Journal of Engineering, Technology & Applied Science
p-ISSN: 2721-7949, e-ISSN: 2721-8090

Original Research Paper

Aplikasi Administrasi SMK Averus Jakarta Menggunakan Arsitektur Client-Server dan Model-View-Controller

Yulianti[1], **Normalisa**[1]

[1] *Program Studi Teknik Informatika, Fakultas Teknik, Universitas Pamulang, Tanggerang Selatan, Indonesia.*

Article History
Received:
05.01.2020

Revised:
07.02.2020

Accepted:
29.03.2020

***Corresponding Author:**
Normalisa
Email:
dosen00377@unpam.ac.id

Abstrak: SMK Averus Jakarta yang terletak di Pondok Pinang, Jakarta Selatan merupakan salah satu institusi pendidikan yang belum menggunakan teknologi informasi karena data administrasinya sebagian dicatat dan diolah secara manual, walaupun sudah ada yang dicatat dan diolah menggunakan Microsoft Excel. Hal ini menyebabkan beberapa masalah seperti, banyak terjadi duplikasi data dan pembuatan laporan-laporan dari data-data administrasi cenderung lambat. Salah satu solusi untuk menyelesaikan permasalahan tersebut yaitu pembuatan aplikasi administrasi sekolah menggunakan arsitektur Client Server dan Model View Controller (MVC). Arsitektur Client-Server membagi aplikasi dalam dua bagian yaitu server yang berhubungan dengan pengelolaan basis data dan client yang berhubungan dengan antarmuka user. Sedangkan arsitektur MVC dapat mempermudah pengembangan aplikasi sesuai dengan kebutuhan di masa depan dan dapat mengurangi jumlah source code. Setelah pembangunan aplikasi selesai, disimpulkan bahwa aplikasi ini dapat mengurangi duplikasi data, memperbaiki organisasi data/dokumen, mempercepat pembuatan laporan yang diperlukan, dan meningkatkan efisiensi dan efektifitas pengembangan aplikasi.

Kata Kunci: Administrasi Sekolah, Client-Server, Model-View-Controller.

Averus Jakarta Vocational Administration Application Using Client-Server Architecture and Model-View-Controller

Abstract: Averus Jakarta Vocational School, located in Pondok Pinang, South Jakarta is one of the educational institutions that has not used information technology because its administrative data is partially recorded and processed manually, although some have been recorded and processed using Microsoft Excel. This causes several problems such as, a lot of data duplication and making reports from administrative data tend to be slow. One solution to solve this problem is the creation of a school administration application using Client Server architecture and Model View Controller (MVC). Client-Server Architecture divides the application into two parts, namely the server that is associated with database management and the client that is related to the user interface. While the MVC architecture can facilitate the development of applications according to future needs and can reduce the amount of source code. After the application development is completed, it is concluded that this application can reduce data duplication, improve data / document organization, speed up the reporting required, and improve the efficiency and effectiveness of application development.

Keywords: Client-Server, Model-View-Controller, School Administration.

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**

1. Pendahuluan

Saat ini media komputer telah menempati peranan penting terhadap dunia pendidikan [1]. Sudah banyak sekolah-sekolah yang memiliki komputer dan mengajarkan tentang teknologi informasi kepada murid-muridnya. Teknologi informasi di sekolah masih sebatas ilmu yang diajarkan di SMP, SMU, dan SMK sesuai kurikulum nasional [2]. Belum banyak sekolah yang mengimplementasikan teknologi informasi untuk menunjang pengolahan administrasinya. Komputerisasi administrasi merupakan hal yang penting dalam lembaga pendidikan karena dapat mempermudah petugas administrasi dalam mengerjakan tugas-tugasnya [3].

Berdasarkan hasil observasi sistem yang sedang berjalan pada SMK Averus, terlihat banyak duplikasi data karena setiap petugas memiliki master data sendiri-sendiri di dalam komputernya. Data-data administrasi sekolah tidak mudah untuk dilakukan pembaharuan (*update*) karena jika dilakukan pembaharuan (*update*) di salah satu komputer, maka data-data di komputer lain tidak ikut diperbaharui, sehingga harus dilakukan pembaharuan (*update*) data pada setiap computer.

Pembangunan aplikasi administrasi sekolah merupakan salah satu solusi pengolahan data dengan cara komputerisasi, sehingga informasi yang dihasilkan efektif dan efisien. Aplikasi yang baik harus didukung oleh model data yang baik yaitu model data yang fleksibel dan dapat disesuaikan dengan kebutuhan masa depan [4]. Aplikasi tersebut akan bermanfaat di masa depan jika ada kebutuhan baru, maka dapat dengan mudah untuk ditambahkan atau diperbaharui.

Berdasarkan dengan permasalahan di atas, agar tidak terjadi duplikasi data/dokumen dan tidak harus memperbaharui data di semua komputer jika ada perubahan, maka perlu dibuat aplikasi yang menggunakan satu basis data dan dapat diakses oleh semua komputer. Arsitektur yang dapat digunakan adalah *Client-Server*. Arsitektur tersebut dapat membagi proses/aktivitas menjadi dua yaitu proses/aktivitas yang berhubungan dengan interaksi dengan pengguna (*user*) dilakukan oleh *client* dan proses yang berhubungan dengan pengolahan basis data yang dilakukan oleh *server* [5]. Pembagian proses/aktivitas tersebut dapat menjadikan beban pada sistem komunikasi (jaringan komputer) dapat dikurangi, sehingga ketika jumlah *client* cukup banyak tidak terlalu berpengaruh pada kinerja system [6]. *Client-Server* masih merupakan arsitektur yang dominan dalam industri komputer karena memiliki keunggulan dalam hal fleksibilitas, interoperabilitas, kinerja, ketahanan, distribusi dan skalabilitas dibandingkan komputer *stand-alone* maupun *mainframe* [7]

Selain infrastruktur dan kinerjanya harus efektif dan efisien, pengembangan aplikasi pun harus efektif dan efisien juga. Arsitektur *Model - View – Controller* (MVC) merupakan arsitektur terbaik untuk aplikasi desktop [8]. MVC merupakan arsitektur yang memisahkan dengan jelas antara data (*model*) dengan *user interface* (*view*), serta telah terbukti sangat efektif [8]. Penggunaan MVC memungkinkan penyusunan *source code* (kode sumber) aplikasi menjadi lebih rapi karena terjadi pemisahan dengan jelas antara *business logic* (logika bisnis) dengan *user interface*, sehingga dapat mengurangi kompleksitas *source code* (kode sumber), meningkatkan fleksibilitas dan modularitas perangkat lunak [9]. Penggunaan MVC membuat alur aplikasi lebih mudah dibaca dan lebih mudah ditelusuri jika terjadi kesalahan [10]. Berdasarkan hal-hal yang telah diuraikan di atas, maka perlu dilakukan penelitian mengenai "Implementasi Arsitektur *Client-Server* dan *Model-View-Controller* (MVC) untuk Membangun Aplikasi Administrasi Sekolah (Studi Kasus: SMK Averus Jakarta)".

2. Tinjauan Pustaka

2.1. Administrasi Sekolah

Administrasi adalah ilmu yang mempelajari proses kegiatan kerjasama untuk mencapai tujuan yang telah ditentukan. Kegiatan kerjasama itu sendiri merupakan gejala yang sifatnya universal dan memerlukan suatu proses pergerakan yang disebut dengan manajemen. Administrasi adalah keseluruhan proses kerjasama antara dua orang manusia atau lebih yang didasarkan atas rasionalitas tertentu untuk mencapai tujuan yang telah ditentukan sebelumnya [3]. Administrasi sekolah meliputi administrasi program pengajaran, administrasi kesiswaan, administrasi kepegawaian, administrasi keuangan dan administrasi perlengkapan atau barang.

2.2. Arsitektur Client-Server

Arsitektur *Client-Server* merupakan arsitektur yang paling baik untuk digunakan, sistem ini mampu menghasilkan aplikasi basis data yang tangguh dalam hal sekuritas, serta mampu mengurangi kepadatan lalu-lintas jaringan [5]. Arsitektur *Client-Server* merupakan model konektivitas pada

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**

jaringan yang membedakan fungsi komputer sebagai client dan server. Pendek kata, arsitektur *Client-Server* membagi beban kerja antara client dan server [6].

2.3. Model-View-Controller

Model-View-Controller (MVC) adalah pola desain atau arsitektur yang digunakan dalam rekayasa perangkat lunak, di mana terjadi pemisahan yang jelas antara data (model), dengan *user interface (view)* [8]. MVC adalah metode untuk membuat aplikasi dengan memisahkan data (model) dari tampilan *(view)* dan bagaimana memprosesnya *(controller)*. Jadi MVC merupakan metode dalam pemrograman yang memisahkan suatu program menjadi beberapa bagian, yaitu bagian model, bagian *view*, dan bagian *controller*. Biasanya bagian model diisi dengan fungsionalitas program tersebut, bagian view berisi tentang user interface dari program, dan bagian controller berisi tentang penanganan event dari program tersebut.

3. Analisa Sistem Saat Ini

3.1. Analisa Data dan Dokumen

Data-data yang dicatat dalam form pendaftaran adalah biodata calon siswa, nilai ujian nasional, peminatan/jurusan yang dipilih, dan lain-lain. Biodata calon siswa antara lain no. pendaftaran, nama lengkap, alamat, tempat dan tanggal lahir, asal sekolah, tahun lulus, dan lain-lain. Sedangkan data-data yang dicatat ketika melakukan pembayaran antara lain jenis pembayaran, besar pembayaran, dan tanggal pembayaran.

Untuk aplikasi yang akan dibuat, akan ditambahkan petugas/staf TU (Tata Usaha) yang menerima pembayaran. Pada pencatatan nilai, data-data yang dicatat adalah rincian dari penilaian siswa, yaitu nilai tugas, nilai ulangan, nilai ujian tengah semester, dan nilai ujian akhir semester.

3.2. Analisa Kebutuhan

Analisa kebutuhan merupakan langkah awal untuk menentukan gambaran perangkat lunak yang akan dihasilkan ketika pengembang melaksanakan sebuah proyek pembuatan perangkat lunak. Analisa kebutuhan adalah sebuah proses untuk mendapatkan informasi, model, spesifikasi tentang perangkat lunak yang diinginkan klien/pengguna.

Pada pengembangan aplikasi administrasi sekolah ini, spesifikasi dari perangkat lunak yang akan dibuat adalah:

1. Aplikasi dikembangkan menggunakan arsitektur *Client-Server* agar dapat digunakan oleh beberapa pengguna secara bersamaan sehingga dapat mempercepat pekerjaan.
2. Aplikasi dikembangkan menggunakan arsitektur MVC agar di masa depan mudah untuk dikembangkan kembali jika ada kebutuhan tambahan.
3. Aplikasi yang dikembangkan mencakup administrasi pendaftaran siswa, seleksi/penerimaan siswa, pembayaran, administrasi nilai, dan pengaturan hak akses pengguna.

3.3. Perancangan

Setelah dilakukan analisa terhadap kebutuhan dan data/dokumen, selanjutnya dibuat perancangan basis data dan perancangan aplikasi. Berdasarkan analisa kebutuhan yang telah dilakukan, maka dibuat ERD disajikan pada Gambar 1.

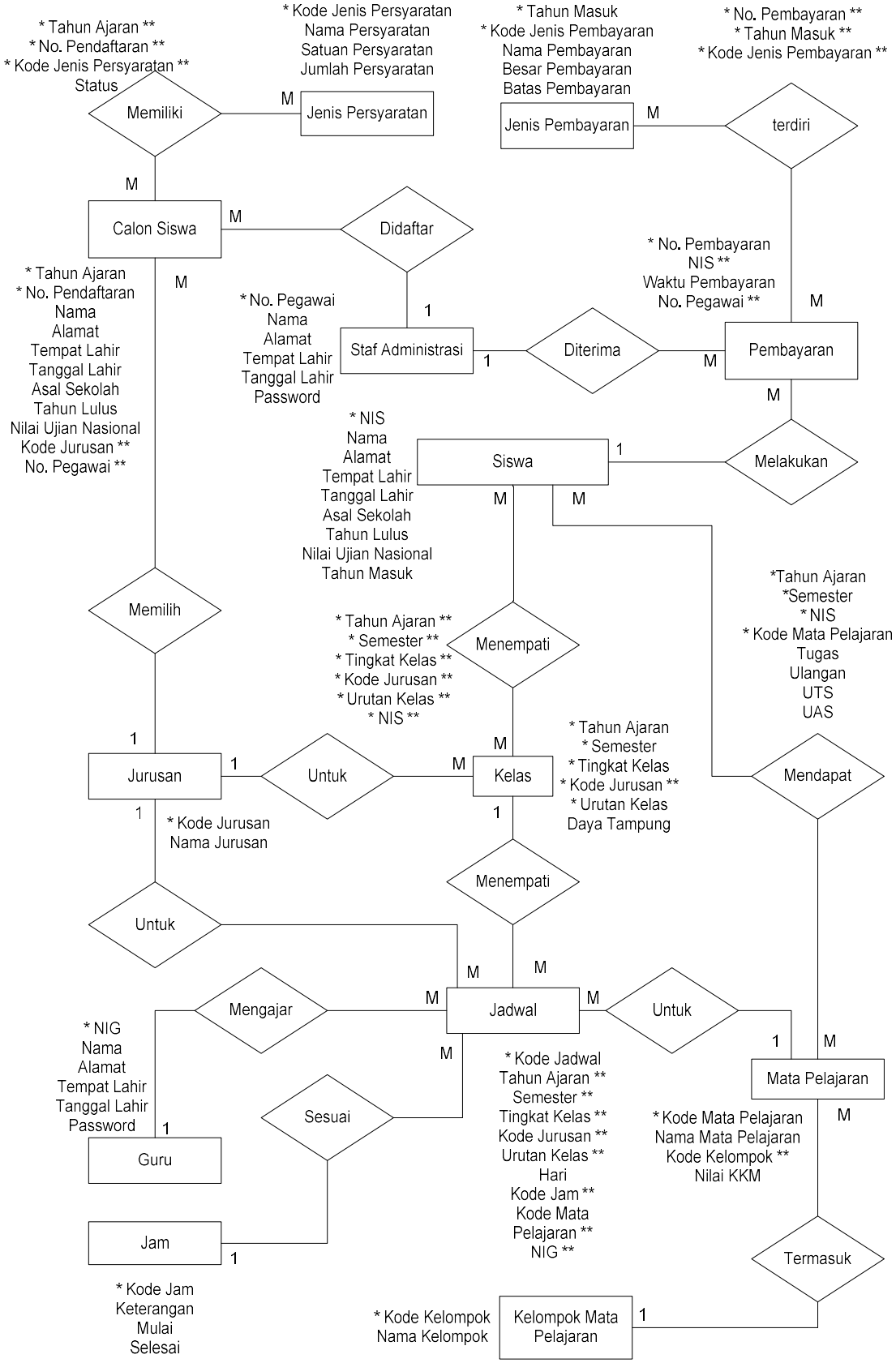
Hasil perancangan basis data berupa ERD (*Entity Relationship Diagram*) kemudian ditransformasi ke bentuk LRS (*Logical Record Structure*) disajikan pada Gambar 2. Setelah proses transformasi dari ERD ke LRS maka diperoleh diagram relasi yang disajikan pada Gambar 3.

3.4. Normalisasi

Normalisasi diperlukan jika tabel-tabel yang dibuat belum normal karena tabel-tabel yang dihasilkan dari transformasi ERD telah memenuhi syarat normal ke-3 (3NF), maka tabel-tabel tersebut tidak perlu dinormalisasi lagi.

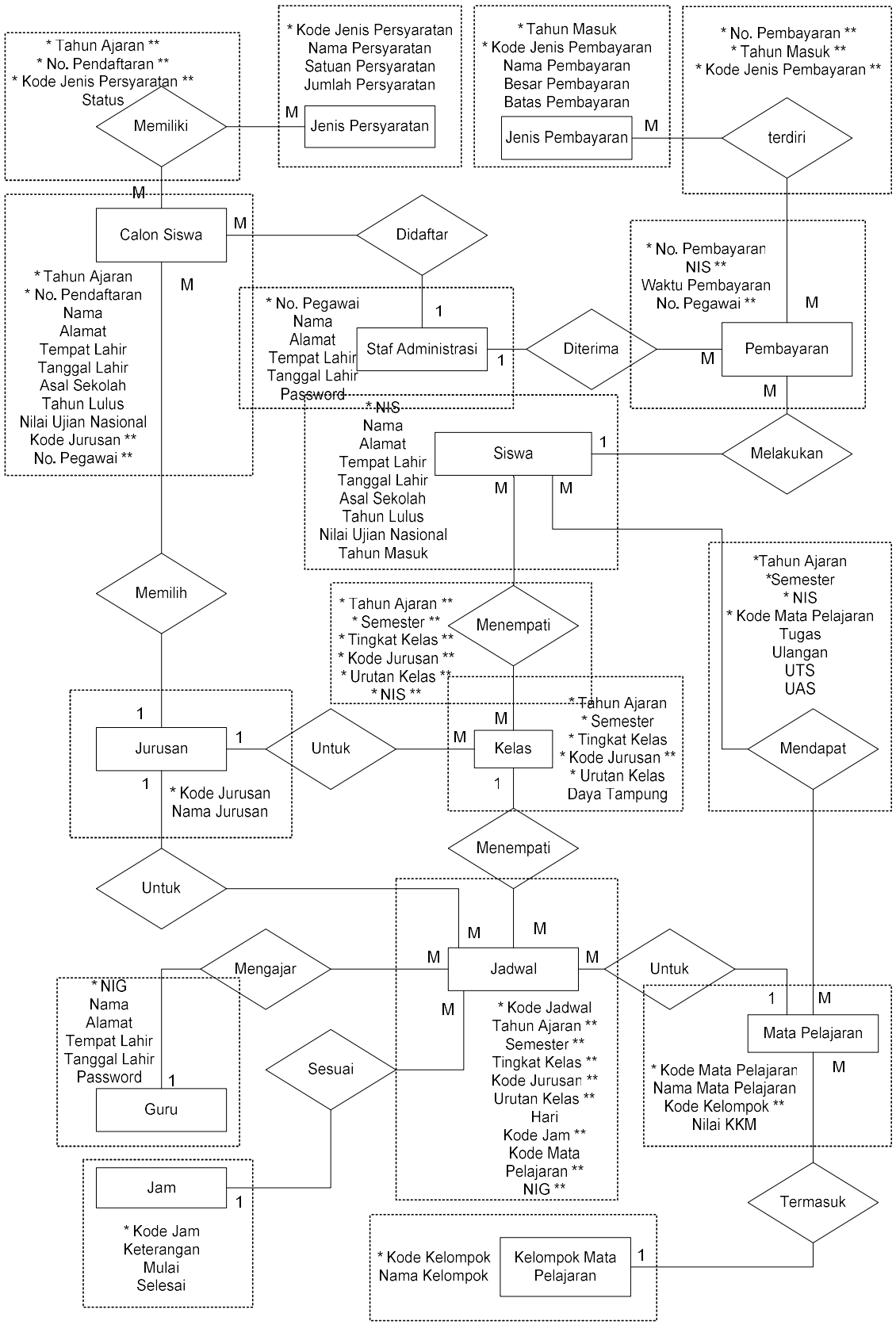
3.5. Spesifikasi Basis Data

Spesifikasi basis data meliputi Jenis Persyaratan, Jenis Pembayaran, Persyaratan, Pembayaran, Detail Pembayaran, Calon Siswa, Tabel Siswa, Tabel Staf Administrasi, Guru, Jurusan, Kelas, Kelas Siswa, Kelompok Mata Pelajaran, Mata Pelajaran, Jadwal, Jam, Nilai. Contoh tabel basis data dapat dilihat pada Tabel 1 – Table 4.



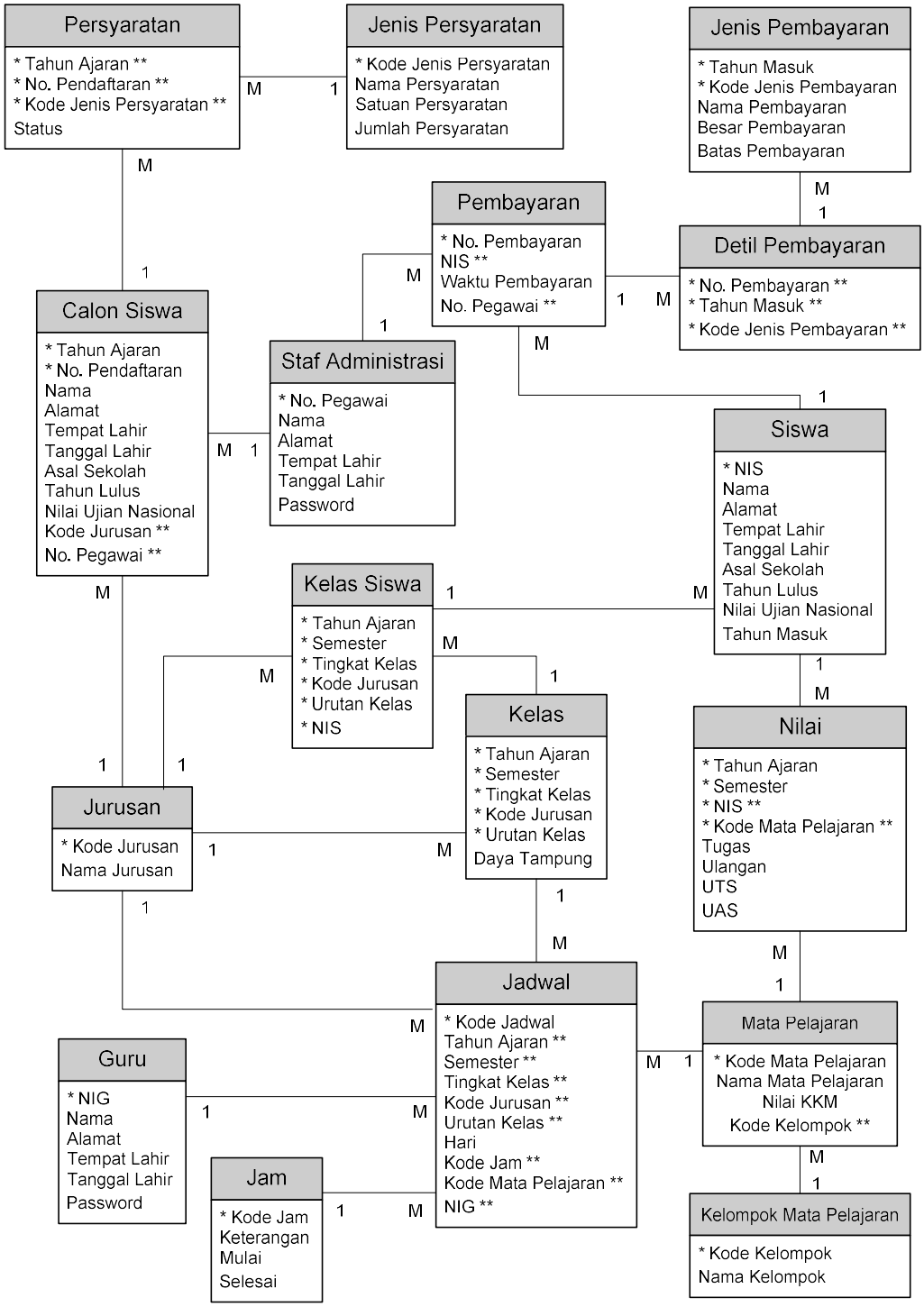
Gambar 1. ERD Basis Data Administrasi Sekolah

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**



Gambar 2. ERD ke LRS Basis Data Administrasi Sekolah

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**



Gambar 3. LRS Basis Data Administrasi Sekolah

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**

Tabel 1. Tabel Jenis Persyaratan

No	Nama Field	Tipe Field	Ukuran	Indeks
1	kodepersyaratan	Varchar	2	Primary
2	namapersyaratan	Varchar	30	
3	satuan	Varchar	10	
4	jumlah	Int		

Tabel 2. Tabel Jenis Pembayaran

No	Nama Field	Tipe Field	Ukuran	Indeks
1	tahunmasuk	Smallint	6	Primary
2	kodejenispembayaran	Varchar	5	Primary
3	namapembayaran	Varchar	30	
4	besarpembayaran	Int		
5	bataspembayaran	Date		

Tabel 3. Tabel Siswa

No	Nama Field	Tipe Field	Ukuran	Indeks
1	nis	Varchar	9	Primary
2	nama	Varchar	30	
3	alamat	Varchar	100	
4	tempatlahir	Varchar	40	
5	tanggallahir	Date		
6	asalsekolah	Varchar	60	
7	tahunlulus	Smallint		
8	nilaiujiannasional	Double		
9	tahunmasuk	Smallint		

Tabel 4. Tabel Jadwal

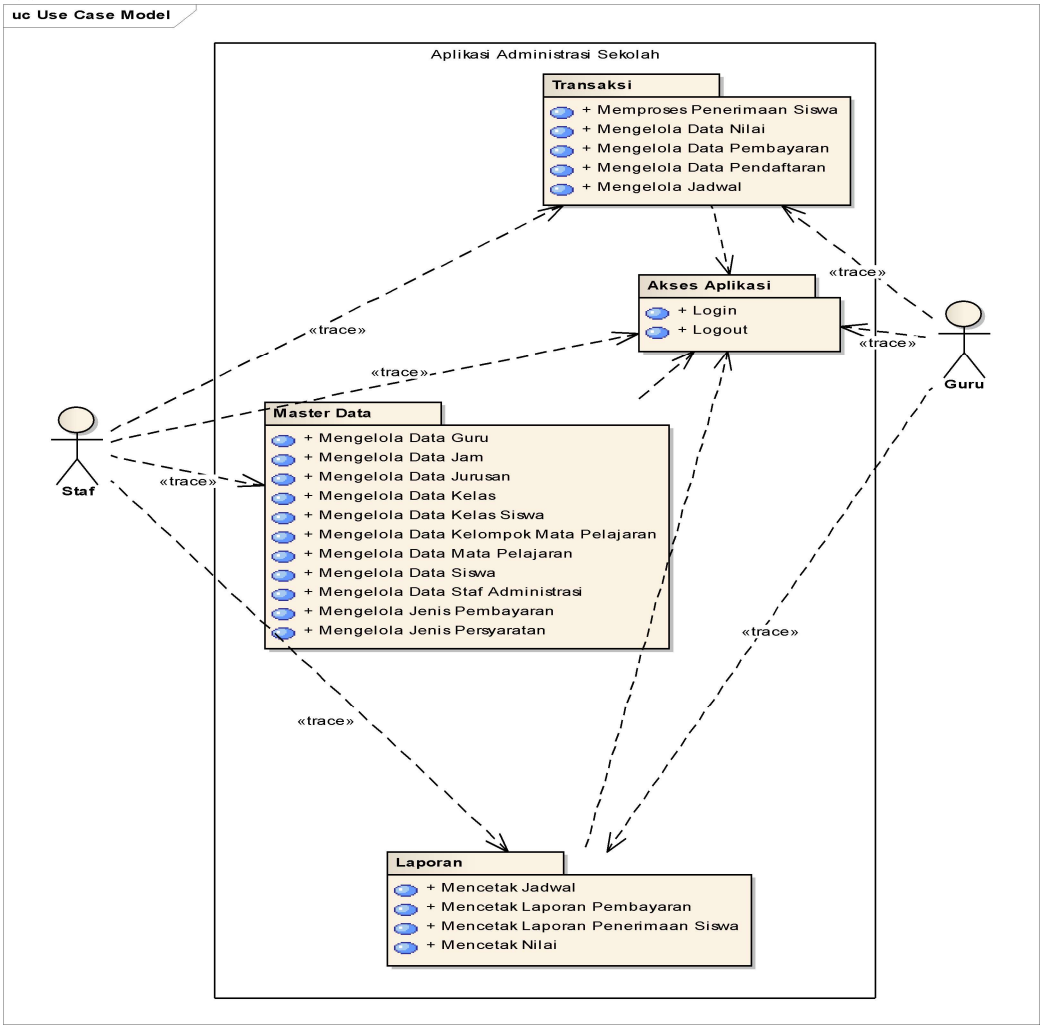
No	Nama Field	Tipe Field	Ukuran	Indeks
1	kodejadwal	Varchar	5	Primary
2	tahunajaran	Varchar	9	
3	semester	Tinyint		
4	tingkatkelas	Tinyint		
5	kodejurusan	Varchar	2	
6	urutankelas	Tinyint		
7	hari	Tinyint		
8	kodejam	Varchar	2	
9	kodematapelajaran	Varchar	5	
10	nig	Varchar	9	

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**

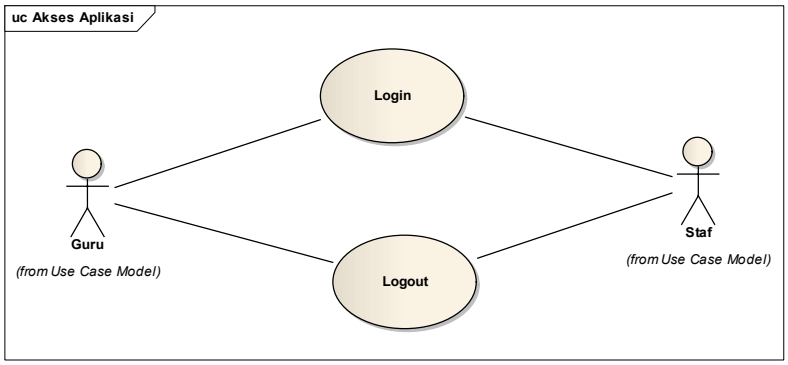
3.6. Perancangan Aplikasi

3.6.1. Use Case Diagram

Aplikasi administrasi yang dirancang menggunakan *use case diagram* seperti pada Gambar 4 sehingga Gambar 8.

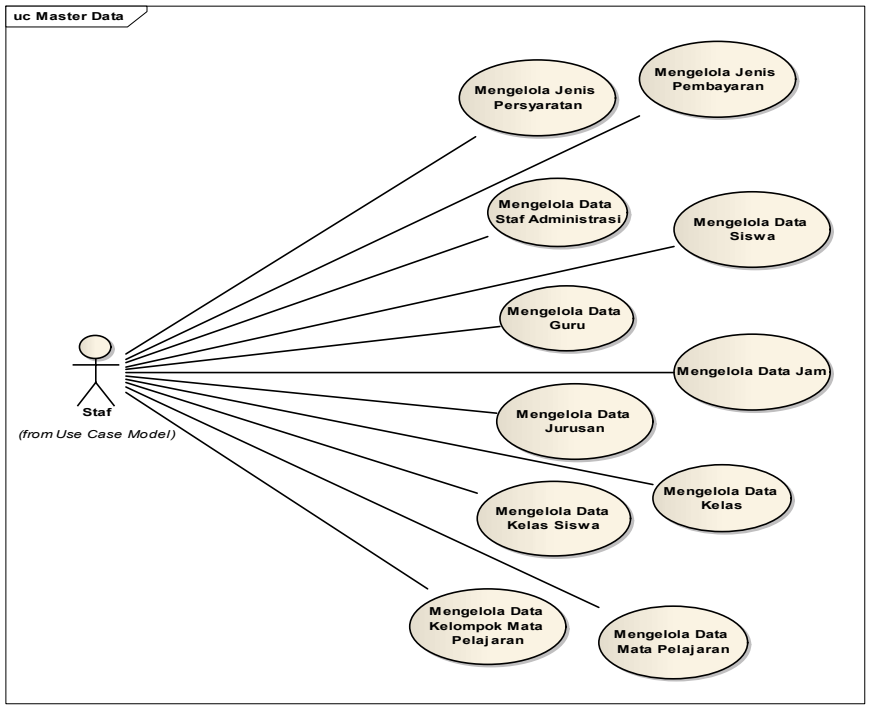


Gambar 4. Use Case Diagram Aplikasi Administrasi Sekolah

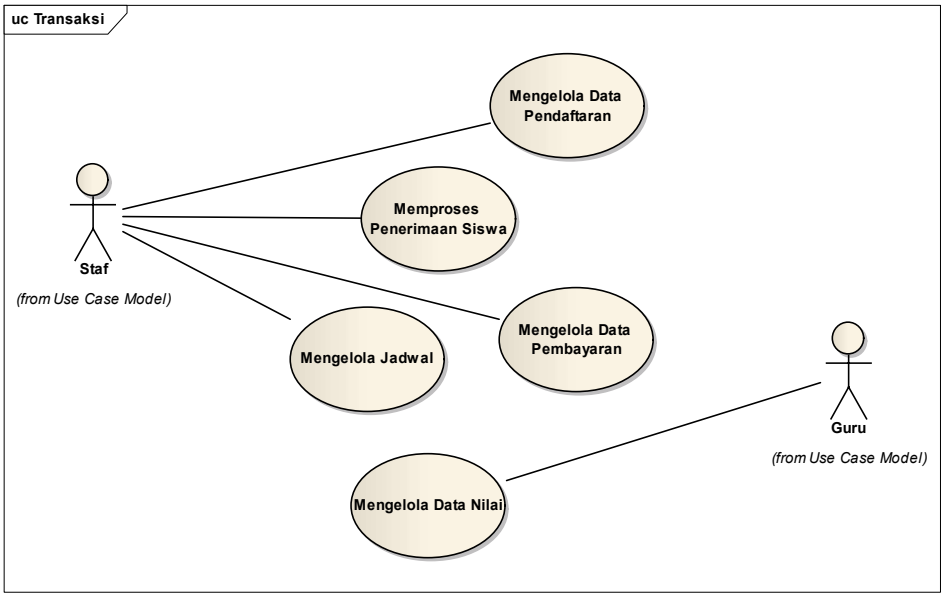


Gambar 5. Paket *Use Case Diagram* Akses Aplikasi

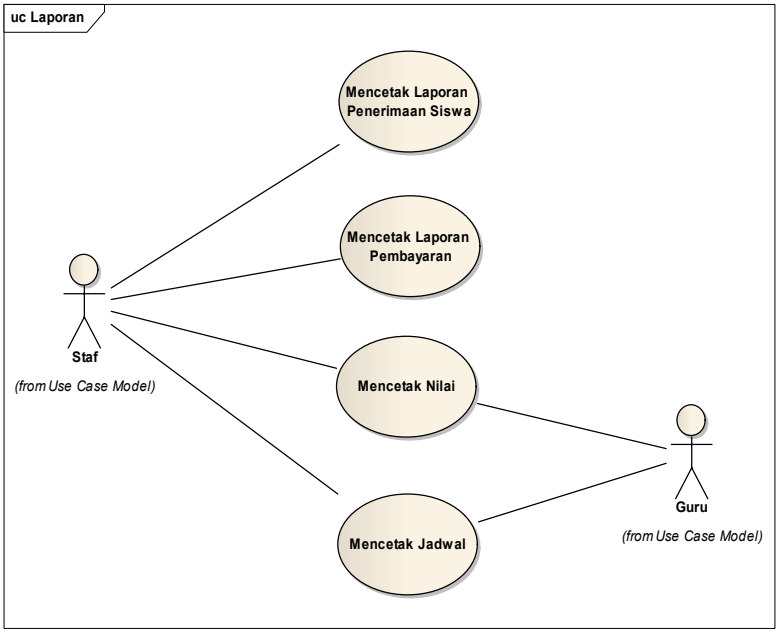
Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**



Gambar 6. Paket *Use Case Diagram* Master Data



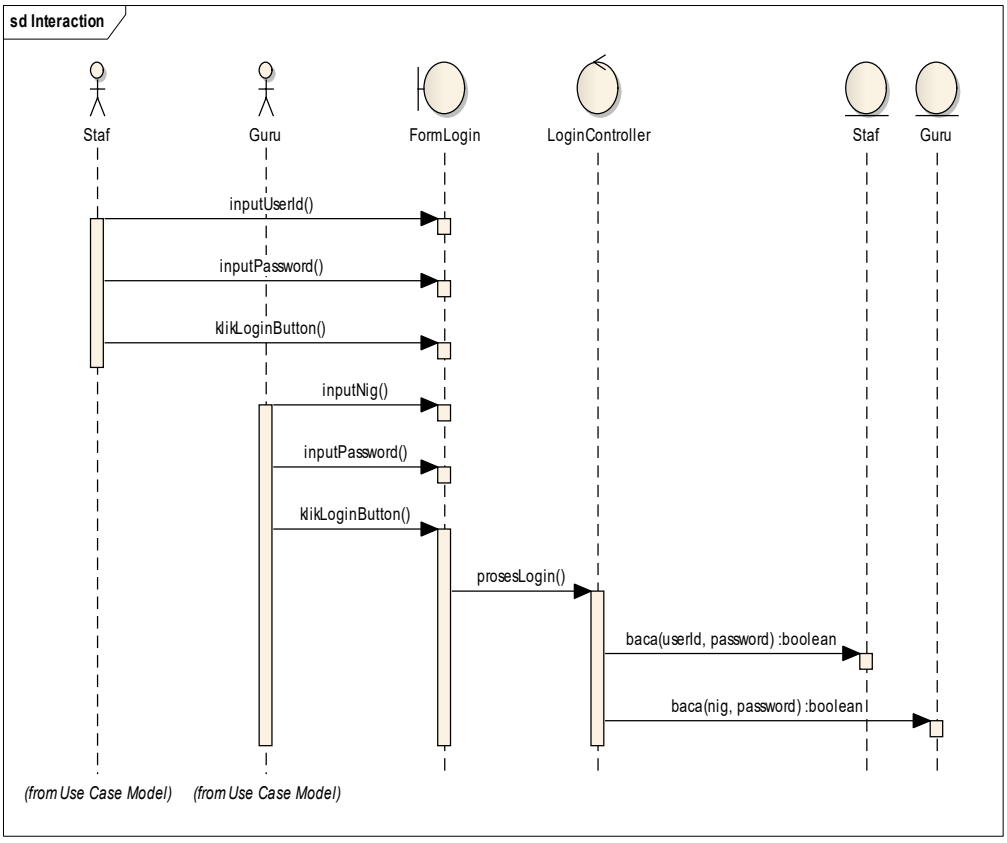
Gambar 7. Paket *Use Case Diagram* Transaksi



Gambar 8. Paket *Use Case Diagram* Laporan

3.6.2. Sequence Diagram

Setelah dibuat use case diagram, selanjutnya masing-masing use case dibuatkan sequence diagram.

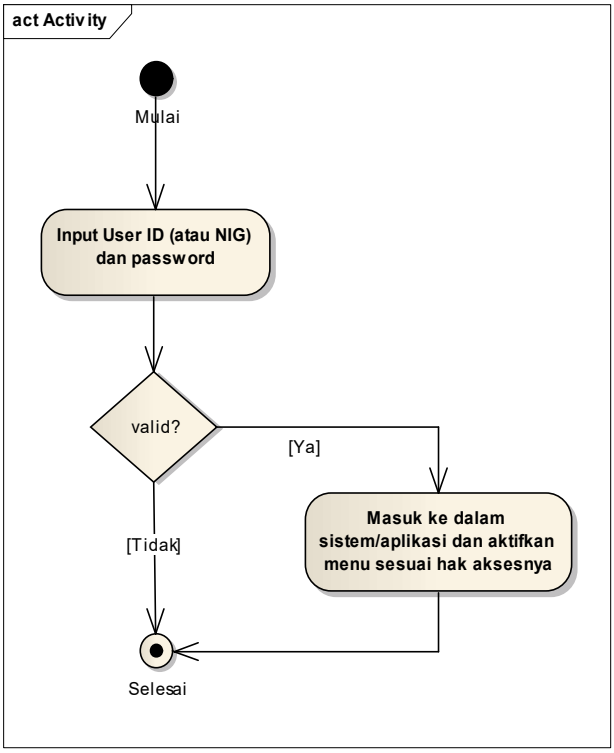


Gambar 9. *Sequence Diagram Login*

Sequence diagram dalam sistem ini meliputi Login, Logout, Mengelola Jenis Persyaratan, Mengelola Jenis Pembayaran, Mengelola Data Staf Administrasi, Mengelola Data Guru, Mengelola Data Siswa, Mengelola Kelompok Mata Pelajaran, Mengelola Mata Pelajaran, Mengelola Data Jam, Mengelola Data Jurusan, Mengelola Data Kelas, Mengelola Data Kelas Siswa, Mengelola Data Pendaftaran, Proses Penerimaan Siswa, Mengelola Data Pembayaran, Mengelola Jadwal, Mengelola Data Nilai, Mencetak Laporan Penerimaan Siswa, Mencetak Laporan Pembayaran, Mencetak Laporan Nilai, Mencetak Laporan Jadwal. Contoh sequence diagram login dapat dilihat pada Gambar 9.

3.6.3. Activity Diagram

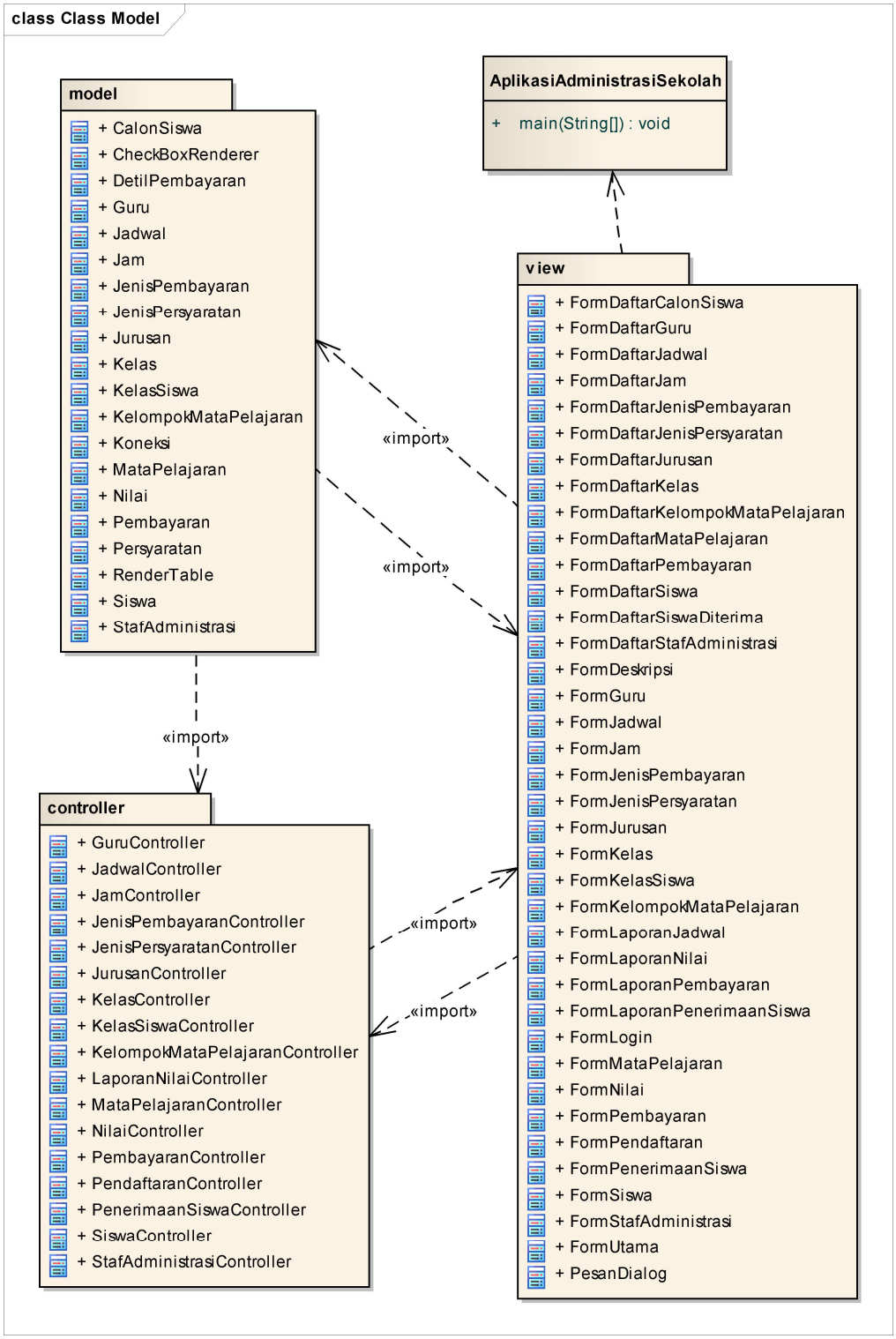
Masing-masing *use case* juga dibuatkan *activity diagram* untuk menunjukkan alur prosesnya. Activity diagram dalam sistem ini meliputi Login, Logout, Mengelola Jenis Persyaratan, Mengelola Jenis Pembayaran, Mengelola Data Staf Administrasi, Mengelola Data Guru, Mengelola Data Siswa, Mengelola Kelompok Mata Pelajaran, Mengelola Mata Pelajaran, Mengelola Data Jam, Mengelola Data Jurusan, Mengelola Data Kelas, Mengelola Data Kelas Siswa, Mengelola Data Pendaftaran, Proses Penerimaan Siswa, Mengelola Data Pembayaran, Mengelola Jadwal, Mengelola Data Nilai, Mencetak Laporan Penerimaan Siswa, Mencetak Laporan Pembayaran, Mencetak Laporan Nilai, Mencetak Laporan Jadwal. Contoh activity diagram login dapat dilihat pada Gambar 9.



Gambar 9. *Activity Diagram Login*

3.6.4. Class Diagram

Rancangan hubungan antar-*class* dibuat dalam diagram *class* disajikan pada Gambar 10.

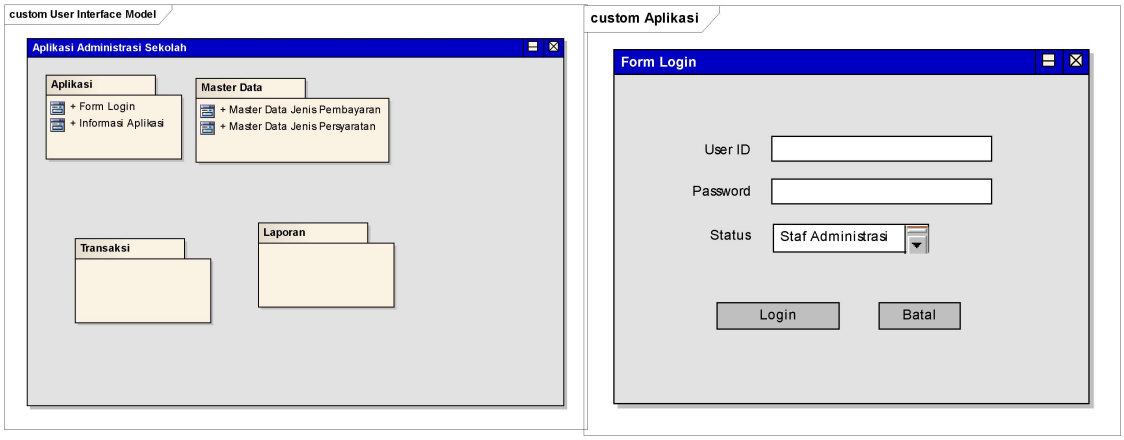


Gambar 10. *Class Diagram* Aplikasi Administrasi Sekolah

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**

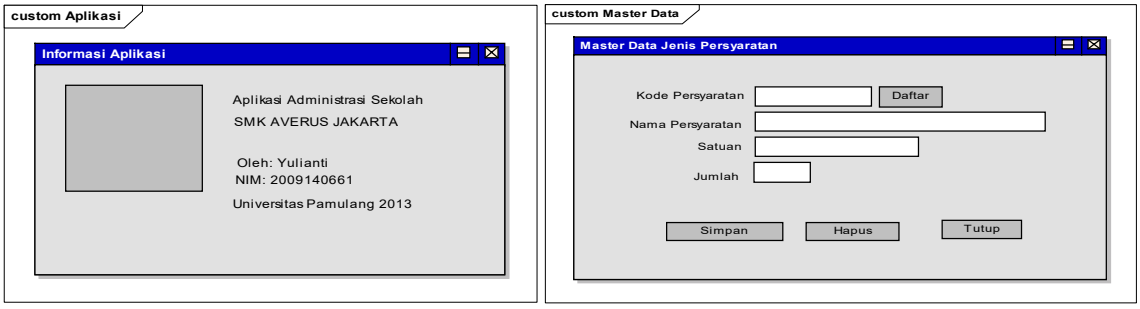
3.6.5. User Interface

Desain antarmuka form utama dan *login* disajikan pada Gambar 11.



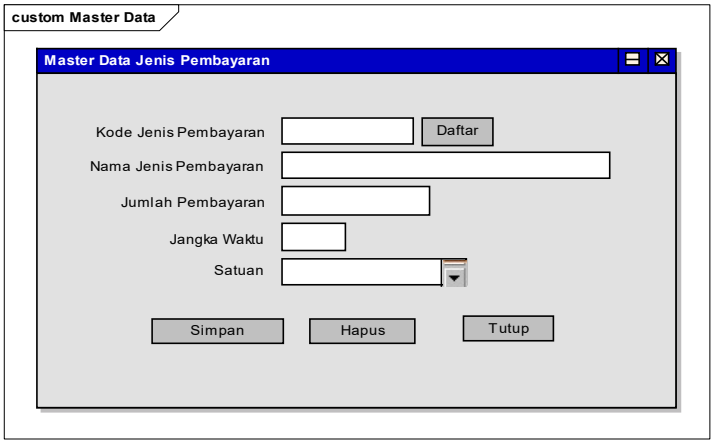
Gambar 11. Desain Antarmuka Form Utama dan *Login*

Desain antarmuka informasi aplikasi dan master data jenis persyaratan disajikan pada Gambar 12.



Gambar 12. Desain Antarmuka Informasi Aplikasi dan Master Data Jenis Persyaratan

Desain antarmuka master data jenis pembayaran disajikan pada Gambar 13.

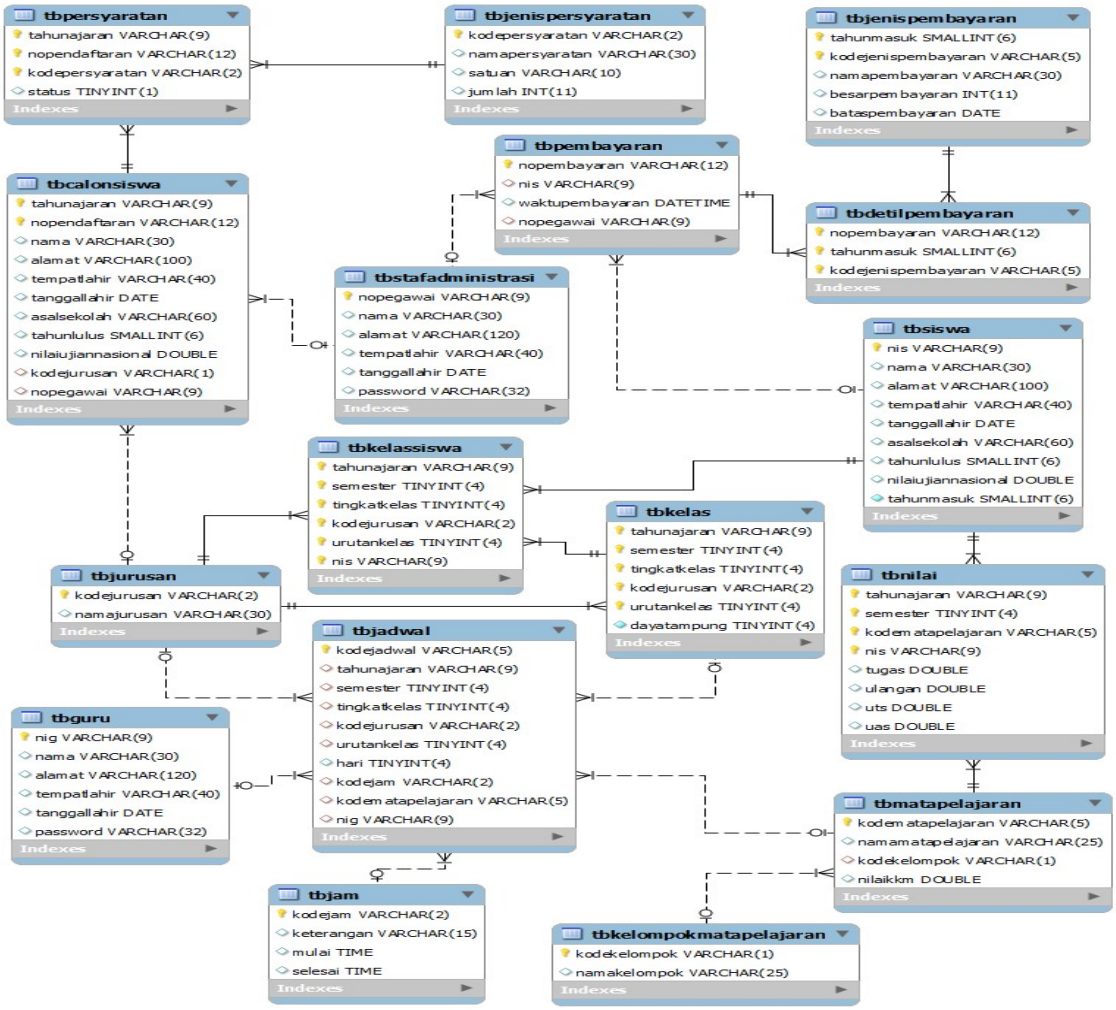


Gambar 13. Desain Antarmuka Master Data Jenis Pembayaran

4. Implemantasi dan Pengujian

4.1. Implementasi Basis Data

Rancangan basis data diimplementasikan pada DBMS (*Database Management System*) MySQL dan dihasilkan diagram relasi yang disajikan pada Gambar 14.



Gambar 14. Diagram Relasi Basis Data Administrasi Sekolah

4.2. Implementasi Aplikasi

Rancangan yang telah dibuat diimplementasikan menggunakan bahasa pemrograman Java menggunakan *Integrated Development Environment* (IDE) NetBeans, sehingga dihasilkan Tampilan form deskripsi dan form *Login* yang disajikan pada Gambar 15.

Gambar 15. Tampilan Form Deskripsi dan *Login*

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**

4.3. Pengujian

Pengujian telah dilakukan menggunakan metode *black box* dan *white box* selama pengembangan aplikasi, pengujian dilakukan untuk memastikan bahwa rancangan algoritma dan implementasi pada *source code* program telah sesuai dengan tujuan yang telah ditentukan. Pengujian telah dilakukan baik per blok *source code* program maupun per modul. *Bug* (cacat) yang ditemukan pada aplikasi langsung diperbaiki agar sesuai dengan proses yang diinginkan.

4.3.1. Pengujian dengan Black Box

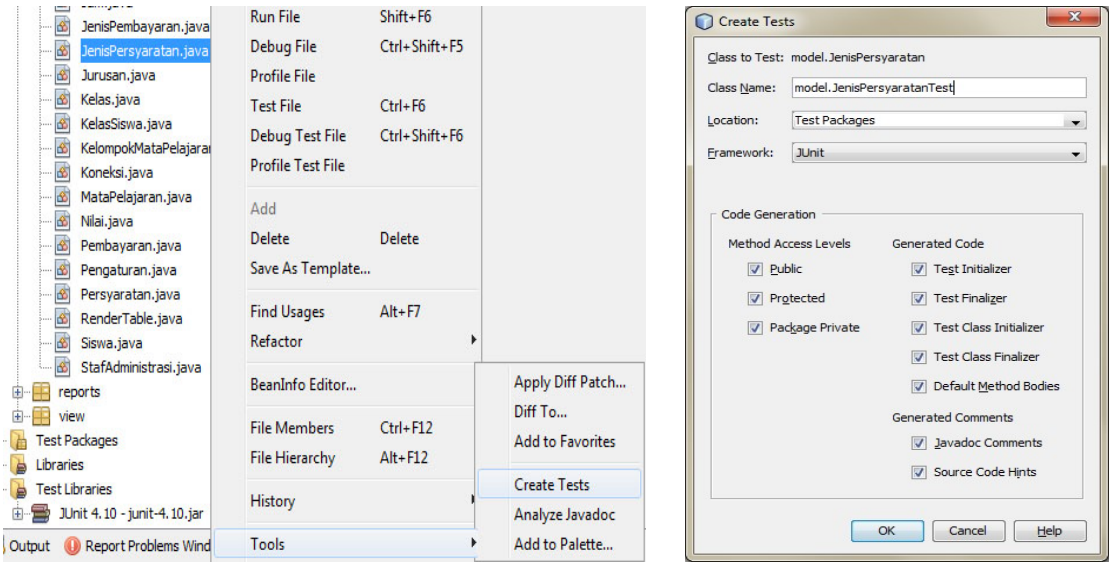
Pengujian *black box* dilakukan dengan menguji fungsionalitas (*functional testing*) aplikasi. Pengujian dilakukan dengan mengecek fungsionalitas pada bagian tertentu sesuai Table 5.

Tabel 5. Pengujian Fungsionalitas

No.	Skenario Pengujian	Uji Kasus	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Memilih (mengklik) menu login	Klik menu item login	Menampilkan form login	Sesuai harapan	valid
2	Mengosongkan form login, kemudian mengklik button login	Jabatan dipilih staf administrasi, no. pegawai kosong, password kosong	Menampilkan keterangan no. pegawai harus diisi	Sesuai harapan	valid
3	Melakukan login dengan memilih jabatan staf dengan data yang benar	Jabatan dipilih staf administrasi, no. pegawai diisi ADMIN, dan password diisi ADMIN	Berhasil login dan menu untuk staf administrasi diaktifkan	Sesuai harapan	valid
4	Mengosongkan kode persyaratan pada form master data jenis persyaratan, kemudian mengklik tombol simpan	Kode persyaratan = ””	Aplikasi menampilkan pesan “Kode persyaratan tidak boleh kosong”	Sesuai harapan	Valid
5	Mengosongkan kode persyaratan pada form master data jenis persyaratan, kemudian mengklik tombol hapus	Kode persyaratan = ””	Aplikasi menampilkan pesan “Kode persyaratan tidak boleh kosong”	Sesuai harapan	Valid

4.3.2. Pengujian dengan White Box

Pengujian *white box* dilakukan dengan menguji unit (*unit testing*) untuk memastikan bahwa modul/unit dapat bekerja sesuai harapan. Pengujian dilakukan menggunakan fasilitas yang tersedia pada IDE NetBeans. Tahap pertama dengan membuat *test* (pengujian) dan menentukan *test* (pengujian) yang disajikan pada Gambar 16.



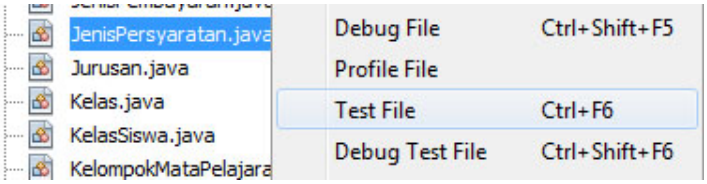
Gambar 16. Membuat Pengujian dan Menentukan Pengujian Unit (*White Box*)

Tahapan selanjutnya yaitu mengubah kondisi *test* (pengujian) yang disajikan pada Gambar 17.

```
@Test
    public void testGetKodePersyaratan() {
        System.out.println("getKodePersyaratan");
        JenisPersyaratan instance = new JenisPersyaratan();
        String expResult = null;
        String result = instance.getKodePersyaratan();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove
the default call to fail.
        //fail("The test case is a prototype.");
    }
```

Gambar 17. Mengubah Kondisi Pengujian

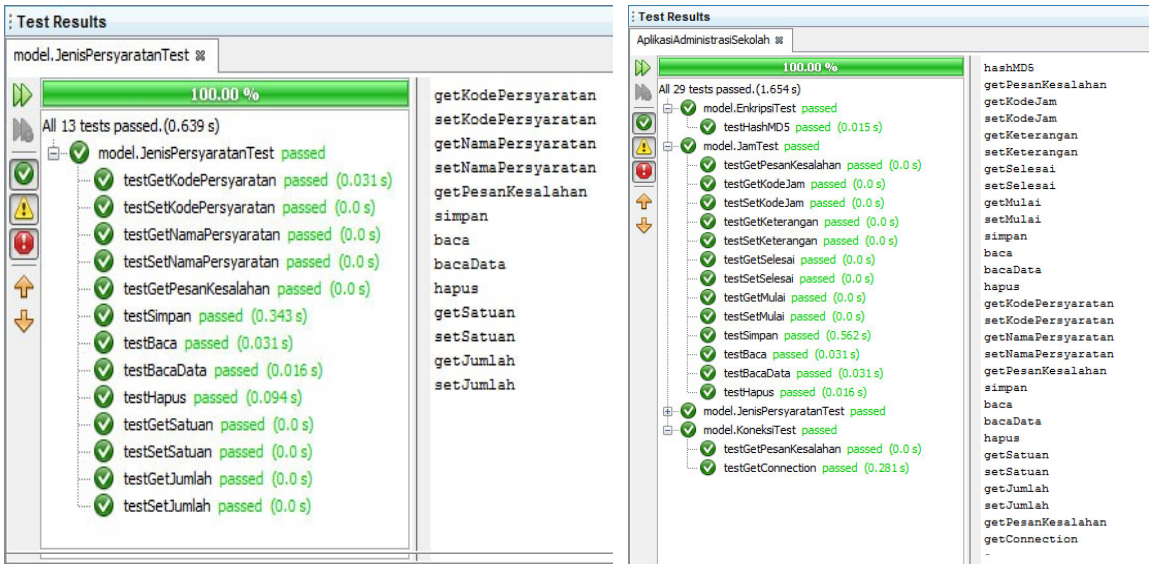
Variabel expResult diisi dengan nilai yang diharapkan setelah metode diproses, sedangkan variabel result adalah nilai setelah metode diproses. Jika kedua variabel bernilai sama, maka dianggap metodenya benar (sesuai yang diharapkan). Setelah mengubah kondisi *test,* maka selanjutnya melakukan *test* (pengujian) seperti yang disajikan pada Gambar 18.



Gambar 18. Melakukan Pengujian Unit (*White Box*)

Journal of Engineering, Technology & Applied Science, vol. 2, no. 1, pp. 1-18, April 2020. **DOI: 10.36079/lamintang.jetas-0201.91**

Tahapan berikutnya yaitu menampilkan hasil pengujian dan hasil pengujian empat unit/modul unit (*white box*) disajikan pada Gambar 19.



Gambar 19. Hasil Pengujian Unit dan Pengujian Empat Unit/Modul Unit (*White Box*)

Jika hasil pengujian tidak menampilkan kesalahan seperti berarti unit/metode yang diuji tidak ada kesalahan.

5. Analisa

Setelah aplikasi yang dibuat diimplementasikan pada jaringan LAN (*Local Area Network*) menggunakan satu server basis data yaitu MySQL, aplikasi dapat digunakan secara bersamaan dan data yang tersimpan dapat digunakan secara bersamaan, sehingga ketika meng-*input* data dapat lebih cepat. Aplikasi yang telah dibuat menggunakan arsitektur MVC (*Model-View-Controller*) dapat memperpendek *source code* program karena beberapa modul (metode atau *class*) dapat digunakan modul lain tanpa mengetik ulang. Misalnya, jika tidak menggunakan arsitektur MVC, maka *source code* untuk membaca atau menampilkan data siswa harus diketik di modul form siswa dan modul form pembayaran. Tetapi jika menggunakan arsitektur MVC, maka *source code* tersebut cukup diketik di satu modul, kemudian modul lain tinggal memanggil nama metodenya. Penggunaan arsitektur MVC tidak harus mengubah bagian model (modul yang berhubungan dengan *database*) ketika melakukan perubahan *source code* pada modul *user interface* (antarmuka pengguna). Hal ini dapat mempermudah untuk menyesuaikan dengan kebutuhan di masa depan.

6. Kesimpulan

Dari hasil implementasi dan analisa, dapat diambil kesimpulan bahwa aplikasi administrasi sekolah yang dibangun dengan arsitektur *Client-Server* dapat mengurangi duplikasi data, karena semua data disimpan dalam basis data yang terpusat dan digunakan dengan sistem berbagi. Aplikasi administrasi sekolah yang telah dibuat dapat memperbaiki pengorganisasian dokumen-dokumen di sekolah, semua data disimpan dalam basis data terpusat, data/dokumen yang diperlukan dapat dilihat, dan dianalisa tanpa harus dicetak. Aplikasi administrasi sekolah dapat mempercepat pembuatan laporan-laporan yang diperlukan dari data-data administrasi, karena untuk membuat laporan, tinggal memilih menu laporan yang diinginkan, dan spesifikasi laporan yang akan dicetak. Selain itu, aplikasi yang dibuat dengan arsitektur MVC menjadi lebih rapi, dapat memperpendek *source code* aplikasi, dan mudah disesuaikan dengan kebutuhan di masa depan. Jika ada perubahan pada tampilan antarmuka, maka tidak perlu mengubah *source code* yang berhubungan dengan basis data.

Daftar Pustaka

[1] Sulindawaty, "Pembuatan Perangkat Lunak Penyimpanan Data Rahasia dengan Menggunakan Teknik Steganography untuk Media Citra Digital," *Jurnal Saintkom,* vol. 10, no. 3, pp. 155-173, 2011.

[2] Redita, N. Gede and T. Kristanti, "Sistem Informasi Nilai SMPN 14 Bandung," *Jurnal Sistem Informasi,* vol. 7, no. 1, pp. 56-67, 2012.

[3] H. Antonio and N. Safriadi, "Rancang Bangun Sistem Informasi Administrasi Informatika (SI-ADIF)," *Jurnal ELKHA,* vol. 4, no.2, pp. 12-15, 2012.

[4] D. Widhyaestoeti, "Rancang Bangun Database Nilai Siswa Tingkat Sekolah Menengah," *Jurnal Ilmiah Teknologi dan Informasi*, vol. 2, no. 5, pp. 4-18, 2011.

[5] D. D. Prasetyo, *Aplikasi Database Client/Server Menggunakan Delphi dan MySQL*. Jakarta: PT.Elex Media Komputindo, 2004.

[6] M. Alam, and J. Agus, *Pemrograman Database Visual Basic dalam SQL Server 7.0 & 2005*. Jakarta: Elex Media Komputindo, 2005.

[7] M. V. Vlugt, and S. Sambasivam, "Redesign of Stand-Alone Applications into Thin-Client/Server Architecture," *Issues in Informing Science and Information Technology*, pp. 723-742, 2005.

[8] N. Widiyanto, *Membangun Aplikasi Java Enterprise dengan Arsitektur Model View Controller (MVC).* Yogyakarta: Andi Offset, 2010.

[9] S. Uyun, and M. R. Ma'arif, "Implementation of Model View Controller (MVC) Architecture on Building Web-based Information System," *Seminar Nasional Aplikasi Teknologi Informasi 2010 (SNATI 2010)*, pp. 47-50, 2010.

[10] D. Martha, C. Harianto, and M. Asfi, "Metode MVC untuk Perancangan Sistem Berorientasi Objek pada Ujian Saringan Masuk Penerimaan Mahasiswa Baru di STMIK CIC Cirebon," *Jurnal Informatika*, vol.6, no.2, pp. 145 – 160, 2010.