**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Pemanfaatan teknologi informasi dan penggunaan teknologi sistem informasi dapat mempengaruhi kualitas organisasi perguruan tinggi. Menurut Arifin(2012) bahwa sistem informasi menjadi salah satu tolak ukur tingkat keberhasilan dalam penyelenggaraan pendidikan. Salah satu perguruan tinggi yang telah memanfaatkan teknologi informasi adalah Universitas Nasional Pasim.

Universitas Nasional Pasim telah menggunakan teknologi informasi untuk menunjang berbagai aktifitas di universitas, seperti aktifitas akademik, perpustakaan, keuangan, kepegawaian, dan yang lainnya. Tujuan digunakannya sistem informasi di Universitas Nasional Pasim ialah agar berbagai aktifitas yang dilakukan di universitas dapat berjalan dengan lancar, efektif, dan efisien. Namun dengan bertambahnya waktu, maka data-data yang ada akan bertambah pula, sehingga akan mengakibatkan proses analisa suatu informasi menjadi lebih sulit dilakukan dan dikelola oleh para pimpinan universitas. Disamping itu, sistem informasi yang ada di Universitas Nasional Pasim belum terintegrasi dengan baik dan data-data yang digunakan untuk proses laporan memiliki format data yang berbeda, sehingga membutuhkan waktu yang lebih lama untuk membuat sebuah laporan, seperti untuk penyusunan borang akreditasi.

Status akreditasi suatu program studi merupakan cerminan kinerja program studi yang bersangkutan dan menggambarkan mutu dan efisiensi suatu program studi yang diselenggarakan. Untuk mendapatkan status akreditasi, diperlukan informasi mengenai program studi yang didapatkan dari pengisian penilaian akreditasi dengan terlebih dahulu membuat dokumen evaluasi diri program studi. Sarana yang digunakan untuk mengumpulkan informasi tersebut adalah borang (formulir) akreditasi. Masing-masing program studi yang ada di Universitas Nasional Pasim pun tidak luput dari upaya memperoleh status akreditasi yang baik yang dilakukan pada periode tertentu. Namun, hingga saat ini untuk penyusunan borang akreditasi di Universitas Nasional Pasim masih dilakukan secara manual dalam pengelolaan informasinya, sehingga Tim penyusun borang akreditasi mengalami kesulitan untuk mendapatkan data yang spesifik jika data yang ada masih tersebar dan belum terintegrasi dengan baik. Selain itu, akan membutuhkan waktu yang cukup lama untuk mengumpulkan dan menganalisis data sesuai dengan format borang akreditasi yang telah ditetapkan.

Berdasarkan permasalahan yang telah penulis kemukakan diatas, penulis bermaksud melakukan penelitian untuk membangun sebuah aplikasi yang dapat membantu proses penyusunan borang akreditasi .Hasil penelitian ini akan penulis tulis dalam bentuk laporan tugas akhir yang berjudul “**Pembangunan Aplikasi Pengolahan Data Borang Akreditasi di Universitas Nasional Pasim**”.

**1.2 Identifikasi Masalah**

Berdasarkan latar belakang masalah yang telah dikemukakan, maka penulis mengidentifikasi masalah-masalah yang akan diteliti, yaitu :

1. Bagaimana merancang aplikasi yang dapat melakukan integrasi dengan aplikasi pendukung yang sudah ada untuk mendapatkan data kebutuhan pengisian borang akreditasi.
2. Bagaimana merancang aplikasi yang dapat mengolah data kebutuhan borang akreditasi dan menyajikannya sesuai dengan format data borang akreditasi.

**1.3 Maksud dan Tujuan**

### Maksud

Adapun maksud dari penyusunan tugas akhir ini adalah membangun aplikasi yang dapat memudahkan Tim Penyusun Borang Akreditasi dalam mendapatkan dan mengumpulkan data sesuai dengan format data borang akreditasi yang telah ditetapkan oleh BAN-PT dengan tepat waktu.

### Tujuan

Adapun tujuan dari penelitian ini adalah :

1. Untuk membangun aplikasi yang dapat berintegrasi dengan aplikasi pendukung yang sudah ada untuk mendapatkan data kebutuhan pengisian borang akreditasi.
2. Untuk membangun aplikasi yang dapat yang dapat memperoleh data kebutuhan borang akreditasi dan menyajikannya sesuai dengan format data borang akreditasi.

**1.4 Batasan Masalah**

Batasan masalah berguna agar pembahasan dan analisis yang dilakukan penulis dapat terarah sesuai dengan tujuan penulisan, maka berdasarkan latar belakang masalah dan identifikasi masalah diatas, penulis membatasi permasalahan sebagai berikut:

1. Aplikasi ini terintegrasi dengan aplikasi lain seperti aplikasi akademik, aplikasi perpustakaan, aplikasi kepegawaian, aplikasi keuangan, dan lainnya.
2. Data yang diolah adalah data pendukung untuk pengisian borang akreditasi.
3. Aplikasi ini akan menampilkan setiap item data yang dibutuhkan sesuai dengan format data borang akreditasi BAN-PT.
4. Bahasa pemrograman yang digunakan adalah PHP (*Personal Home Page*).

## Kegunaan Penelitian

Adapaun kegunaan yang ingin dicapai dengan dibangunnya aplikasi ini adalah sebagai berikut :

1. Bagi Penulis

Untuk memenuhi salah satu syarat kelulusan Ahli Madya (D3) Program Studi Manajemen Informatika Fakultas Ilmu Komputer Universitas Nasional Pasim Bandung, serta menambah pengetahuan bagi penulis tentang pengaplikasian ilmu teknologi yang didapat di bangku kuliah dalam kehidupan sehari – hari.

1. Bagi Penyusun Borang

Membantu kinerja Tim Penyusun Borang dalam proses pengumpulan data-data yang dibutuhkan sebagai bahan informasi Borang Akreditasi Program Studi maupun Akreditasi Perguruan Tinggi.

1. Bagi Instansi

Penyusunan Borang Akreditasi Program Studi yang ada di Universitas Nasional Pasim dapat dilaksanakan dengan tepat waktu.

1. Bagi Pihak Lain

Sebagai salah satu referensi untuk perkembangan ilmu pengetahuan, sehingga diharapkan pembangunan aplikasi ini dapat dikembangkan guna penelitian selanjutnya.

## Model Penelitian

### Metode Pengumpulan Data

Metode pengumpulan data dilakukan melalui dua cara yaitu :

1. Pengambilan data secara langsung (*Fields Research*)

Untuk pelaksanaan pengambilan data secara langsung ini menggunakan 2 cara yaitu :

1. Observasi Langsung, yaitu teknik pengumpulan data dengan cara mengamati langsung operasi maupun prosedur yang berlaku pada objek penelitian.
2. Wawancara (*interview*),yaitu teknik pengempulan data dengan mengadakan wawancara dengan pihak-pihak terkait guna mendapatkan data-data serta keterangan yang dibutukan oleh penulis. Dalam hal ini penulis melakukan tanya jawab dengan pemilik perusahaan secara langsung dan staff gudang terkait dengan prosedur sistem yang digunakan sehingga diharapkan dapat memberikan solusi terbaik dalam pembangunan sistem informasi.
3. Studi Literatur (Studi Pustaka)

Selain pengambilan data secara langsung, penyusun juga melakukan studi literatur demi tercapainya tujuan pembangunan sistem informasi sehingga dapat memenuhi kriteria pembanguan sistem informasi yang baik dan sistematis. Studi literatur ini yaitu suatu metode pengumpulan data dengan membaca dan mempelajari buku, literatur, artikel, karya ilmiah, catatan perkuliahan, dan sumber-sumber lainnya yang berkaitan dengan topik penelitian. Studi literatur ini dimaksudkan penyusun untuk membandingkan serta mencocokkan antara fakta yang terjadi di dunia nyata dengan aspek-aspek yang ada dalam pembangunan sistem informasi.

### Model Proses

Model proses yang digunakan untuk pembangunan sistem informasi ini adalah model proses ***Prototyping***. *Prototyping* model adalah model yang dapat diterapkan pada model apapun. Model ini tidak memerlukan data yang lengkap dari sisi klien*.* Terkadang klien hanya memberikan beberapa kebutuhan umum *software* tanpa detil input, proses atau detil output. Model ini tepat digunakan jika pihak klien menginginkan *prototype* dari *software* dalam waktu yang singkat. Dan *prototype* inilah yang akan menjadi acuan dari *client* untuk memberikan data kebutuhan yang lebih lengkap pada pembuat *software(developer).*

Berikut adalah penjelasan dari tahap-tahap pemodelan proses *Prototype* :

1. Mengidentifikasi kebutuhan pemakai

Pengembang mewawancarai pemakai untuk mendapatkan gagasan dari apa yang diinginkan pemakai terhadap sistem.

1. Membangun *Prototype*

**Pengembang membangun *prototyping* dengan membuat perancangan sementara yang berfokus pada penyajian kepada pemakai (misalnya dengan membuat format *input* dan *output*).**

1. Evaluasi *Prototype*

Evaluasi dilakukan oleh pemakai dengan menilai *prototype* yang telah dibuat oleh pengembang apakah sudah sesuai dengan keinginan pemakai. Jika sudah sesuai maka langkah selanjutnya dilakukan tetapi jika tidak sesuai maka *prototype* akan diperbaiki kembali dengan mengulang langkah sebelumnya.

1. Pengkodean

**Dalam tahap ini *Prototype* yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.**

1. Menguji Sistem

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, maka harus di tes dahulu sebelum digunakan. Pengujian ini dilakukan dengan *Black Box*.

1. Evaluasi Sistem

Pada tahap ini, pemakai mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan. Jika sudah sesuai maka langkah selanjutnya dilakukan, akan tetapi jika tidak maka akan diperbaiki kembali dengan mengulang langkah Pengkodean dan Menguji Sistem.

1. Menggunakan Sistem

Perangkat lunak yang telah diuji dan diterima pemakai siap untuk digunakan.



**Gambar 1.1** Metode *Prototype*

(Sumber: Raymond Mc.Leod, Jr, 2001)

**1.6.3 Pendekatan Sistem**

Pendekatan sistem yang digunakan dalam pembangunan sistem informasi ini yaitu Teknik Pembangunan Sistem dengan Metoda pendekatan *Object Oriented Analisys Design (OOAD)*. OOAD(*Object Oriented Analisys Design*) adalah metode analisis yang memeriksa kebutuhan-kebutuhan pada suatu kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarah pada arsitektur Software yang didasarkan pada manipulasi objek-objek *system* atau subsitem. Terdapat beberapa konsep dalam OOAD yaitu : kelas, object, metode, atribut, abstaksi,dll.

Alat bantu yang penulis pilih untuk digunakan dalam pengembangan sistem adalah UML (*Unified Modeling Language*). UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat rancangan atas visinya dalam bentuk yang baku.

Jenis-jenis Diagram UML :

1. Use Case Diagram

*Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng- *create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng- *include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di- *include* akan dipanggil setiap kali *use case* yang meng- *include* dieksekusi secara normal. Sebuah *use case* dapat di- *include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common* . Sebuah *use case* juga dapat meng- *extend use case* lain dengan *behaviour* -nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

1. Class Diagram

*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungannya satu sama.

*Class* memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private* , tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected* , hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public* , dapat dipanggil oleh siapa saja

*Class* dapat merupakan implementasi dari sebuah *interface* , yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time* .

1. Statechart Diagram

*Statechart diagram* menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram* ). Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

1. A*ctivity Diagram*

*Activity diagrams* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di- *trigger* oleh selesainya *state* sebelumnya ( *internal processing* ). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state* , standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel ( *fork* dan *join* ) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

1. Sequence Diagram

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display* , dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men- *trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity* .

1. Collaboration Diagram

*Collaboration diagram* juga menggambarkan interaksi antar objek seperti *sequence diagram* , tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message* . Setiap *message* memiliki *sequence number* , di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

1. Component Diagram

*Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan ( *dependency* ) di antaranya. Komponen piranti lunak adalah modul berisi *code* , baik berisi *source code* maupun *binary code* , baik *library* maupun *executable* , baik yang muncul pada *compile time, link time* , maupun *run time* . Umumnya komponen terbentuk dari beberapa *class* dan/atau *package* , tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface* , yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

1. Deployment Diagram

*Deployment/physical diagram* menggambarkan detail bagaimana komponen di- *deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server, *workstation* , atau piranti keras lain yang digunakan untuk men- *deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

## Lokasi dan Waktu Penelitian

Lokasi penelitian laporan tugas akhir ini dilakukan di Universtas Nasional PASIM di Jalan Dr. Djundjunan No. 167. Yang dilaksanakan mulai bulan November 2017 sampai dengan Desember 2017. Sedangkan lokasi pembangunan aplikasi adalah di tempat penyusun sendiri. Adapun rincian kegiatan penyusunan tugas akhir yang direncanakan penulis adalah sebagai berikut :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kegiatan** | **November** | | | | **Desember** | | | | **Januari** | | | |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| 1. | Identifikasi Kebutuhan |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. | Membangun *Prototype* |  |  |  |  |  |  |  |  |  |  |  |  |
| 3. | Evaluasi *Prototype* |  |  |  |  |  |  |  |  |  |  |  |  |
| 4. | Pengkodean |  |  |  |  |  |  |  |  |  |  |  |  |
| 5. | Menguji Aplikasi |  |  |  |  |  |  |  |  |  |  |  |  |
| 6. | Evaluasi Aplikasi |  |  |  |  |  |  |  |  |  |  |  |  |
| 7. | Penggunaan Aplikasi |  |  |  |  |  |  |  |  |  |  |  |  |

**Tabel** **1.1.** Rincian Kegiatan Penyusunan Tugas Akhir