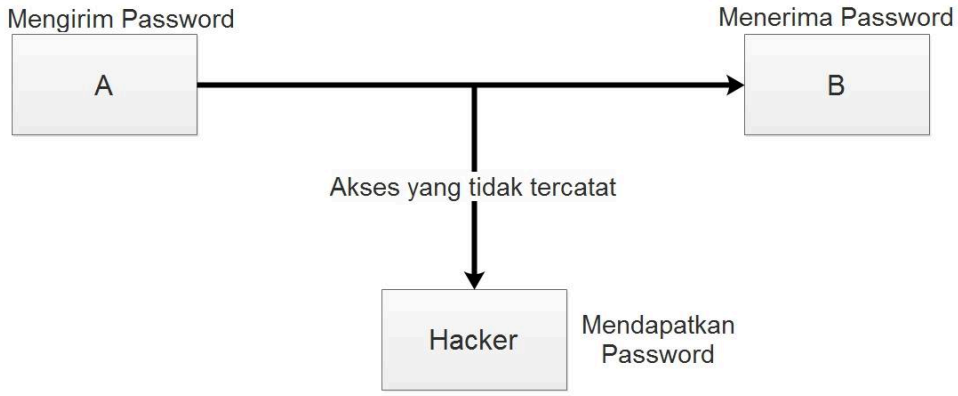
mengirimkan pesan permintaan ke server HTTP (setelah jabat tangan TCP) yang *host* sebuah situs *web*, *server* kemudian menjawab dengan pesan respon. Pesan respon berisi informasi status penyelesaian, seperti *"HTTP/1.1 200 OK"* (Keycdn).

TCP telah memiliki perangkat tambahan selama bertahun-tahun tetapi untuk sebagian besar sangat banyak yang sama seperti itu ketika pertama kali didefinisikan pada tahun 1974, RFC 675. HTTP juga menggunakan UDP (*User Datagram Protocol*), yang dirancang oleh David Reed pada tahun 1980, didefinisikan dalam RFC 768. Hal ini kurang handal tetapi banyak digunakan dalam konferensi video, video game, dan streaming. Hal ini memungkinkan paket-paket individu yang akan turun dan diterima dalam urutan yang berbeda untuk kinerja yang lebih baik. *Hypertext* istilah yang awalnya berasal dari Ted Nelson pada tahun 1965. HTTP asli dikembangkan dan awalnya diusulkan oleh Tim Berners-Lee, direktur *World Wide Web Consortium* (W3C). Misi W3C adalah untuk memimpin *web* secara maksimal dengan mengembangkan protokol dan pedoman yang memastikan pertumbuhan jangka panjang dari *web* (Keycdn).

Dokumentasi pertama HTTP diterbitkan pada tahun 1991 sebagai HTTP / 0.9 yang hanya terdiri dari satu HTTP metode permintaan, GET (data permintaan dari sumber daya yang ditentukan). Pada tahun 1996 HTTP 1.0, RFC 1945, dikembangkan dan ini terdiri dari metode permintaan tiga HTTP, GET, HEAD, dan POST (menyampaikan data yang akan diproses ke sumber daya tertentu). Akhirnya pada tahun 1997, HTTP / 1.1 protokol, RFC 2068, dikembangkan sebagai revisi HTTP 1.0 dan setelah 19 tahun itu masih digunakan hari ini untuk semua permintaan HTTP.

Selama bertahun-tahun ada telah beberapa revisi sedikit untuk HTTP / 1.1. Pada tahun 1999, RFC 2616 memperkenalkan 5 metode baru, PILIHAN, PUT, TRACE, CONNECT, dan DELETE. Dan kemudian pada bulan Maret 2010, RFC 5789 memperkenalkan metode PATCH. Sampai saat ini versi saat ini mendefinisikan 9 metode permintaan yang berbeda.

Dalam HTTP / 0.9 dan 1.0 sambungan ditutup setelah satu permintaan. Dalam HTTP / 1.1 bertahan koneksi (lebih dari satu permintaan / respon pada koneksi HTTP yang sama) diperkenalkan, yang secara dramatis mengurangi latency. perbaikan lainnya seperti *caching*, dukungan kompresi yang lebih baik, dan *Cross-Origin Resource Sharing* (CORS) juga ditambahkan.



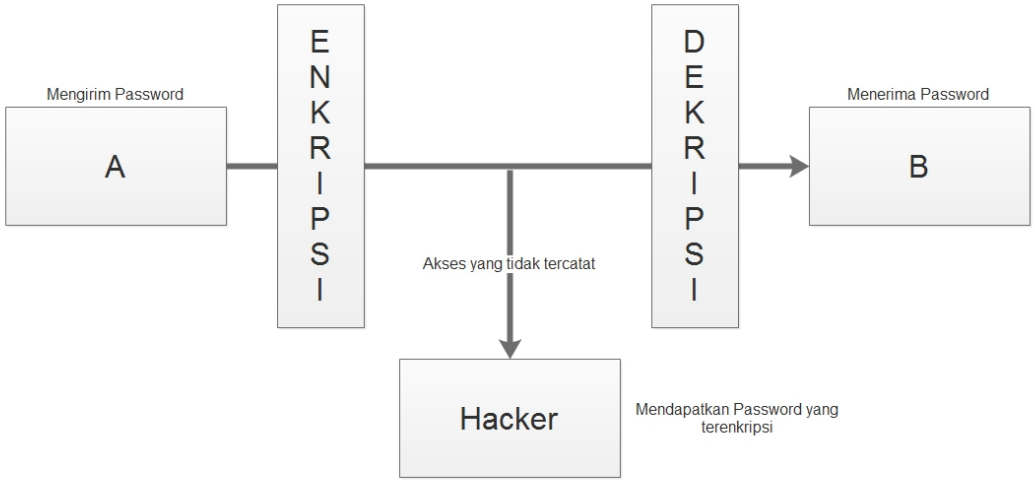
Gambar 2.8 Cara kerja HTTP

Pada Gambar 2.8, HTTP digambarkan tidak memiliki kemanan yang terjamin. Saat proses pentransferan *password*, *Hacker* dapat dengan mudah mengambil ditengah jalan proses pentransferan tanpa diketahui oleh pihak pengirim maupun penerima *password*.

Sedangkan HTTPS adalah singkatan dari *Hypertext Transfer Protocol Secure* (juga disebut sebagai HTTP melalui TLS atau HTTP melalui SSL). Bila Anda memasukkan HTTPS: // di *address bar* di depan *domain*, ia memberitahu *browser*

untuk menghubungkan melalui HTTPS. Umumnya situs berjalan HTTPS akan memiliki *redirect* di tempat sehingga bahkan jika mengetik di HTTP: // itu akan mengarahkan untuk memberikan lebih dari sambungan aman. HTTPS juga menggunakan TCP (*Transmission Control Protocol*) untuk mengirim dan menerima paket data, tetapi ia melakukannya di port 443, dalam koneksi dienkripsi dengan *Transport Layer Security* (TLS) (Keycdn).

HTTPS mentransmisikan keamanan data dengan menggunakan koneksi terenkripsi. Pada dasarnya ini menggunakan kunci publik yang kemudian didekripsi pada sisi penerima. Kunci publik digunakan pada *server*, dan termasuk dalam apa yang Anda ketahui sebagai sertifikat SSL. Sertifikat ini kriptografi ditandatangani oleh Otoritas Sertifikat (CA), dan masing-masing *browser* memiliki daftar CA secara implisit percaya. Setiap sertifikat yang ditandatangani oleh CA dalam daftar dipercaya diberikan kunci gembok hijau di *address bar browser*, karena itu terbukti "terpercaya" dan milik domain tersebut. Perusahaan seperti *Mari Encrypt* sekarang telah membuat proses penerbitan sertifikat SSL gratis.



Gambar 2.9 Cara kerja HTTPS

Pada Gambar 2.9, HTTPS digambarkan memiliki kemanan yang terjamin. Saat proses pentransferan *password*, terdapat proses enkripsi-dekripsi password. *Hacker* dapat dengan mudah mengambil ditengah jalan proses pentransferan , namun hanya mendapatkan *password* yang sudah terenkripsi.

Berikut adalah beberapa perbedaan utama antara HTTP dan HTTPS protokol:

1. URL HTTP di *address bar browser* adalah *HTTP://* dan URL HTTPS adalah *HTTPS://*.
2. HTTP tanpa jaminan sementara HTTPS dijamin.
3. HTTP mengirimkan data melalui port 80 sedangkan HTTPS menggunakan port 443.
4. HTTP beroperasi pada lapisan application, sementara HTTPS beroperasi pada lapisan transport.
5. Tidak ada sertifikat SSL yang diperlukan untuk HTTP, HTTPS memerlukan sertifikat SSL dan ditandatangani oleh CA.
6. HTTP tidak memerlukan domain validasi, di mana sebagai HTTPS membutuhkan setidaknya validasi domain dan sertifikat tertentu bahkan memerlukan validasi dokumen hukum.

2.1.5.1. SSL dan TLS

Transport Layer Security (TLS) protokol, *Secure Sockets Layer* (SSL) protokol, versi 2.0 dan 3.0, didasarkan pada kriptografi kunci publik. Keamanan Channel (Schannel) protokol otentikasi *Suite* menyediakan protokol ini. Semua protokol Schannel menggunakan model *client* / *server* (Microsoft).

Dalam proses otentikasi, klien TLS / SSL mengirimkan pesan ke server TLS / SSL, dan *server* merespon dengan informasi bahwa *server* perlu mengotentikasi sendiri. Klien *dan server* melakukan pertukaran tambahan kunci sesi, dan dialog otentikasi berakhir. Ketika otentikasi selesai, komunikasi SSL aman dapat mulai antara server dan klien menggunakan kunci enkripsi simetris yang dibentuk selama proses otentikasi (Microsoft).

SSL dikembangkan oleh *Netscape Communications Corporation* pada tahun 1994 *untuk* mengamankan transaksi melalui *World Wide Web*. Segera setelah itu, *Internet Engineering Task Force* (IETF) mulai bekerja untuk mengembangkan protokol standar yang menyediakan fungsi yang sama. Mereka menggunakan SSL 3.0 sebagai dasar untuk pekerjaan itu, yang menjadi protokol TLS (Microsoft).

TLS dan SSL paling luas diakui sebagai protokol yang menyediakan HTTP aman (HTTPS) untuk transaksi Internet antara *Web browser* dan *server Web*. TLS/SSL juga dapat digunakan untuk protokol tingkat aplikasi lainnya, seperti *File Transfer Protocol* (FTP), *Lightweight Directory Access Protocol* (LDAP), dan *Simple Mail Transfer Protocol* (SMTP). TLS / SSL memungkinkan otentikasi *server*, otentikasi klien, enkripsi data, dan integritas data melalui jaringan seperti *World Wide Web* (Microsoft).

2.1.6. Model *Prototyping*

Prototyping adalah proses iteratif dalam pengembangan sistem dimana kebutuhan diubah ke dalam sistem yang bekerja (*working system*) secara terus menerus diperbaiki melalui kerjasama antara pengguna dan analis. *Prototype* juga bisa dibangun melalui beberapa *tool* pengembangan untuk menyederhanakan

proses. *Prototyping* merupakan bentuk dari *Rapid Application Development* (RAD) (Fatta,2007:36).

Prototyping merupkaan metode pengembangan lebih efektif dan efisien dalam memperbaiki dan mengoptimalkan produk yang dibuat melalui diskusi, eksplorasi, percobaan dan perbaikan secara berulang-ulang sesuai dengan keinginan pengguna (Argawal,dkk., 2010:42).

Idealnya, *prototype* berfungsi sebagai mekanisme mengidentifikasi kebutuhan perangkat lunak. Jika mengembangkan perangkat lunak dengan metode ini, maka pengembang menggunakan penggalan program dari *tools* yang siap menjalankan program yang dihasilkan dengan cepat.

Secara umum, pengembangan prototype meliputi 3 sub proses yang dilakukan secara iteratif (Pressman, 2012:31):

1. Mendengarkan pelanggan

   Tahap ini dilakukan pengumpulan kebutuhan dari sistem dengan cara mendengarkan keluhan dari pelanggan. Untuk membuat sistem yang sesuai kebutuhan, maka harus diketahui terlebih dahulu bagaimana sistem yang sedang berjalan untuk kemudian mengetahui masalah yang terjadi;

2. Merancang dan membuat *prototype*

   Pada tahap ini, dilakukan perancangan dan pembuatan *prototype* sistem. *Prototype* yang dibuat disesuaikan dengan kebutuhan sistem yang telah didefinisikan sebelumnya dari keluhan pengguna;

3. Uji coba

   Pengembangan *prototype* pada tahap ini dari sistem diuji coba oleh pengguna, kemudian dilakukan evaluasi kekurangan dari kebutuhan pengguna.

Pengembangan kemudian kembali mendengarkan keluhan untuk memperbaiki *prototype* yang ada.

Metode *prototyping* memiliki beberapa kelebihan, antara lain sebagai berikut:

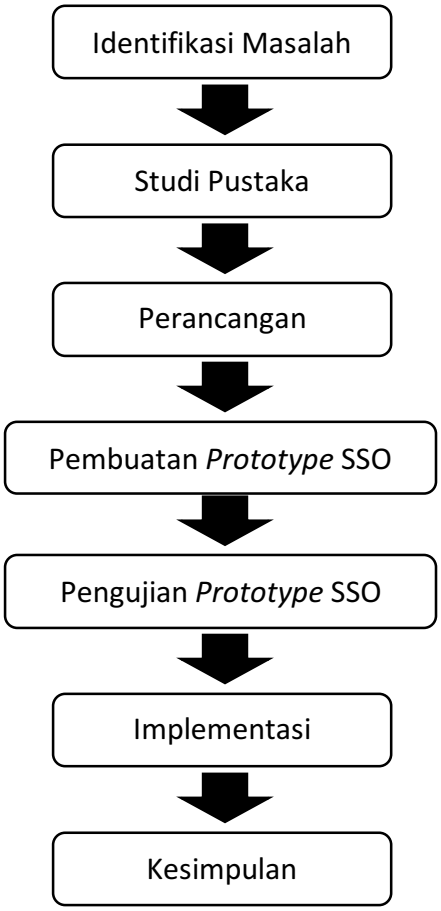
1. Adanya komunikasi yang baik antara pengembang aplikasi dan pengguna;
2. Pengembang aplikasi dapat bekerja lebih baik dalam menentukan kebutuhan pengguna;
3. Pengguna berperan aktif dalam pengembangan sistem;
4. Lebih menghemat waktu dalam pengembangan sistem;
5. Penerapan menjadi lebih mudah karena pengguna mengetahui apa yang diharapkannya.

Adapun kekurangan dari metode *prototyping* adalah sebagai berikut:

1. Pengguna tidak melihat bahwa perangkat lunak belum mencerminkan kualitas perangkat lunak secara keseluruhan dan belum memikirkan pemeliharaan dalam jangka waktu yang lama;
2. Pengembang biasanya ingin cepat menyelesaikan proyek sehingga menggunakan algoritma dan bahasa pemograman sederhana;
3. Hubungan pengguna dengan komputer mungkin tidak menggambarkan teknik perancangan yang baik.

2.2. Prosedur Penelitian

Prosedur penelitian yang dilakukan penulis dapat dilihat pada Gambar 2.10:



Gambar 2.10 Prosedur Penelitian

Berdasarkan identifikasi masalah yang telah dijelaskan pada Bab I, langkah berikutnya adalah melakukan studi pustaka dari penelitian yang sudah ada atau dari sumber terpercaya lainnya termasuk proses pengumpulan data. Tahap berikutnya adalah merancang sistem yang akan dikembangkan yaitu merancang *database* dan *service*. Merancang *database* dan *service* bertujuan untuk memetakan tabel mana saja yang akan terhubung dan *service* apa saja yang nantinya akan saling berhubungan. *Service* ini meliputi *service* pada SSO *server* dan *service* pada

aplikasi yang akan diintegrasikan dengan SSO *server*. Setelah merancang, langkah selanjutnya adalah pembuatan *prototype*. Pembuatan *prototype* adalah langkah membuat *database* dan *service* yang sebelumnya sudah dirancang. Langkah selanjutnya adalah melakukan pengujian *prototype* dengan membuat skenario pengujian yang tepat dan menerapkannya pada *prototype* yang sudah dibuat. Setelah pengujian *prototype* selesai, selanjutnya adalah mengimplementasikan *prototype* ke dalam sistem. Proses implementasi *prototype* ini dilakukan pada SIDOS dan SIKUR UNJ. Kemudian diadakan pengujian kembali, yaitu dengan melihat keberhasilan *prototype* yang diterapkan ke dalam sistem. Jika sudah berjalan dengan baik, maka langkah terakhir adalah menarik kesimpulan.

BAB III
METODE PENELITIAN

3.1. Tempat dan Waktu Penelitian

Penelitian dilakukan di Unit Pelaksana Teknis Teknologi Informasi dan Komunikasi Universitas Negeri Jakarta. Waktu penelitian dilaksanakan sejak bulan Oktober 2016 hingga Juli 2017.

3.2. Alat dan Bahan Penelitian

3.2.1. Alat Penelitian

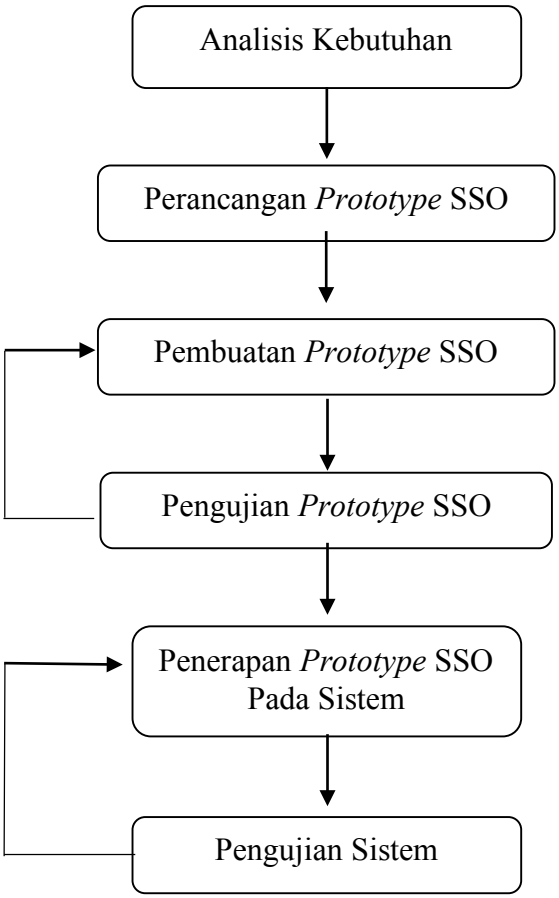
1. Perangkat Keras
    a. 2.5GHz *dual-core* Intel Core i5 *processor* (*Turbo Boost up to* 3.1GHz) *with* 3MB L3 *cache*
    b. 2.5GHz 4GB of 1600MHz DDR3 *memory*
    c. 2.5GHz 500GB 5400-rpm *hard drive*
2. Perangkat Lunak
    a. Sistem Operasi MacOS Sierra versi 10.12.3
    b. Visual Studio Code
    c. XAMPP untuk macOS 7.0.13-1
    d. Google Chrome
    e. Internet

3.2.2. Bahan Penelitian

Bahan yang digunakan pada penelitian adalah sekumpulan data akun pengguna yang diperoleh dari UPT TIK (Unit Pelaksana Teknis Teknologi Informasi dan Komunikasi) UNJ dengan berbagai macam *role* yang sesuai dengan kebutuhan SIDOS dan SIKUR.

3.3. Diagram Alir Penelitian

Aliran perancangan dan pengembangan SSO pada SIDOS dan SIKUR dapat dilihat pada Gambar 3.1:



Gambar 3.1 Diagram Alir Penelitian

Metode yang digunakan pada pengembangan SSO ini adalah metode *prototyping*. Penulis menggunakan metode ini karena untuk mempercepat proses

pengembangan sistem sesuai dengan kebutuhan yang dijelaskan oleh pengguna sehingga apabila terjadi perubahan *system environment*, pengembangan teknologi SSO dapat segera beradaptasi dengan perubahan tersebut.

Adapun penggunaan metode *prototyping* pada penelitian adalah sebagai berikut:

1. Analisis Kebutuhan

   Pada tahap ini, penulis mendefinisikan kebutuhan apa saja yang akan dimuat pada SSO berdasarkan variasi *role user* yang ada di lingkungan UNJ berdasarkan penjelasan dari pengguna.

2. Perancangan *Prototype* SSO

   Setelah menganalisis kebutuhan, penulis melakukan perancangan sistem meliputi perancangan algoritma SSO dan perancangan *database*. Penulis menentukan pendekatan SSO yang tepat terlebih dahulu kemudian membuat algoritma SSO yang sesuai dengan pendekatan SSO dan kebutuhan kasus ini. Meskipun *database* telah diakomodasi oleh Laravel Passport, perancangan *database* tetap dilakukan untuk menyesuaikan SSO yang disediakan oleh Laravel Passport dengan kebutuhan yang telah dianalisis sebelumnya khususnya perancangan struktur tabel *user* untuk pengelolaan otorisasi.

3. Pembuatan *Prototype* SSO

   Pada tahap ini penulis membuat sistem SSO berdasarkan perancangan yang telah dibuat sebelumnya. Tahap pembuatan *prototype* ini dimaksudkan sebagai usaha untuk mewujudkan hasil dari perancangan perangkat lunak.

4. Pengujian *Prototype* SSO

   Pengujian dilakukan dengan metode *black box* untuk menganalisis kualitas dari *prototype* yang telah dibuat. Apabila hasil pengujian sudah mencapai kategori baik dan sesuai dengan kebutuhan, maka penelitian dilanjutkan ke tahap berikutnya. Namun apabila hasil pengujian masih belum cukup, maka kembali lagi ke tahap pembuatan *prototype* untuk memperbaiki kekurangan yang ditemukan pada saat pengujian.

5. Implementasi *Prototype* ke Sistem

   Setelah *prototype* lolos tahap pengujian, maka selanjutnya adalah mengimplementasikan *prototype* ke dalam sistem yang sudah berjalan yaitu SIDOS dan SIKUR. Penerapan *prototype* ini menyebabkan perlu adanya penyesuaian antara sistem yang sudah berjalan dengan model *prototype* SSO.

6. Pengujian Sistem

   Tahap terakhir yaitu melakukan pengujian pada sistem yang sudah diterapkan SSO berdasarkan *prototype* yang dibuat sebelumnya. Hal ini untuk memastikan bahwa *prototype* memang sudah berfungsi sepenuhnya dan berjalan dengan baik pada sistem sebenarnya.

3.3.1. Analisis Kebutuhan

3.3.1.1. Daftar Kebutuhan Fungsional

Tahap ini merupakan tahap awal untuk menemukan kebutuhan umum yang akan dikembangkan menjadi *prototype* SSO. Fungsional ini dibagi menjadi dua bagian, yaitu: 1) Fungsional Utama; dan 2) Fungsional Admin. Fungsional utama merupakan serangkaian fungsi utama dari otentikasi SSO terhadap masing-masing

aplikasi yang terintegrasi sedangkan fungsional admin merupakan fungsi yang tidak akan muncul pada pengguna umum dan hanya akan digunakan untuk admin SSO saja.

Adapun daftar kebutuhan fungsional tersebut dapat dilihat pada Tabel 3.1 dan Tabel 3.2:

Tabel 3.1 Daftar Fungsional Utama

No.	Deskripsi Fungsional
1	Mengarahkan pengguna ke halaman *login* apabila pengguna mengakses aplikasi *client* tanpa memiliki *token*
2	Memverifikasi *credential* yang dimasukkan oleh pengguna dengan data pengguna di *database*
3	Mengarahkan pengguna ke halaman *callback* ketika sedang melakukan verifikasi *credential*
4	Memverifikasi *secret key* dan *id* aplikasi *client* yang dikirim dari aplikasi dengan data yang ada di *database*
5	Menghasilkan *token* dan mengirimkannya ke aplikasi *client*
6	Mengatur *lifetime token* yang diberikan ke aplikasi *client*
7	Memverifikasi *token* apabila pengguna mengakses aplikasi yang berbeda ketika sudah melakukan *login*
8	Menonaktifkan *token* yang dikirim pengguna sebagai proses *logout* dari seluruh aplikasi

Tabel 3.2 Daftar Fungsional *Admin*

No.	Deskripsi Fungsional
1	Mengarahkan tampilan ke halaman panel admin apabila pengguna menekan tombol menu panel admin
2	Menampilkan daftar aplikasi *client* yang sudah terdaftar
3	Menampilkan *form* untuk pengisian aplikasi *client*
4	Menyimpan data aplikasi *client* baru
5	Menghasilkan *secret key* dan *id* aplikasi *client* yang baru ditambahkan
6	Menampilkan data aplikasi *client* ketika *admin* menekan tombol ubah pada daftar aplikasi *client*
7	Menghapus data aplikasi *client* ketika *admin* menekan tombol hapus pada daftar aplikasi *client* tersebut
8	Mengarahkan tampilan ke halaman data pengguna apabila *admin* menekan tombol menu data pengguna
9	Menampilkan daftar data pengguna yang tersimpan di *database*
10	Menampilkan *form* untuk pengisian data pengguna
11	Menyimpan data pengguna baru

12	Menampilkan data pengguna ketika admin menekan tombol ubah pada daftar data pengguna
13	Menghapus data pengguna ketika *admin* menekan tombol hapus pada data pengguna tersebut
14	Mengarahkan tampilan ke domain aplikasi tujuan yang dipilih oleh pengguna

3.3.1.2. Daftar *Role* Pengguna

Pada umumnya, sebuah sistem memiliki pengguna dengan *role* yang berbeda-beda untuk masing-masing pengguna. *Role* pengguna adalah sebuah status pengguna atau *privileges* yang mendeskripsikan peran pengguna tersebut pada sistem. Fitur yang diberikan oleh sistem akan berbeda-beda berdasarkan *role* pengguna yang sedang aktif. Adapun daftar *role* pengguna pada SSO dapat dilihat pada Tabel 3.3:

Tabel 3.3 Daftar *Role* Pengguna

No.	Aplikasi	Nama *Role*
1	Sistem Informasi Dosen	Dosen
2	Sistem Informasi Kurikulum	Dosen
3	Sistem Informasi Kurikulum	Staff Program Studi
4	Sistem Informasi Kurikulum	Ketua Program Studi
5	Sistem Informasi Kurikulum	Staff Wakil Rektor I
6	Sistem Informasi Kurikulum	*Admin* MKU
7	Sistem Informasi Kurikulum	*Admin* MKDK
8	Sistem Informasi Kurikulum	*Admin* MKP
9	Sistem Informasi Kurikulum	Pejabat Fakultas
10	Sistem Informasi Kurikulum	*Admin*

3.3.2. Perancangan *Prototype* SSO

3.3.2.1. Rancangan *Database*

Perancangan *database* dilakukan sebagai gambaran keadaan struktur data yang digunakan pada SSO. Pada dasarnya Laravel Passport telah menyediakan struktur *database* bawaan untuk mengatur proses otentikasi namun penyesuaian struktur *database* dinilai tetap perlu dilakukan sehingga penulis

melakukan modifikasi *database* untuk menyesuaikan SSO dengan lingkungan pengembangan. *Database* yang disediakan Laravel Passport berisi 6 tabel yang sudah saling berkaitan satu sama lain dalam menangani proses otentikasi SSO.

Adapun desain tabel-tabel tambahan yang akan digunakan dalam pengembangan SSO adalah sebagai berikut:

Tabel 3.4 Tabel Referensi Fakultas

No.	Nama Kolom	Tipe Data	Keterangan
1	kode	INT (2)	NOT NULL (PK)
2	nama	VARCHAR (255)	NOT NULL

Tabel 3.5 Tabel Referensi Program Studi

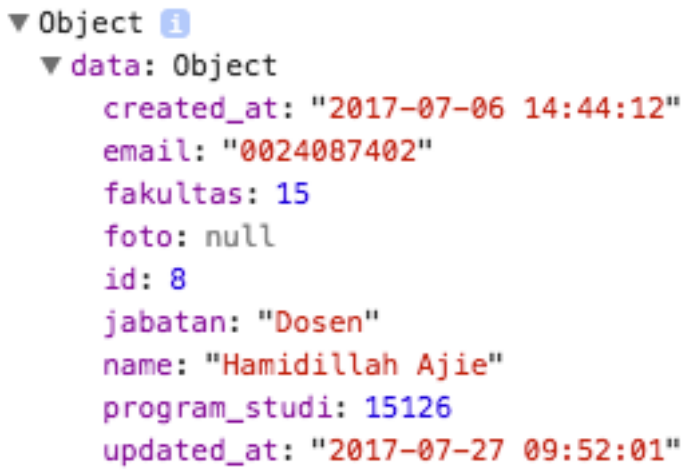
No.	Nama Kolom	Tipe Data	Keterangan
1	kode	VARCHAR (5)	NOT NULL (PK)
2	jalur	INT (1)	NOT NULL
3	nama	VARCHAR (255)	NOT NULL
4	jenjang	VARCHAR (2)	NOT NULL
5	kode_fak	INT (1)	NOT NULL
6	kode_lama	VARCHAR (4)	NOT NULL

Tabel 3.6 Tabel *Users*

No.	Nama Kolom	Tipe Data	Keterangan
1	kode	INT (10)	AUTO_INCREMENT (PK)
2	name	VARCHAR (255)	NOT NULL
3	email	VARCHAR (255)	NOT NULL (UK)
4	password	VARCHAR (255)	NOT NULL
5	jabatan	VARCHAR (255)	NULL
6	foto	TEXT	NULL
7	fakultas	INT (11)	NULL
8	program_studi	INT (11)	NULL
9	remember_token	VARCHAR (100)	NULL
10	created_at	TIMESTAMP	NULL
11	updated_at	TIMESTAMP	NULL

Pada dasarnya Tabel 3.6 merupakan tabel bawaan dari Laravel Passport, namun penulis melakukan penambahan kolom, yaitu: jabatan, foto, fakultas, dan program_studi sebagai pelengkap identitas yang dimiliki oleh setiap pengguna.

Selain *database* yang terdapat pada SSO, terdapat susunan tabel khusus yang harus ada pada masing-masing aplikasi. Tujuan dari adanya tabel khusus tersebut adalah untuk menyesuaikan proses otorisasi yang dilakukan oleh masing-masing aplikasi dengan data pengguna yang diberikan oleh SSO ketika sudah berhasil *login* dan memperoleh *token*. Struktur data pengguna yang diberikan oleh SSO dapat dilihat pada Gambar 3.2:



Gambar 3.2 Struktur Data Pengguna Dari SSO

Dari data pengguna yang dikirim dari SSO seperti Gambar 3.2, selanjutnya aplikasi akan melakukan otorisasi dengan cara mengambil *role* pengguna berdasarkan *email* data pengguna. *Email* tersebut sebenarnya merupakan identitas unik pada masing-masing data pengguna yang memungkinkan berisi data selain *email* melainkan nidn, nim, nip, ataupun *username*. Sedangkan tabel khusus yang terdapat pada aplikasi dapat dilihat pada Tabel 3.7 dan Tabel 3.8:

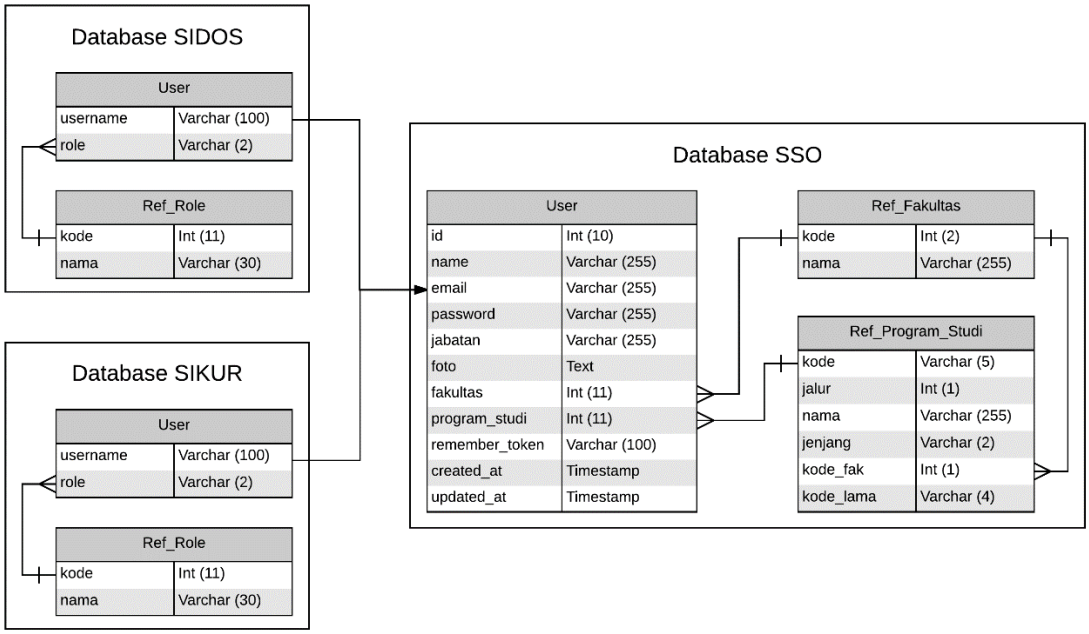
Tabel 3.7 Tabel *User* Pada *Database* Aplikasi

No.	Nama Kolom	Tipe Data	Keterangan
1	username	VARCHAR (100)	NOT NULL (UK)
2	role	VARCHAR (2)	NOT NULL

Tabel 3.8 Tabel *Role* Pada *Database* Aplikasi

No.	Nama Kolom	Tipe Data	Keterangan
1	kode	INT (10)	AUTO_INCREMENT (PK)
2	nama	VARCHAR (30)	NOT NULL

Tabel 3.7 dan Tabel 3.8 merupakan bentuk minimal dari struktur tabel yang dimiliki oleh aplikasi sehingga pengembang aplikasi dapat menambahkan kolom pada tabel-tabel tersebut. Secara garis besar, hubungan antara *database* SSO dengan *database* aplikasi dapat dilihat pada Gambar 3.3:



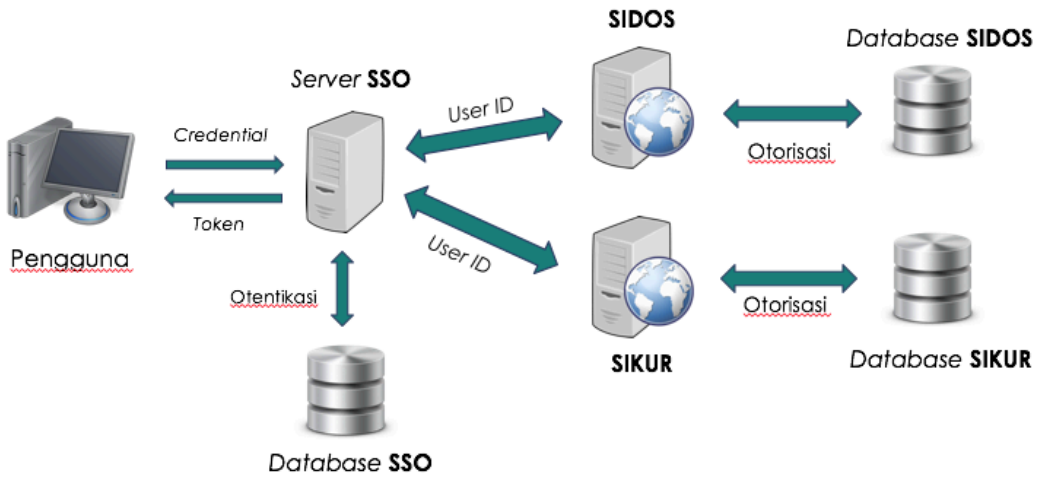
Gambar 3.3 Hubungan Antara *Database* SSO dengan *Database* SIDOS dan *Database* SIKUR

Terdapat 3 *container* pada Gambar 3.3, yaitu: *database* SSO, *database* SIDOS, dan *database* SIKUR. Pada *database* SSO terdapat 3 tabel yang akan berkaitan dengan *database* SIDOS dan SIKUR. Namun sebenarnya, hanya 1 tabel yang benar-benar berkaitan secara langsung dengan SIDOS dan SIKUR, yaitu: tabel *user*. Ketika pengguna sudah berhasil memperoleh *token* dari SSO, maka *token* tersebut dapat digunakan sebagai *tiket* untuk memperoleh informasi

pengguna yang sedang aktif. Data *user* yang dikirim dari SSO berupa *json* seperti pada Gambar 3.2. Dari *json* itulah hubungan SSO dengan SIDOS dan SIKUR terjadi. SIDOS dan SIKUR hanya perlu mencari *username* pada tabel *user* mereka berdasarkan data *json* dari SSO.

3.3.2.2. Rancangan *Prototype* SSO

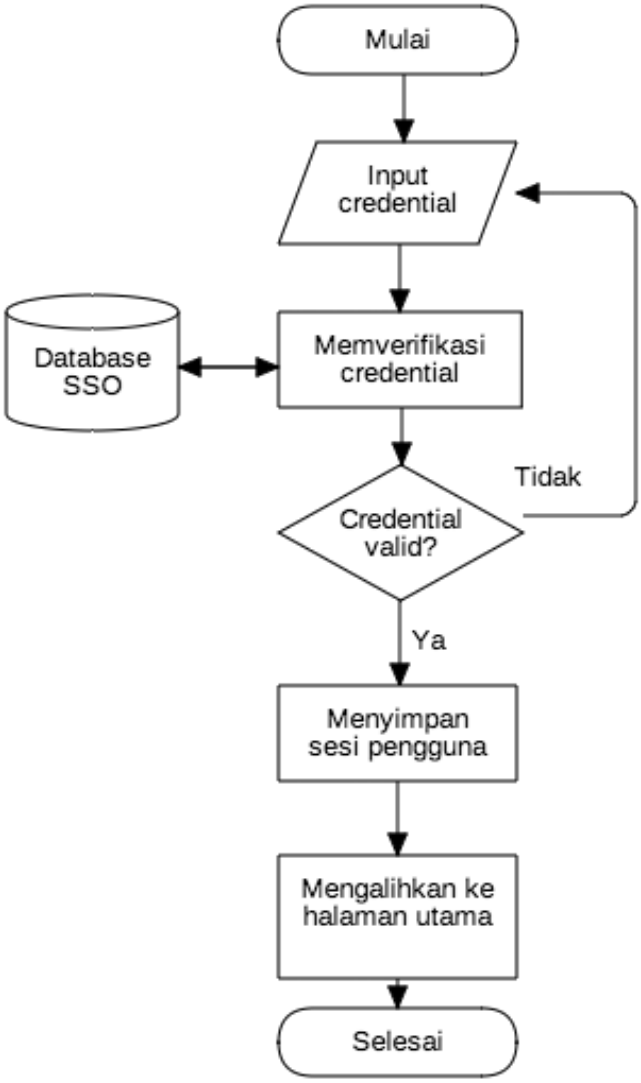
Setelah melakukan perancangan *database*, tahap selanjutnya adalah melakukan perancangan *prototype* SSO dengan membuat *model* dan *flowchart. Model* digunakan sebagai gambaran alur proses kerja otentikasi SSO secara keseluruhan sedangkan *Flowchart* dibuat sebagai gambaran alur dari cara kerja *prototype* SSO secara lebih rinci.



Gambar 3.4 *Model Single Sign-On*

Pada Gambar 3.4 menjelaskan alur proses SSO secara keseluruhan. Diawali dengan pengguna yang memasukkan *credential* ke SSO kemudian *credential* tersebut akan diverifikasi dengan data yang ada pada *database* SSO. Jika *credential*

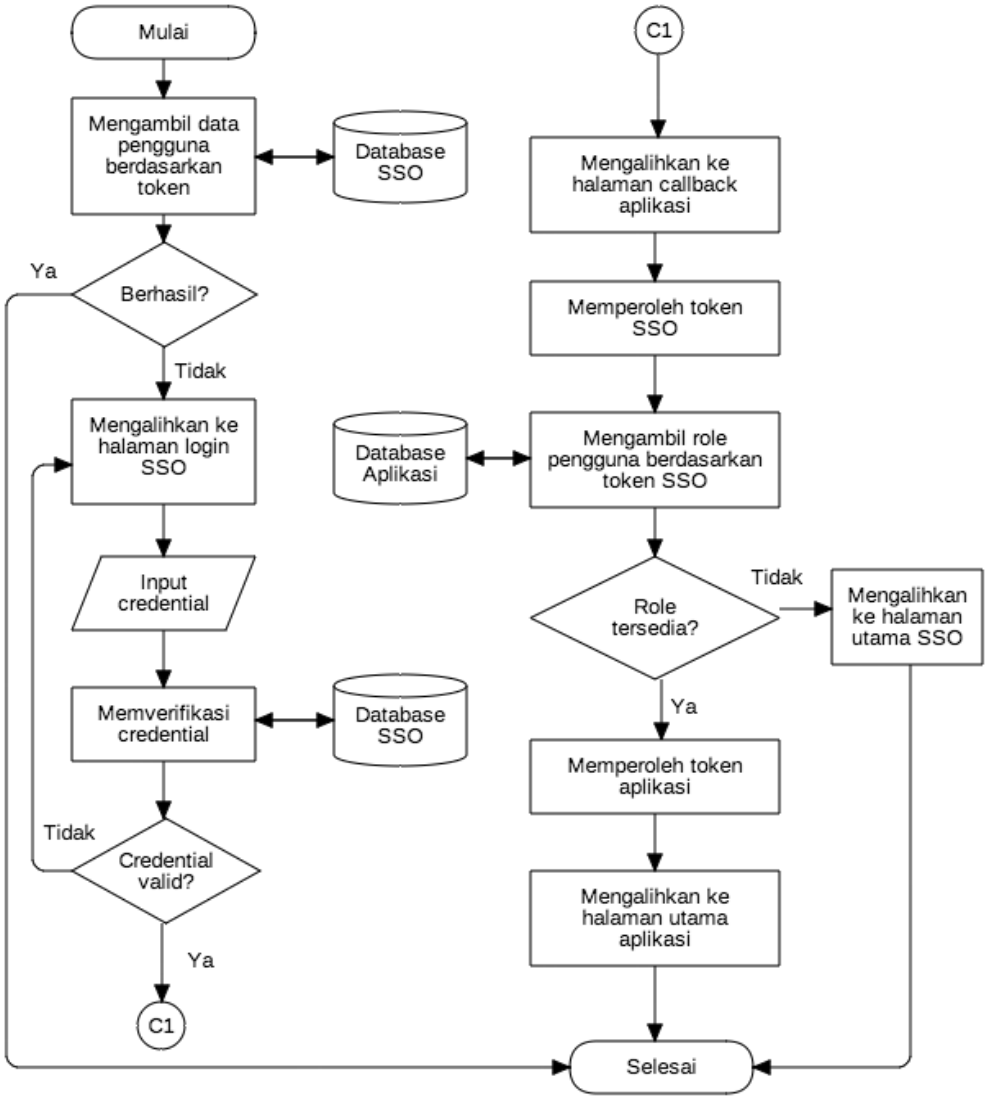
tersebut telah terverifikasi, maka SSO akan mengembalikan *token* kepada pengguna dan *token* tersebut akan disimpan pada *cookies browser* pengguna.



Gambar 3.5 *Flowchart Sign In* Pada Aplikasi SSO

Pada Gambar 3.5 menjelaskan proses alur *sign in* yang dilakukan oleh *user* ketika langsung mengakses domain SSO. Alur *sign in* ini hampir sama dengan alur otentikasi aplikasi tunggal pada umumnya. Pengguna akan diminta *credential* dan aplikasi akan memverifikasinya. Jika *credential* berhasil diverifikasi dengan data pada *database*, maka pengguna tersebut akan diarahkan ke halaman utama aplikasi SSO yang akan menampilkan berbagai domain aplikasi yang sudah terintegrasi

dengan SSO. Namun jika verifikasi *credential* gagal, maka pengguna diarahkan ke halaman *login* kembali.



Gambar 3.6 *Flowchart Single Sign-On*

Pada Gambar 3.6 menjelaskan alur proses *single sign-on* ketika pengguna mengakses domain aplikasi secara langsung. Setiap aplikasi akan melakukan *request* ke API SSO untuk meminta data pengguna berdasarkan *token* pada setiap kali diakses oleh pengguna. Jika aplikasi tersebut tidak memiliki *token* atau *token* yang dimiliki sudah tidak valid, maka pengguna akan diarahkan ke halaman *login* SSO. Pengguna harus memasukkan *credential* terlebih dahulu dan selanjutnya akan

diverifikasi oleh SSO. Ketika *credential* valid, maka SSO akan mengarahkan pengguna ke halaman *callback*. Pada halaman tersebut, aplikasi melakukan *request* ke API SSO untuk memperoleh *token* dan data pengguna. Kemudia aplikasi akan mengambil *role* pengguna tersebut dari *database* aplikasi. Jika *role* berhasil diperoleh, maka pengguna akan diarahkan kembali ke halaman utama aplikasi yang dituju namun jika *role* tidak tersedia, maka aplikasi akan memberikan notifikasi bahwa akses ke dalam aplikasi tidak diizinkan bagi pengguna tersebut dan pengguna akan diarahkan ke halaman utama SSO.

Gambar 3.7 *Flowchart Logout* SSO

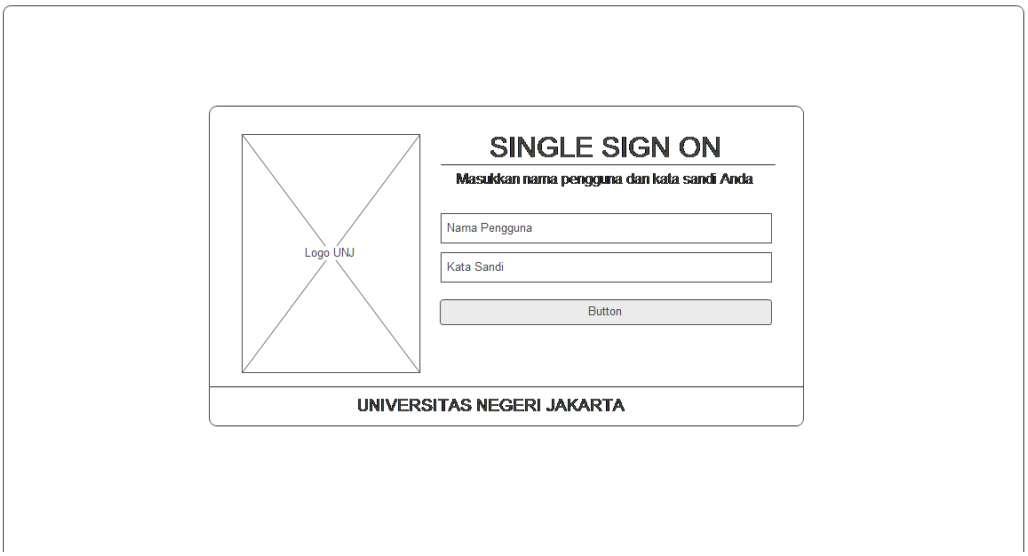
Pada Gambar 3.7 menjelaskan alur proses *logout* pada SSO. Diawali dengan pengguna yang menekan tombol *logout* pada aplikasi kemudian menonaktifkan *token* SSO terlebih dahulu dengan cara melakukan *request* ke API *logout* SSO

dengan menyertakan *token* SSO sebagai *Header Authorization*. Setelah *token* SSO telah berhasil dinonaktifkan, aplikasi melakukan *request* ke API *logout* aplikasi dengan menyertakan *token* aplikasi sebagai *Header Authorization*. Setelah semua *token* dinonaktifkan, aplikasi akan mengarahkan pengguna ke halaman *login* SSO.

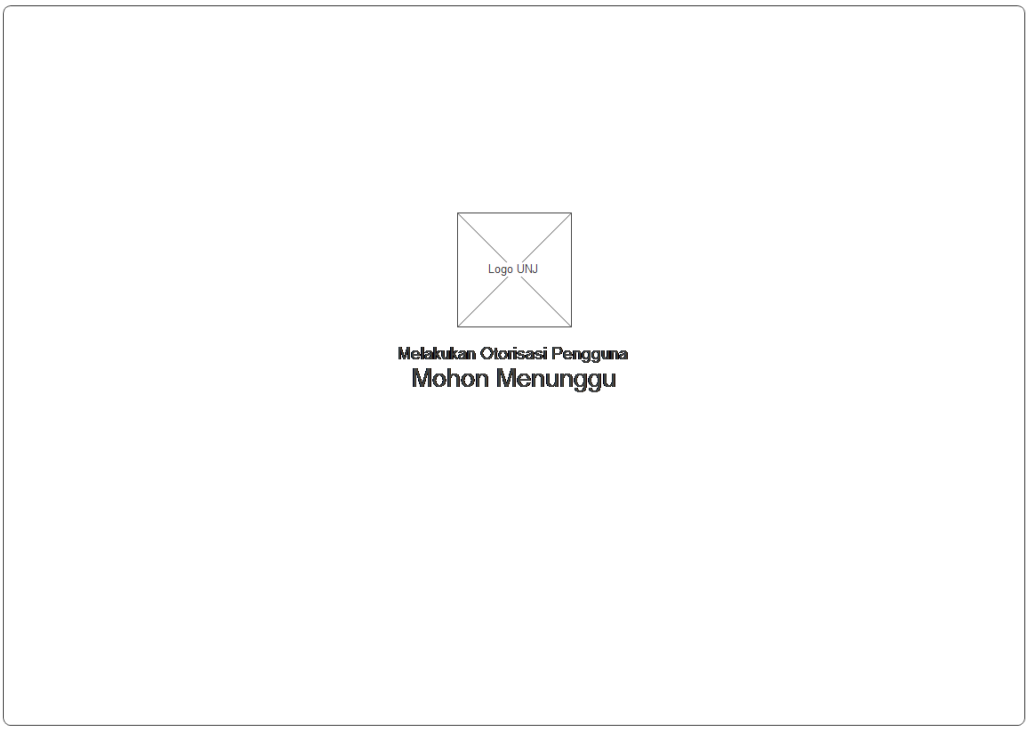
3.3.2.3. Rancangan Tampilan

Pada tahap ini dilakukan perancangan desain tata letak aplikasi berupa *wireframe*. *Wireframe* dibuat dengan menggunakan *software* Edraw Max. Tujuan dibuatnya *wireframe* ini, yaitu: sebagai gambaran tata letak dari aplikasi yang akan dikembangkan sehingga proses pengembangan menjadi lebih terarah.

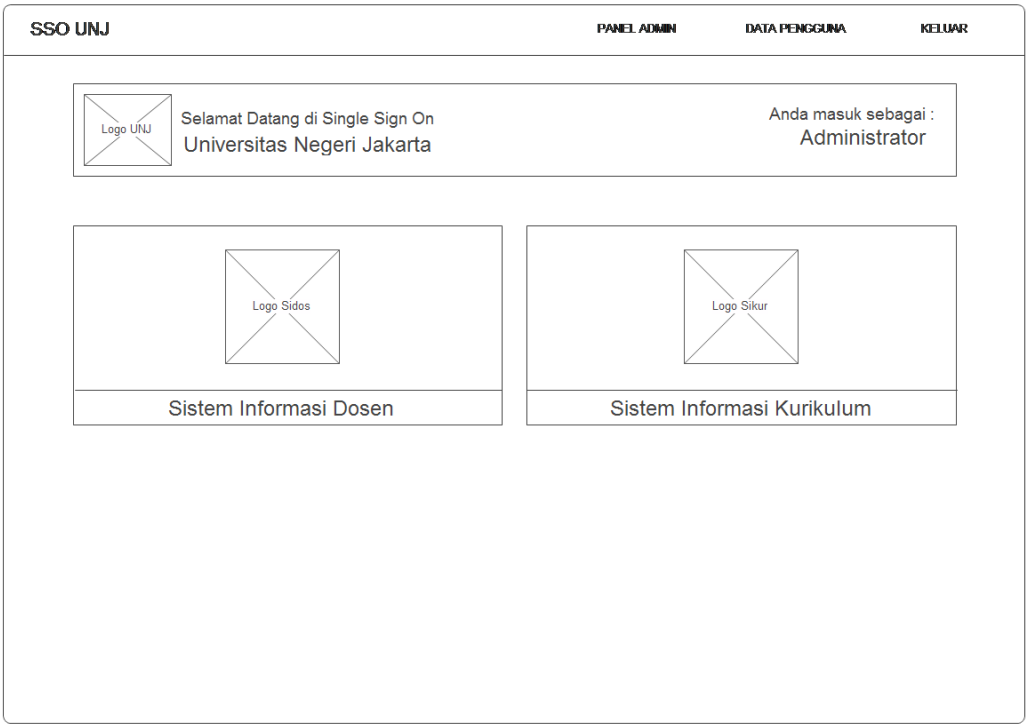
Adapun rancangan tampilan untuk aplikasi SSO yang akan dikembangkan dapat dilihat pada Gambar 3.8:



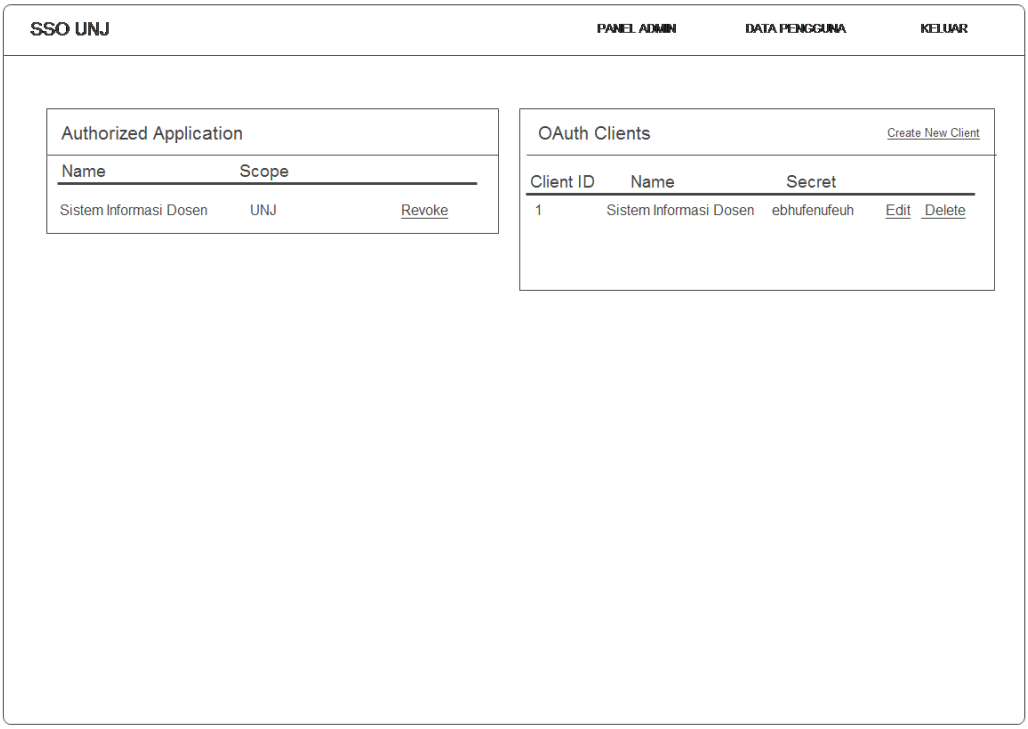
Gambar 3.8 Halaman *Login Single Sign-On*



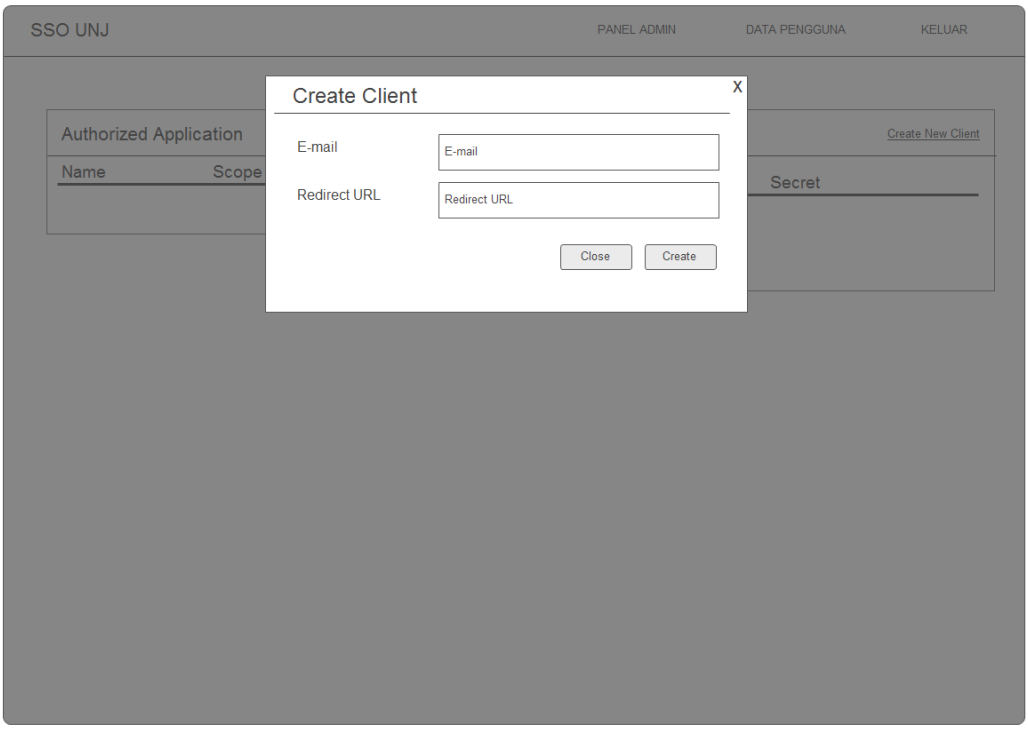
Gambar 3.9 Halaman *Callback* Aplikasi *Client*



Gambar 3.10 Halaman Utama *Single Sign-On*

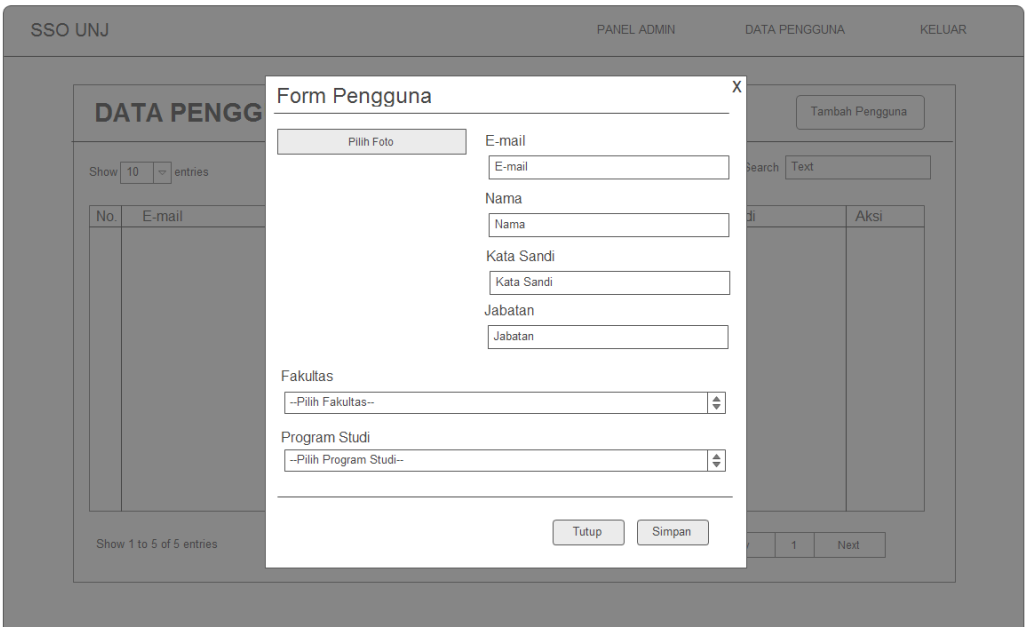


Gambar 3.11 Halaman Panel *Admin*



Gambar 3.12 *Form OAuth Clients*

Gambar 3.13 Halaman Data Pengguna



Gambar 3.14 *Form* Data Pengguna

3.4. Teknik dan Prosedur Pengumpulan Data

Teknik pengumpulan data dilakukan dengan teknik wawancara. Wawancara dilakukan dengan narasumber M. Ficky Duskarnaen selaku Kepala UPT TIK UNJ. Wawancara tersebut dilakukan untuk mendapatkan informasi mengenai:

1. Karakteristik data akun pengguna pada SIDOS dan SIKUR UNJ;
2. Proses penanganan data akun pengguna pada SIDOS dan SIKUR UNJ;
3. Kekurangan dari proses penanganan data akun pengguna pada SIDOS dan SIKUR UNJ;
4. Karakteristik perangkat lunak yang dibutuhkan untuk mengatasi kekurangan tersebut;
5. Fitur-fitur yang ingin diterapkan pada sistem *Single Sign On*.

3.5. Teknik Analisis Data

Teknik analisis data dilakukan dengan pengujian sistem menggunakan metode *black box.* Pengujian *Black Box* (*Black Box Testing*) merupakan pengujian yang dilakukan dengan mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari sistem SSO. Dalam hal ini data akun pengguna dengan variasi *role user* akan menjadi data uji. Pada penerapan *black box* umumnya tidak memerlukan pengetahuan khusus dari struktur internal aplikasi dan pengetahuan pemrograman. Perancang pengujian hanya perlu memilih masukkan yang valid dan tidak valid dan menentukan keluaran yang benar.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak baik unit, integrasi, dan fungsional. Pengujian pada *Black Box* berusaha menemukan kesalahan seperti:

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan antar muka
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

Teknik analisis data dilakukan dengan pengujian sistem menggunakan metode *Black-box* dengan teknik *feature test*, yaitu pengujian untuk menguji fungsi-fungsi dari aplikasi yang dirancang tanpa melihat *source code* aplikasi. Metode ini digunakan untuk mengetahui apakah sistem berfungsi berdasarkan kreteria keberhasilan program secara fungsional sehingga kesalahan dalam aplikasi dalam memenuhi kebutuhan dapat diketahui.

Adapun skenario pengujian dengan metode *black box* dapat dilihat pada Tabel 3.9 dan Tabel 3.10:

Tabel 3.9 Pengujian Fungsional Utama

No.	Fungsi	Skenario Proses	Hasil yang Diharapkan	Sistem Bekerja (Ya/Tidak)*
1	*Redirecting*	Pada saat pengguna mengakses domain *client*	Diarahkan ke halaman *login* SSO *server* dengan membawa kode *Client ID*	
2	*Login Client*	Pada saat pengguna memasukkan *username* dan *password*, klik tombol *login*	Diarahkan ke halaman *callback client* dan melakukan otentikasi. Jika *username* dan *password* benar,	

			maka SSO *server* akan menghasilkan *token* dan mengarahkan ke domain *client* utama. Jika salah, muncul pesan kesalahan dan mengarahkan ke halaman *login* SSO	
3	Otentikasi Terpusat	Pada saat pengguna sudah berhasil *login* pada satu aplikasi *client*, kemudian pengguna mengakses domain *client* lainnya	Diarahkan ke halaman *callback client* dan melakukan otentikasi. Jika data pengguna terdapat di aplikasi *client* yang akan diakses, maka SSO *server* mengirimkan *token* ke domain *client* dan diarahkan ke halaman utama *client*. Jika tidak, maka diarahkan ke halaman utama SSO *server*	
4	Otorisasi	Pada saat berhasil *login* di SSO *server* dan diarahkan ke halaman utama aplikasi *client*	*User ID* akan dicocokkan dengan tabel *role database client* kemudian menampilkan menu sesuai dengan *role* tersebut. Jika tidak ada *role*, maka diarahkan ke halaman utama SSO *server*	
5	*Logout*	Pada saat pengguna klik tombol *logout*	Sistem akan menghapus sesi dan menonaktifkan *token* pada SSO *server* dan *client* kemudian mengarahkan ke halaman *login*	

Tabel 3.10 Pengujian Fungsional *Admin*

No.	Fungsi	Skenario Proses	Hasil yang Diharapkan	Sistem Bekerja (Ya/Tidak)*
1	Navigasi	Pada saat pengguna sudah masuk ke sistem, klik menu panel admin	Tampilan akan diarahkan ke panel admin	
2	Navigasi	Pada saat berada di halaman panel admin, klik tombol *create new client*	Sebuah borang akan muncul bergeser dari atas	
3	Navigasi	Pada daftar *client* di halaman panel admin, klik tombol *edit* pada *client*	Sebuah borang akan muncul beserta data *client* yang sedang diubah	
4	Navigasi	Pada saat pengguna sudah masuk ke sistem, klik menu data pengguna	Tampilan akan diarahkan ke halaman data pengguna	
5	Navigasi	Pada saat berada di halaman data pengguna, klik tombol tambah pengguna	Sebuah borang akan muncul bergeser dari atas	
6	Navigasi	Pada saat borang berbentuk *modal* muncul, klik tombol *cancel*	Borang akan tertutup dengan bergeser ke atas	
7	Navigasi	Pada halaman utama, klik salah satu *client*	Tampilan akan diarahkan ke domain *client*	
8	Menyimpan *Client*	Pada saat mengisi borang *client*, kemudian klik tombol *submit*	Data *client* baru akan tersimpan dan tampilan diarahkan ke panel admin beserta daftar *client* terbaru akan muncul	
9	Mengubah *Client*	Pada saat mengisi borang ubah *client*, klik tombol *submit*	Data *client* akan diperbarui dan tampilan diarahkan ke panel admin beserta daftar *client* terbaru akan muncul	
10	Menghapus *Client*	Pada daftar *client* di halaman panel	Data *client* akan terhapus dan tampilan diarahkan	

		admin, klik tombol *delete*	ke panel admin beserta daftar *client* terbaru akan muncul	
11	Menyimpan data pengguna	Pada saat mengisi borang pengguna, kemudian klik tombol *submit*	Data pengguna baru akan tersimpan dan tampilan diarahkan ke halaman data pengguna beserta daftar pengguna terbaru akan muncul	
12	Mengubah data pengguna	Pada saat mengisi borang ubah pengguna, klik tombol *submit*	Data pengguna akan diperbarui dan tampilan diarahkan ke halaman data pengguna beserta daftar pengguna terbaru akan muncul	
13	Menghapus data pengguna	Pada daftar pengguna di halaman data pengguna, klik tombol *delete*	Data pengguna akan terhapus dan tampilan diarahkan ke halaman data pengguna beserta daftar pengguna terbaru akan muncul	

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1. Deskripsi Hasil Penelitian

Hasil penelitian adalah sebuah sistem otentikasi terpusat yang terintegrasi pada lebih dari 1 aplikasi berbasis *web* yang dalam hal ini, yaitu: SIDOS dan SIKUR. Namun SIKUR yang terintegrasi pada sistem otentikasi tersebut dibuat dalam bentuk *prototype* karena aplikasi tersebut masih dalam proses pengembangan. *Prototype* SIKUR dibuat dengan batasan kebutuhan manajemen pengguna sehingga tidak ada fitur SIKUR yang benar-benar aktif selain pengelolaan *role* pengguna dan pengaturan menu berdasarkan *role* pengguna yang sedang aktif. Adapun aplikasi SIDOS yang sudah diterbitkan secara *online* diunduh ke komputer lokal beserta struktur *database* untuk dilakukan penyesuaian dengan sistem otentikasi yang dikembangkan.

Sistem otentikasi yang dikembangkan menyimpan dan mengelola data pengguna untuk seluruh aplikasi yang telah terintegrasi. Setiap aplikasi yang terintegrasi dengan sistem otentikasi tersebut akan memiliki *id* dan *secret key* yang diperoleh dari sistem secara acak sebagai informasi unik yang digunakan untuk validasi ketika otentikasi dilakukan. Jika aplikasi tidak terdaftar pada sistem otentikasi, maka otentikasi tidak bisa dilakukan. Setiap aplikasi tidak lagi melakukan otentikasi secara internal tetapi akan mengarahkan pengguna ke halaman sistem otentikasi dan meminta *credential* kepada pengguna apabila halaman yang diakses membutuhkan data pengguna yang sedang aktif. Sistem otentikasi tersebut akan meminta *credential* dan mengarahkan pengguna ke

halaman *callback* kemudian memeriksa *credential* tersebut dan memberikan *token* kepada aplikasi sebagai tiket untuk mendapatkan data pengguna yang sedang aktif. Setelah berhasil memperoleh *token* dari sistem otentikasi, aplikasi dapat menggunakan metode otorisasi apa saja yang paling cocok untuk aplikasi tersebut. Jika pengguna sudah aktif pada salah satu aplikasi, maka pengguna tidak perlu mengirimkan *credential* lagi ketika akan mengakses aplikasi lainnya. Pada proses *logout,* aplikasi harus menonaktifkan *token* yang diberikan oleh sistem otentikasi sehingga ketika pengguna melakukan *logout* pada salah satu aplikasi kemudian mengakses aplikasi lainnya, sistem akan menghapus seluruh sesi yang sedang aktif dan mengarahkan pengguna ke halaman *login* sistem otentikasi. Selain itu, sistem otentikasi ini juga memiliki beberapa fungsi lainnya sesuai dengan kebutuhan yang telah dianalisis.

Setelah pengembangan secara lokal dilakukan, maka sistem otentikasi tersebut diunggah ke *server* agar bisa diintegrasikan dengan aplikasi SIDOS yang sudah berjalan secara *online*. Setiap penyesuaian aplikasi SIDOS yang dilakukan di komputer lokal juga diunggah ke *server* aplikasi SIDOS sehingga sistem otentikasi tersebut bisa berjalan secara *online* dan sudah terintegrasi dengan aplikasi SIDOS pada *server*.

4.2. Analisis Data Penelitian

Setelah produk berhasil dikembangkan, selanjutnya dilakukan pengujian produk untuk menguji kelayakan dari produk tersebut. Pengujian dengan metode *black box testing* dilakukan dengan menjalankan seluruh fungsional yang telah

didefinisikan dan menganalisis keselarasan antara hasil yang diharapkan pada skenario proses dan *feedback* dari sistem.

Adapun hasil pengujian produk dapat dilihat pada Tabel 4.1 dan Tabel 4.2:

Tabel 4.1 Pengujian Fungsional Utama

No.	Fungsi	Skenario Proses	Hasil yang Diharapkan	Sistem Bekerja (Ya/Tidak)*
1	*Redirecting*	Pada saat pengguna mengakses domain *client*	Diarahkan ke halaman *login* SSO *server* dengan membawa kode *Client ID*	Ya
2	*Login Client*	Pada saat pengguna memasukkan *username* dan *password*, klik tombol *login*	Diarahkan ke halaman *callback client* dan melakukan otentikasi. Jika *username* dan *password* benar, maka SSO *server* akan menghasilkan *token* dan mengarahkan ke domain *client* utama. Jika salah, muncul pesan kesalahan dan mengarahkan ke halaman *login* SSO	Ya
3	Otentikasi Terpusat	Pada saat pengguna sudah berhasil *login* pada satu aplikasi *client*, kemudian pengguna mengakses domain *client* lainnya	Diarahkan ke halaman *callback client* dan melakukan otentikasi. Jika data pengguna terdapat di aplikasi *client* yang akan diakses, maka SSO *server* mengirimkan *token* ke domain *client* dan diarahkan ke halaman utama *client*. Jika tidak,	Ya

			maka diarahkan ke halaman utama SSO *server*	
4	Otorisasi	Pada saat berhasil *login* di SSO *server* dan diarahkan ke halaman utama aplikasi *client*	*User ID* akan dicocokkan dengan tabel *role database client* kemudian menampilkan menu sesuai dengan *role* tersebut. Jika tidak ada *role*, maka diarahkan ke halaman utama SSO *server*	Ya
5	*Logout*	Pada saat pengguna klik tombol *logout*	Sistem akan menghapus sesi dan menonaktifkan *token* pada SSO *server* dan *client* kemudian mengarahkan ke halaman *login*	Ya

Tabel 4.2 Pengujian Fungsional *Admin*

No.	Fungsi	Skenario Proses	Hasil yang Diharapkan	Sistem Bekerja (Ya/Tidak)*
1	Navigasi	Pada saat pengguna sudah masuk ke sistem, klik menu panel admin	Tampilan akan diarahkan ke panel admin	Ya
2	Navigasi	Pada saat berada di halaman panel admin, klik tombol *create new client*	Sebuah borang akan muncul bergeser dari atas	Ya
3	Navigasi	Pada daftar *client* di halaman panel admin, klik tombol *edit* pada *client*	Sebuah borang akan muncul beserta data *client* yang sedang diubah	Ya
4	Navigasi	Pada saat pengguna sudah masuk ke sistem, klik menu data pengguna	Tampilan akan diarahkan ke halaman data pengguna	Ya
5	Navigasi	Pada saat berada di halaman data pengguna, klik tombol tambah pengguna	Sebuah borang akan muncul bergeser dari atas	Ya

6	Navigasi	Pada saat borang berbentuk *modal* muncul, klik tombol *cancel*	Borang akan tertutup dengan bergeser ke atas	Ya
7	Navigasi	Pada halaman utama, klik salah satu *client*	Tampilan akan diarahkan ke domain *client*	Ya
8	Menyimpan *Client*	Pada saat mengisi borang *client*, kemudian klik tombol *submit*	Data *client* baru akan tersimpan dan tampilan diarahkan ke panel admin beserta daftar *client* terbaru akan muncul	Ya
9	Mengubah *Client*	Pada saat mengisi borang ubah *client*, klik tombol *submit*	Data *client* akan diperbarui dan tampilan diarahkan ke panel admin beserta daftar *client* terbaru akan muncul	Ya
10	Menghapus *Client*	Pada daftar *client* di halaman panel admin, klik tombol *delete*	Data *client* akan terhapus dan tampilan diarahkan ke panel admin beserta daftar *client* terbaru akan muncul	Ya
11	Menyimpan data pengguna	Pada saat mengisi borang pengguna, kemudian klik tombol *submit*	Data pengguna baru akan tersimpan dan tampilan diarahkan ke halaman data pengguna beserta daftar pengguna terbaru akan muncul	Ya
12	Mengubah data pengguna	Pada saat mengisi borang ubah pengguna, klik tombol *submit*	Data pengguna akan diperbarui dan tampilan diarahkan ke halaman data pengguna beserta daftar pengguna terbaru akan muncul	Ya

13	Menghapus data pengguna	Pada daftar pengguna di halaman data pengguna, klik tombol *delete*	Data pengguna akan terhapus dan tampilan diarahkan ke halaman data pengguna beserta daftar pengguna terbaru akan muncul	Ya

4.3. Pembahasan

Pengujian dilakukan oleh 2 orang penguji yang bekerja di UPT TIK Universitas Negeri Jakarta, yaitu: *programmer* yang bernama Fajar Maulana, S.Pd. dan *web designer* yang bernama Hanifa Fissalma, S.Pd.. Penguji menggunakan sebuah komputer yang terkoneksi dengan *server prototype* SSO yang merupakan komputer pribadi penulis dan melakukan pengujian berdasarkan Tabel 3.9 dan Tabel 3.10. Masing-masing penguji menggunakan akun yang berbeda untuk mendapatkan hasil pengujian yang optimal.

Pengujian berlangsung pada tanggal 20 Juli 2017 di UPT TIK UNJ dan berjalan dengan cukup baik namun sempat terjadi *error* pada saat proses *logout*. Ketika pengguna menekan tombol *logout* pada aplikasi SIKUR kemudian kembali mengakses aplikasi SIDOS, pengguna masih bisa menggunakan fitur-fitur yang ada pada SIDOS padahal seharusnya sesi untuk aplikasi SIDOS juga turut dinonaktifkan bersamaan pada saat sesi SIKUR dan SSO dihapus. Namun tidak sebaliknya, ketika pengguna menekan tombol *logout* pada aplikasi SIDOS kemudian kembali mengakses aplikasi SIKUR, maka pengguna akan diarahkan ke halaman *login* SSO karena dianggap sudah tidak memiliki sesi yang aktif atau dengan kata lain *token* telah kedaluwarsa. Setelah diteliti oleh penulis, ternyata terdapat *script* yang belum dijalankan pada aplikasi SIDOS yakni pengecekan *token*

SSO yang seharusnya dieksekusi pada setiap *request* sebagai acuan bahwa pengguna masih memiliki sesi aktif di salah satu aplikasi yang terintegrasi dengan SSO. Solusi yang dilakukan adalah menambahkan *script* tersebut ke dalam aplikasi SIDOS. Setelah perbaikan dilakukan, maka pengujian pada fungsi *logout* dilakukan kembali dan memberikan hasil yang diharapkan.

Pada akhirnya, berdasarkan hasil pengujian pada Tabel 4.1 dan Tabel 4.2 yang telah dievaluasi oleh penguji, dapat dilihat bahwa hasil pengujian menunjukkan seluruh fungsional sistem otentikasi telah berjalan dengan baik sesuai dengan skenario proses yang diharapkan. Oleh sebab itu, aplikasi SSO dianggap layak untuk digunakan oleh pengguna secara massal.

4.4. Aplikasi Hasil Penelitian

Hasil dari penelitian berupa sistem otentikasi SSO dapat diterapkan pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum di Universitas Negeri Jakarta. SSO ini juga dapat dikembangkan untuk bisa diintegrasikan dengan sistem informasi lainnya yang berjalan di UNJ. Dengan diterapkannya SSO pada sistem informasi UNJ diharapkan dapat memudahkan dosen, mahasiswa, staff program studi, staff fakultas, dan seluruh pengguna aktif sistem informasi UNJ untuk mengakses berbagai sistem informasi yang sedang berjalan.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil penelitian yang telah diimplementasikan, maka dapat ditarik kesimpulan sebagai berikut:

1. Pengembangan teknologi SSO pada SIDOS dan SIKUR di Universitas Negeri Jakarta dilakukan dengan membuat sistem otentikasi terpusat yang mengarahkan pengguna ke halaman *login* ketika pengguna mengakses langsung ke aplikasi yang sudah terintegrasi SSO. Kemudian meminta *credential* pengguna dan memverifikasinya. Jika *credential* tersebut valid, maka sistem memberikan *token* sebagai tiket untuk masing-masing aplikasi. Masing-masing aplikasi harus melakukan pengecekan *token* pada setiap *request* yang dilakukan pengguna. Pengguna tidak perlu melakukan *login* lagi jika ingin mengakses aplikasi lainnya. Apabila pengguna telah keluar dari salah satu aplikasi yang terintegrasi dengan SSO, maka sistem akan menonaktifkan *token* yang dimiliki oleh pengguna dan pengguna harus kembali memasukkan *credential* untuk proses *login*;
2. Data pengguna untuk otentikasi disimpan secara terpusat pada *database* SSO dan data *role* pengguna tetap disimpan dan dikelola oleh masing-masing aplikasi;
3. Aplikasi SIKUR yang diintegrasikan dengan SSO dibuat dalam bentuk *prototype* karena aplikasi tersebut masih dalam tahap pengembangan. *Prototype* SIKUR dibuat dengan batasan kebutuhan manajemen pengguna

saja artinya tidak ada fitur SIKUR yang benar-benar aktif kecuali pengelolaan *role* pengguna dan pengaturan menu berdasarkan *role* pengguna yang sedang aktif;

4. Berdasarkan hasil pengujian dengan menggunakan *black box testing* pada masing-masing fungsional yang digambarkan pada Tabel 4.1 dan Tabel 4.2, dapat disimpulkan bahwa sistem otentikasi SSO telah berjalan dengan baik sesuai dengan kebutuhan fungsional dan layak untuk digunakan.

5.2. Saran

Menurut penulis, saran yang dapat dilakukan untuk penelitian berikutnya antara lain sebagai berikut:

1. Melakukan penelitian mengenai *Single Sign-On* dengan menggunakan LDAP (*Lightweight Directory Access Protocol*) sebagai media penyimpanan data pengguna;
2. Meningkatkan keamanan pada sistem otentikasi.

DAFTAR PUSTAKA

Abdurrahman, Luthfi. 2012. *Implementasi Sistem Single Sign On menggunakan OpenAM dengan Otentikasi Kerberos dan OpenLDAP*. *Skripsi*. Jurusan Teknik Elektro, Universitas Sumatra Utara

Aminudin. 2008. *Implementasi Single Sign On (SSO) untuk Mendukung Interaktivitas Aplikasi E-Commerce menggunalan Protocol OAUTH*. Malang:Jurnal GAMMA,ISSN 2086-3071:109-115.

Anhar. 2010. *Panduan Menguasai PHP & MySQL secara Otodidak*. Jakarta Selatan : Mediakita.

Anzizhan, Syafruddin. 2004. *Sistem Pengambilan Keputusan Pendidikan*. Surabaya : Grasindo.

Argawal, B.B, Tayal, S.P dan Gupta, M. 2010. *Software Engineering and Computer Software Testing*. United State : Sudbury.

Arief, Muhammad. 2011. *Pemograman Web Dinamis menggunakan PHP dan MySQL*. Yogyakarta : Andi Offset.

Buecker, Axel, dkk. 2012. *Enterprise Single Sign-On Design Guide*. Amerika Serikat : International Business Machine Corporation.

Djayusman, Doni. 2013. *Pembangunan Aplikasi E-Commerce di Mag and Shoes Shop [Skripsi]*. Bandung : Fakultas Teknik dan Ilmu Komputer, Unikom.

Fatta Al, Hanif. 2007. *Analisis & Perancangan Sistem Informasi*. Yogyakarta : CV. Andi Offset.

Gaffin, J.C. 2007. *Internet Protocol 6*. New York : Nova Science Publisher.

Hutahaean, Jepersen. 2014. *Konsep Sistem Informasi*. Yogyakarta : Deepublish.

IdCloudHost. *Pengertian dan Keunggulan Framework Laravel*. Diambil dari https://idcloudhost.com/pengertian-dan-keunggulan-framework-laravel/ pada tanggal 28 Juli 2017 Pukul 10.15 WIB.

Jimmy, Chr. 2008. *Sistem Informasi Manajemen*. Surabaya : Grasindo.

Keycdn. 2016. *What is the Difference Between HTTP and HTTPS?*. Diambil dari https://www.keycdn.com/blog/difference-between-http-and-https/. Diakses pada tanggal 17 April 2017 Pukul 12.53 WIB.

Kusrini. 2007. *Strategi Perancangan dan Pengelolaan Database*. Yogyakarta : CV Andi Offset.

Laravel *Documentation*. API *Authentication* (Passport). Diambil dari https://laravel.com/docs/5.4/passport pada tanggal 28 Juli 2017 Pukul 10.34 WIB.

Marimin dkk. 2006. *Sistem Informasi Manajemen Sumber Daya Manusia*. Bogor : Grasindo.

Microsoft. 2003. *What is TLS/SSL*. Diambil dari https://technet.microsoft.com/en-us/library/cc784450(v=ws.10).aspx. Diakses pada tanggal 17 April 2017 Pukul 13.34 WIB.

Mozilla. *Tentang Cookie*. Diambil dari https://support.mozilla.org/t5/Cookies-and-cache/Tentang-Cookie/ta-p/17722. Diakses pada tanggal 17 April 2017 Pukul 14.56 WIB.

OAuth2. *Halaman Utama*. Diambil dari https://oauth.net/2/. Diakses pada tanggal 19 Juli 2017 Pukul 11.22 WIB.

Octafian, Tri. 2015. *Web Multi E-Commerce berbasis Framework CodeIgniter*. Palembang : Jurnal Teknologi dan Informatika (TEKNOMATIKA):1-22

PHP. *What is PHP?*. Diambil dari http://php.net/manual/en/intro-whatis.php. Diakses pada tanggal 17 April 2017 Pukul 14.03 WIB.

Pressman, Roger. S. 2012. *Rekayasa Perangkat Lunak – A Practitioner's Approach*. New York : McGraw Hill.

Ramadhan, Gilang. 2012. *Analisis Teknologi Single Sign On (SSO) dengan Penerapan Central Authentication Service (CAS) pada Universitas Bina Darma*. *Skripsi*. Fakultas Ilmu Komputer, Universitas Bina Darma Palembang.

Simarmata, Janner. 2010. *Rekayasa Web*. Yogyakarta : CV.Andi Offset.

Sistem Informasi Dosen. 2017. *Halaman Login.* Diambil dari http://sidos.unj.ac.id. Diakses pada tanggal 15 Juni 2017 Pukul 13.56 WIB.

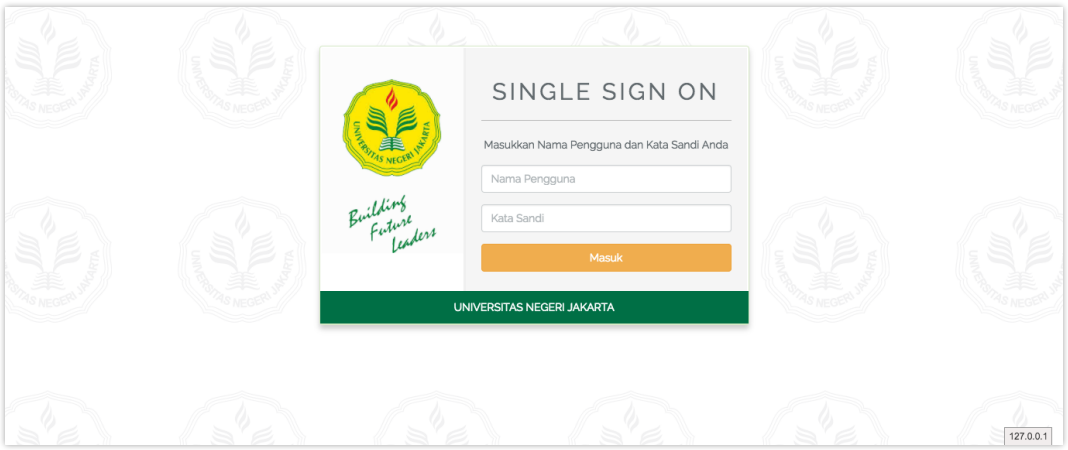
Stephen, dan Plew. 2000. *Database Design*. USA : Sams Publishing.

Tim Air Putih. 2014. *Panduan Framework Laravel*. Jakarta : Creative Commons.

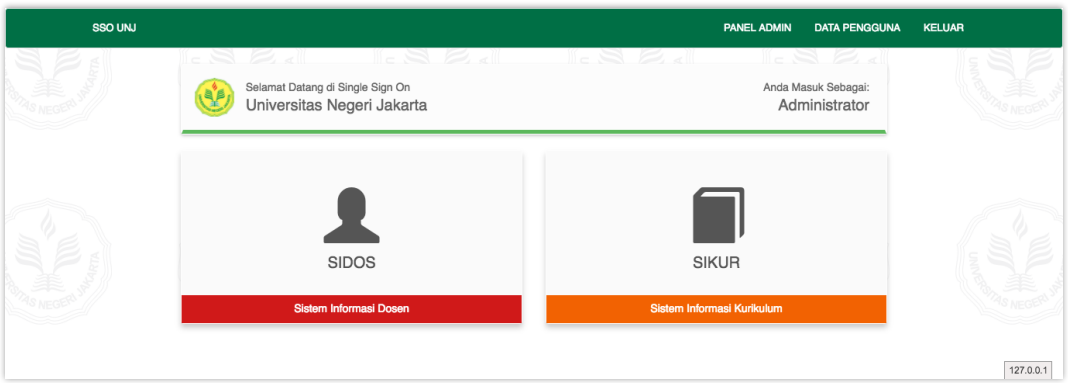
Universitas Negeri Jakarta. 2017. *Halaman Sejarah*. Diambil dari http://unj.ac.id. Diakses pada tanggal 15 Juni 2017 Pukul 13.40 WIB.

Lampiran 1. Tampilan Aplikasi *Single Sign-On*

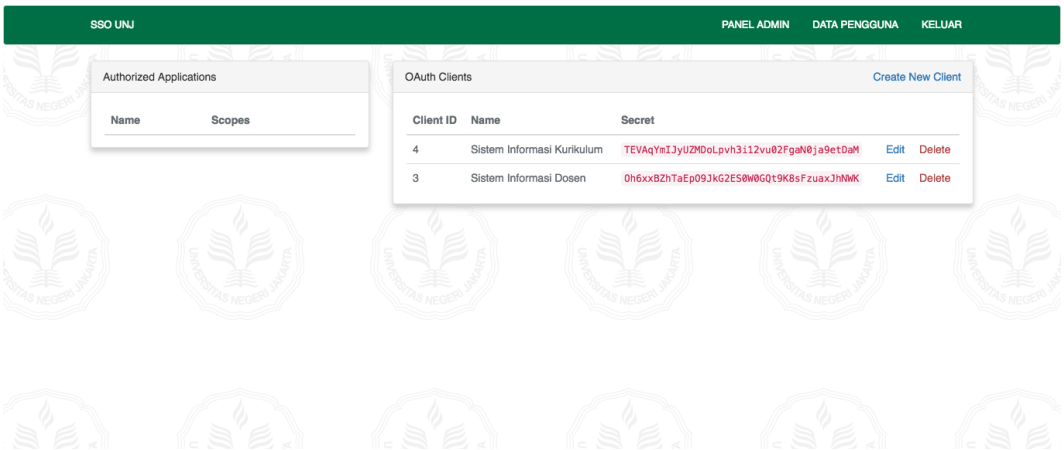
1. Halaman *Login*



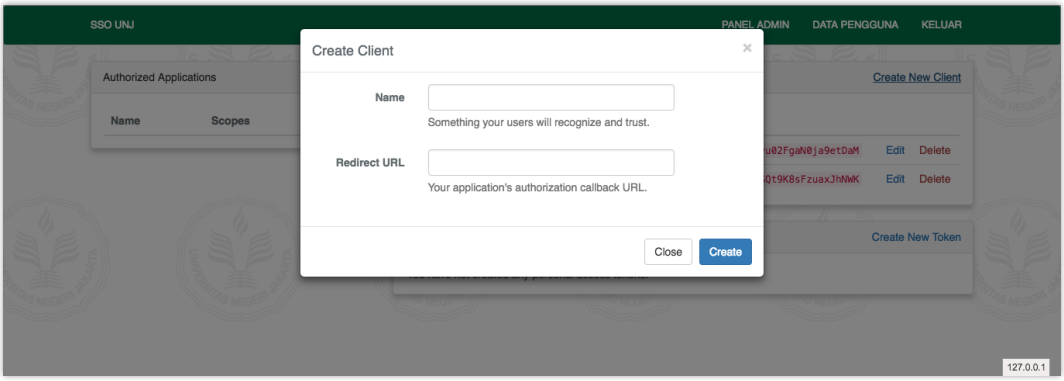
2. Halaman Utama



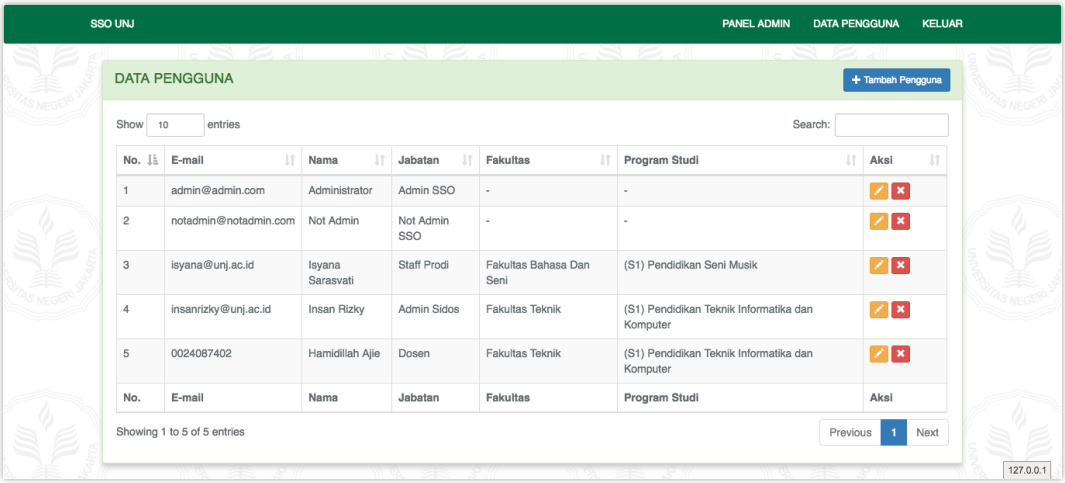
3. Halaman Panel *Admin*



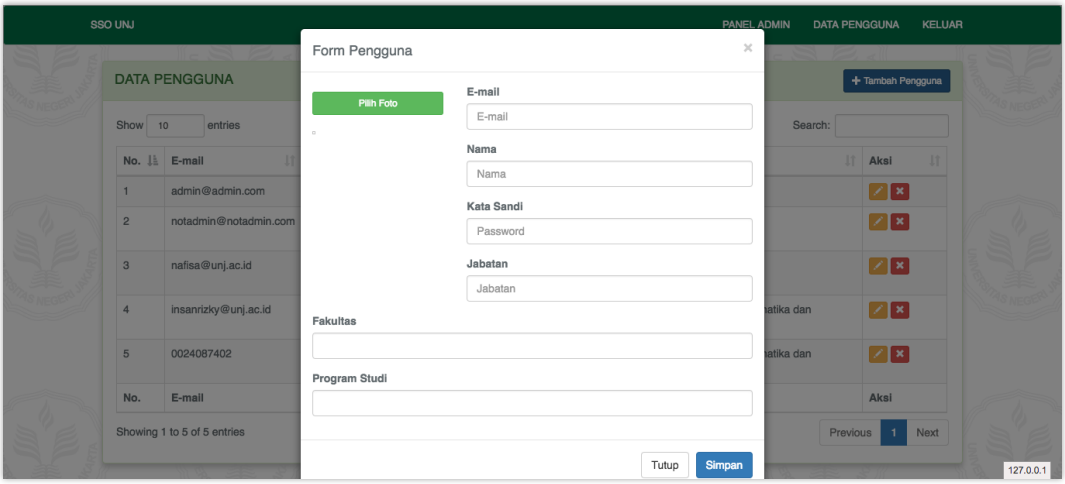
4. *Form Oauth Clients*



5. Halaman Data Pengguna



6. *Form* Data Pengguna



7. Halaman *Callback* Aplikasi *Client*

Lampiran 2. Struktur *Database* Laravel Passport

1. Tabel *OAuth Access Tokens*

No.	Nama Kolom	Tipe Data	Keterangan
1	id	varchar (100)	not null (pk)
2	user_id	int (11)	null (uk)
3	client_id	int (11)	not null
4	name	varchar (255)	null
5	scopes	text	null
6	revoked	tinyint (1)	not null
7	created_at	timestamp	null
8	updated_at	timestamp	null
9	expires_at	datetime	null

2. Tabel *OAuth Auth Codes*

No.	Nama Kolom	Tipe Data	Keterangan
1	id	VARCHAR (100)	NOT NULL (PK)
2	user_id	INT (11)	NOT NULL
3	client_id	INT (11)	NOT NULL
4	scopes	TEXT	NULL
5	revoked	TINYINT (1)	NOT NULL
6	expires_at	DATETIME	NULL

3. Tabel *OAuth Clients*

No.	Nama Kolom	Tipe Data	Keterangan
1	id	INT (10)	AUTO_INCREMENT (PK)
2	user_id	INT (11)	NULL (UK)
3	name	VARCHAR (255)	NOT NULL
4	secret	VARCHAR (100)	NOT NULL
5	redirect	TEXT	NOT NULL
6	personal_access_client	TINYINT (1)	NOT NULL
7	password_client	TINYINT (1)	NOT NULL
8	revoked	TINYINT (1)	NOT NULL
9	created_at	TIMESTAMP	NULL
10	updated_at	TIMESTAMP	NULL

4. **Tabel *Personal Access Clients***

No.	Nama Kolom	Tipe Data	Keterangan
1	id	INT (10)	AUTO_INCREMENT (PK)
2	client_id	INT (11)	NOT NULL (UK)
3	created_at	TIMESTAMP	NULL
4	updated_at	TIMESTAMP	NULL

5. **Tabel *OAuth Refresh Tokens***

No.	Nama Kolom	Tipe Data	Keterangan
1	id	VARCHAR (100)	NOT NULL (PK)
2	access_token_id	VARCHAR (100)	NOT NULL (UK)
3	revoked	TINYINT (1)	NOT NULL
4	expires_at	TIMESTAMP	NULL

6. **Tabel *Password Resets***

No.	Nama Kolom	Tipe Data	Keterangan
1	email	VARCHAR (255)	NOT NULL (UK)
2	token	VARCHAR (255)	NOT NULL (UK)
3	created_at	TIMESTAMP	NULL

Lampiran 3. Hasil Wawancara

HASIL WAWANCARA

Narasumber : M. Ficky Duskarnaen, M.Sc.

Jabatan : Kepala UPT TIK UNJ

Tanggal : 04 Oktober 2017

Pertanyaan	:	Apa sajakah sistem informasi yang ada di Universitas Negeri Jakarta?
Jawaban	:	Banyak, ada Sistem Informasi Penerimaan Mahasiswa Baru (SIPENMABA), Sistem Informasi UKT (SIUKAT), Sistem Informasi Akademik (SIAKAD), Sistem Informasi Dosen (SIDOS), Sistem Informasi Kuriukulum (SIKUR), dan Sistem Informasi Mahasiswa (SIMAS).
Pertanyaan	:	Apakah ada keterkaitan hubungan antara data satu sistem dengan yang lainnya?
Jawaban	:	Ya, tentu ada keterkaitan pada beberapa sistem khususnya sistem informasi yang mengolah data dalam lingkup perkuliahan, yakni data mahasiswa, data dosen, dan data mata kuliah atau kurikulum.

Pertanyaan : Mengacu dari pernyataan bapak sebelumnya, dari tiga data, yaitu data mahasiswa, dosen, dan mata kuliah, sistem yang manakah yang mengolah data-data tersebut?

Jawaban : Masing-masing data tersebut memiliki sistem yang mengolahnya. Data mahasiswa diolah oleh Sistem Informasi Mahasiswa (SIMAS), data dosen diolah oleh Sistem Informasi Dosen (SIDOS), dan data kurikulum diolah oleh Sistem Kurikulum (SIKUR). Saat ini ketiga data tersebut diolah pada SIAKAD. Namun ke depannya akan dipisah menjadi tiga sistem tersebut.

Pertanyaan : Apakah ketiga aplikasi tersebut sudah diimplementasikan?

Jawaban : Sejauh ini yang sudah terimplementasi adalah SIDOS yang bisa diakses di http://sidos.unj.ac.id, sedangkan untuk SIKUR dan SIMAS masih dalam tahap pengembangan oleh tim IT kami.

Pertanyaan : Jika nanti ketiga aplikasi tersebut sudah diimplementasikan, siapa sajakah *user* yang terdaftar pada masing-masing sistem?

Jawaban : *User* yang terdaftar di SIDOS adalah seluruh dosen yang menjadi dosen di UNJ, *user* pada SIKUR yakni pejabat maupun dosen yang berperan dalam perancangan kurikulum, sedangkan SIMAS untuk mahasiswa mengisi identitas dirinya. Namun pada ketiga sistem tersebut, terdapat *role user* lainnya yang mampu melakukan rekap seperti pejabat misalnya.

Pertanyaan : Bagaimana penanganan data akun *user* pada ketiga sistem tersebut?

Jawaban : Penanganan data akun *user* dilakukan secara terpisah pada masing-masing sistem tersebut. Akun *user* dosen ditangani di SIDOS dan akun *user* kurikulum di tangani di SIKUR, begitu pula dengan SIMAS.

Pertanyaan : Apakah ada kesamaan dari segi fungsi di setiap akun yang ada di ketiga sistem tersebut?

Jawaban : Ada, sebagai contoh akun terdaftar Bu Yuli di SIDOS adalah sebagai dosen akan tetapi karena Bu Yuli merangkap sebagai Kaprodi, *user* terdaftar di SIKUR atas nama Bu Yuli sebagai Kaprodi yang menjadi validator kurikulum di program studi PTIK. Bu Yuli memiliki dua akun berbeda baik yakni sebagai dosen di SIDOS dan sebagai validator di SIKUR. Hal ini sangat memungkinkan terjadi *multiple account* yang terpisah untuk satu orang yang sama di sistem yang berbeda, tentu hal ini sangat merepotkan, perlu mengingat banyak akun jika terdaftar di banyak sistem yang ada di UNJ.

Pertanyaan : Perlukah dibuat sistem baru untuk mengakomodir akun *user* tersebut agar lebih terintegrasi?

Jawaban : Ya perlu, agar meminimalisir jumlah akun untuk diingat pada saat *login* ke sistem, tanpa menggeser fungsi akun di sistem yang sudah berjalan.

Pertanyaan : Saya berencana untuk membuat sistem tersebut dengan teknologi *single sign on*. *Single Sign On* (SSO) ini adalah sistem yang mengizinkan pengguna agar dapat mengakses seluruh sumber daya dalam jaringan hanya dengan menggunakan satu *credential* saja. Dengan adanya SSO ini, pengelolaan akun pengguna bisa dilakukan secara terpusat dan meminimalisir kesalahan *user* dalam memberikan *credential* ketika masuk ke dalam sebuah sistem.

Jawaban : Sebenarnya UPT TIK UNJ memang sedang merencanakan hal tersebut namun kami sampai saat ini belum mengembangkannya karena suatu dan lain hal. Hal yang sangat bagus jika ada yang ingin mengembangkan hal tersebut.

Pertanyaan : Adakah fitur lain yang ingin ditambahkan pada SSO yang akan dibuat?

Jawaban : SSO yang kami harapkan mampu menangani otentikasi secara terpusat sehingga *user* cukup memasukkan identitasnya satu kali saja untuk bisa mengakses beberapa sistem sesuai dengan *role*-nya. Namun untuk otorisasinya, karena *user* pada masing-masing sistem memiliki *role* yang berbeda-beda, maka menurut

saya tidak masalah jika otorisasi dilakukan di masing-masing sistem. Jadi SSO hanya melakukan otentikasi dan mengirimkan identitasnya ke masing-masing sistem nanti sistemnya yang melakukan pengecekan *role* sesuai dengan *user* tersebut sehingga fitur yang disediakan oleh sistem sesuai dengan *role user* tersebut. Adapun pengelolaan akun pada SSO, cukup identitas umumnya saja. Yang penting, admin SSO bisa melihat seluruh *user* yang ada di UNJ beserta identitasnya dan SSO mampu melakukan otentikasi kemudian mengirimkan identitas tersebut ke masing-masing sistem.

Pertanyaan : Kira-kira sistem informasi apakah yang sangat memadai untuk mulai diterapkan SSO?

Jawaban : Dari beberapa sistem informasi yang sudah disebutkan sebelumnya, mungkin SIDOS dan SIKUR merupakan sistem yang cocok untuk mulai diterapkannya SSO karena saat ini dua sistem tersebut lah yang sedang gencar dikembangkan dan memiliki variasi *role* yang cukup banyak sehingga akan membantu pengembangan SSO menjadi lebih baik.

Lampiran 4. Surat Keterangan Pengujian

SURAT KETERANGAN PENGUJIAN

Yang bertanda tangan di bawah ini:

Nama : Hanifa Fissalma

Instansi : UPT TIK UNJ

Jabatan : Web Developer

Telah melakukan pengujian terhadap fungsional utama dan fungsional *admin* yang digunakan dalam penelitian skripsi dengan judul "Pengembangan Teknologi *Single Sign-On* Pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum di Universitas Negeri Jakarta"

Yang dibuat oleh:

Nama : Muhamad Insan Rizky

NIM : 5235134409

Program Studi : Pendidikan Teknik Informatika dan Komputer

Setelah melakukan pengujian, maka masukan untuk penelitian tersebut adalah:

Tampilan login diperbaiki lagi backgroundnya. Selebihnya sudah sesuai dengan daftar fungsional yang ada.

Demikian surat keterangan ini dibuat untuk dipergunakan sebagaimana mestinya.

Jakarta, 21 Juli 2017

Hanifa Fissalma

SURAT KETERANGAN PENGUJIAN

Yang bertanda tangan di bawah ini:

Nama : Fajar Maulana, S.Pd.

Instansi : UPT TIK

Jabatan : Web Developer

Telah melakukan pengujian terhadap fungsional utama dan fungsional *admin* yang digunakan dalam penelitian skripsi dengan judul "Pengembangan Teknologi *Single Sign-On* Pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum di Universitas Negeri Jakarta"

Yang dibuat oleh:

Nama : Muhamad Insan Rizky

NIM : 5235134409

Program Studi : Pendidikan Teknik Informatika dan Komputer

Setelah melakukan pengujian, maka masukan untuk penelitian tersebut adalah:

Sudah sesuai kebutuhan

Demikian surat keterangan ini dibuat untuk dipergunakan sebagaimana mestinya.

Jakarta, 21 Juli 2017

Fajar Maulana, S.Pd.

Lampiran 5. Permohonan Izin Mengadakan Penelitian untuk Penulisan Skripsi

KEMENTERIAN RISET, TEKNOLOGI , DAN PENDIDIKAN TINGGI
UNIVERSITAS NEGERI JAKARTA
Kampus Universitas Negeri Jakarta, Jalan Rawamangun Muka, Jakarta 13220
Telepon/Faximile : Rektor : (021) 4893854, PRI : 4895130, PR II : 4893918, PR III : 4892926, PR IV :4893982
BUK : 4750930, BAKHUM : 4759081, BK : 4752180
Bagian UHT : Telepon, 4893726, Bagian Keuangan : 4892414, Bagian Kepegawaian : 4890536, Bagian Humas : 4898486
Laman : www.unj.ac.id

Building Future Leaders

Nomor : **3173/UN39.12/KM/2017** **18 Juli 2017**
Lamp. : -
H a l : **Permohonan Izin Mengadakan Penelitian untuk Penulisan Skripsi**

Yth. **Kepala UPT TIK**
Universitas Negeri Jakarta

Kami mohon kesediaan Saudara untuk dapat menerima Mahasiswa Universitas Negeri Jakarta :

N a m a	:	**Muhamad Irfan Rizky**
Nomor Registrasi	:	5235134409
Program Studi	:	Pendidikan Teknik Informatika dan Komputer
Fakultas	:	Teknik Universitas Negeri Jakarta
No. Telp/HP	:	087886701338

Dengan ini kami mohon diberikan ijin mahasiswa tersebut, untuk dapat mengadakan penelitian guna mendapatkan data yang diperlukan dalam rangka penulisan skripsi dengan judul :

"Pengembangan Teknologi *Single Sign-On* Pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum di Universitas Negeri Jakarta"
(Yang Dilaksanakan di Gedung D UPT TIK Universitas Negeri Jakarta)

Atas perhatian dan kerjasama Saudara, kami sampaikan terima kasih.

Kepala Biro Akademik, Kemahasiswaan,
dan Hubungan Masyarakat

Woro Sasmoyo, SH
NIP. 19630403 198510 2 001

Tembusan :
1. Dekan Fakultas Teknik
2. Koordinator Prodi Pendidikan Teknik Informatika dan Komputer

DAFTAR RIWAYAT HIDUP

Penulis bernama lengkap Muhamad Insan Rizky, lahir di Ciamis pada tanggal 02 November 1995, merupakan anak ke-dua dari pasangan Noor Sardono dan Tati Suryati. Penulis menempuh pendidikan formalnya di SDN 5 Rangkasbitung Barat, SMPN 1 Rangkasbitung, dan SMKN 1 Rangkasbitung. Pada tahun 2013, penulis diterima sebagai mahasiswa Program Studi Pendidikan Teknik Informatika dan Komputer, Fakultas Teknik, Universitas Negeri Jakarta melalui jalur SBMPTN. Dalam menyelesaikan studinya, penulis melaksanakan sebuah penelitian untuk pengerjaan skripsi dengan judul "Pengembangan Teknologi *Single Sign On* Pada Sistem Informasi Dosen dan Sistem Informasi Kurikulum di Universitas Negeri Jakarta" sebagai salah satu syarat dalam memperoleh gelar sarjana pendidikan.