BAB I

PENDAHULUAN

A. Latar Belakang

Pada zaman yang semakin maju ini banyak sekali aplikasi dan sistem in*form*asi yang dibuat untuk menunjang dan menciptakan kemajuan di berbagai bidang dalam segala aktifitas yang dilakukan oleh manusia. Dengan hadirnya teknologi in*form*asi banyak membuat kemudahan dalam mendapatkan in*form*asi yang cepat, Teknologi in*form*asi bisa digunakan pada setiap bidang. Kemudahan dalam penggunaan teknologi in*form*asi ini diterapkan agar biasa di manfaatkan dengan baik oleh pengguna. Keberadaan teknologi in*form*asi sangat membantu, hampir di semua perusahaan atau istansi dari pemerintah maupun swasta di Indonesia sudah memiliki teknologi yang canggih dalam pengelolaan data tepat dan akurat.

Pemanfaatan teknologi in*form*asi yang telah banyak digunakan sebagai salah satu kinerja peningkatan mutu bagi penggunanya. Karena dengan adanya pemanfaatan teknologi in*form*asi khususnya sistem in*form*asi yang dapat membantu dan mempermudah semua kalangan masyarakat, perusahaan mendapatkan kemudahan dalam mengakses data menggunakan teknologi.

Pemanfaatan teknologi in*form*asi terdapat diberbagai bidang salah satunya perusahaan konstruksi, sistem in*form*asi ini dapat membantu perusahaan memudahkan dalam perencanaan anggaran biaya, dan juga dalam perhitungan biaya dapat berjalan dengan efektif, efisien dan akurat.

Untuk mendapatkan perencanaan anggaran biaya yang akurat harus didukung pula dengan perhitungan yang baik. Terkadang perhitungan yang rumit dan membutuhkan waktu yang lama membuat seseorang malas untuk menghitung, rencana anggaran biaya yang dilakukan secara manual juga dapat terjadi kesalahaan perhitungan ataupun penulisan yang membuat perhitungan menjadi tidak akurat. Oleh karna itu, dirasa perlunya sistem terutama pada sebuah perusahaan agar dapat memudahkan dalam perencanaan anggaran.

CV Patodongi merupakan perusahaan konstruksi yang didirikan pada 20 Mei 2019 di Parepare, Sulawesi Selatan, Indonesia, menawarkan berbagai layanan konstruksi. Dipimpin oleh Bapak Rahman sebagai direktur, perusahaan ini bergerak dalam sektor konstruksi, melibatkan pembangunan gedung, perbaikan dan renovasi bangunan, pembangunan infrastruktur, serta berbagai jasa konstruksi lainnya. Dengan jenis badan usaha CV (*Commanditaire Vennootschap*).

CV. Patodongi membuat perencanaan anggaran biaya masih menggunakan *Microsoft Excel*, sehingga mengalami beberapa kendala seperti : proses estimasi dan pengumpulan data untuk bahan yang diambil dari nota dan biaya yang dikeluarkan membutuhkan waktu yang cukup lama, perhitungan estimasti untuk analisa bahan sering terjadi kesalahan sehingga perencanaan anggaran biaya yang dibentuk tidak sesuai dengan harapan yang dapat menyebabkan terjadinya kerugian di pihak perusahaan, dan pembuatan laporan RAB yang memakan waktu cukup lama dan sering terjadinya kesalahan dalam perhitungan.

Oleh karena itu dibutuhkan aplikasi untuk menghitung secara cepat dan tepat. Perencanaan anggaran biaya suatu pekerjaan pembangunan hasil dari aplikasi dapat memudahkan perusahaan dalam pengelolaan data RAB. Penggunaan program aplikasi akan sangat membantu meningkatkan efektivitas dan efisiensi pelaksanaan kegiatan anggaran belanja yang akan menghasilkan in*form*asi yang cepat dan akurat serta relevan.

B. Rumusan Masalah

Dengan latar belakang yang telah dijelaskan, maka yang menjadi rumusan masalah pada penelitian ini adalah bagaimana sistem in*form*asi dapat membantu CV. Patodongi dalam perencanaan anggaran biaya untuk mengoptimalkan pengelolaan?

C. Batasan Masalah

Berdasarkan rumusan masalah di atas, maka batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sistem tidak membahas mengenai biaya pemesanan bahan baku bangunan
2. Sistem tidak membahas proses desain bangunan
3. Sistem tidak membahas faktor eksternal, meliputi cuaca dan bencana
4. Sistem tidak membahas masalah instalasi air (seperti kran), listrik, (kabel,stop kontak, dan saklar listrik), dan pemasangan kunci.
5. Aplikasi ini hanya di rancang untuk Rencana Anggaran Biaya (RAB).
6. Sistem tidak mencakup absensi dan pengujian karyawan.
7. Sempel harga dalam perencanaan ini hanya untuk harga Kota parepare.
8. Simulasi perhitungan digunakan data pada periode Agustus-Oktober 2024

D. Tujuan Penelitian

Tujuan dari penelitian ini adalah menghasilkan aplikasi berbasis *dekstop* perencanaan anggaran biaya yang dapat memberikan analisa perhitungan pembangunan proyek infastruktur dan menentukan biaya keseluruhan proyek dari sisi finansial.

E. Manfaat Penelitian

Adapun manfaat yang dapat diberikan dari penelitian ini adalah sebagai berikut:

1. Bagi akademik

   Sebagai bahan referensi bagi penulis lain untuk mengembangkan kemampuan di bidang yang sama.

2. Bagi penulis

   Penulis dapat mengimplementasikan ilmu yang didapat selama proses perkuliahan.

3. Bagi perusahaan

   Mengurangi resiko terjadinya kesalahan dalam pengolahan data dan membantu meningkatkan kinerja dan pembuatan laporan serta memudahkan karyawan dalam membuat perencanaan anggaran biaya.

BAB II

TINJAUAN PUSTAKA

A. Kajian Teori

1. Sistem In*form*asi

Secara umum pengertian sistem in*form*asi adalah suatu yang menyediakan in*form*asi untuk menejemen pengambilan keputusan atau kebijakan dan menjalankan operasional dari kombinasi orang-orang, teknologi, dan prosedur-prosedur yang terorganisasi.

Sistem in*form*asi juga dapat di artikan sebagai kombinasi dari teknologi untuk mendukung operasi dan menejemen. Sebuah sistem merupakan piranti penting dan sangat dibutuhkan oleh suatu perusahaan atau instansi resmi. Hal ini karena adanya sistem yang terintegrasi, kinerja suatu perusahaan atau istansi akan lebih terarah dan sistematis. Namun untuk mendapatkan dampak positif dari pengguna sistem, semua unsur-unsur yang terkait didalamnya harus bekerja sama guna mencapai tujuan yang sudah ditentukan sebelumnya.

Menurut John F. Nash pengertian sistem in*form*asi adalah kombinasi dari manusia, fasilitas ataupun teknologi, media, prosedur serta pengendalian yang memiliki maksud untuk menata jaringan komunikasi yang penting, proses maupun transaksi tertentu secara rutin, membantu manajemen serta pemakai intern maupun ekstern dan menyediakan dasar dari pengambilan keputusan yang tepat.

Menurut Robert A. Leitch sistem in*form*asi merupakan suatu sistem yang ada di dalam suatu organisasi yang dapat mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi yang bersifat manajerial serta kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan yang diperlukan.

Menurut Henri Lucas pengertian sistem in*form*asi adalah suatu kegiatan dari prosedur yang diorganisasikan dan jika dieksekusi maka akan menyediakan in*form*asi guna mendukung pengambilan keputusan serta pengendalian yang ada di dalam.

Menurut Alter, pengertian sistem in*form*asi adalah tipe khusus dari sistem kerja, di mana manusia maupun mesin melakukan suatu pekerjaan dengan menggunakan sumber daya guna memproduksi produk-produk tertentu dan atau jasa untuk pelanggan.

Sistem in*form*asi menurut Husein dan Wibowo ialah seperangkat komponen yang saling berhubungan serta memiliki fungsi untuk memproses, mengumpulkan, menyimpan serta menyebarkan in*form*asi guna mendukung pembuatan keputusan dan pengawasan dalam organisasi. Sistem in*form*asi terdiri dari in*form*asi yang membuat tentang manusia, tempat serta komponen yang ada di dalam organisasi maupun lingkungan yang melingkupinya.

Dari pengertian sistem in*form*asi di atas, maka dapat disimpulkan bahwa sistem in*form*asi memiliki hubungan dengan sistem data yang ada pada satu sisi serta sistem aktivitas pada sisi yang lainnya.

Sistem in*form*asi merupakan bentuk dari komunikasi sistem yang dapat mewakili data serta diproses sebagai bentuk dari memori sosial. Sistem in*form*asi dapat dianggap pula sebagai bahasa semi *form*al untuk mendukung manusia dalam mengambil keputusan atau tindakan.

Sistem in*form*asi menjadi fokus utama dari studi untuk disiplin sistem in*form*asi serta organisasi in*form*atika. Ada empat langkah untuk membangun suatu sistem in*form*asi, di antaranya adalah sebagai berikut:

a. Mengetahui permasalahan
b. Menyediakan solusi alternatif
c. Memiliki solusi
d. Menerapkan solusi yang telah dipilih

2. Rencana Anggaran Biaya (RAB)

Rencana anggaran biaya atau RAB merupakan perkiraan biaya yang biasanya digunakan untuk melaksanakan sebuah kegiatan bisnis atau proyek lainnya. Rencana anggaran biaya berfungsi sebagai panduan untuk mengelola sumber daya finansial yang terkait dengan proyek.

Rencana anggaran biaya juga memperhitungkan jangka waktu proyek atau kegiatan, serta alokasi biaya pada periode waktu yang relevan. Hal ini membantu dalam pengendalian dan pengawasan biaya selama pelaksanaan proyek.

Penyusunan Rencana Anggaran Biaya (RAB) ialah untuk menghitung biaya-biaya yang diperlukan dari suatu bangunan dan dengan biaya ini bangunan tersebut dapat terwujud sesuai dengan yang direncanakan.

Mengingat besar serta luasnya arti bangunan yang harus dihitung pembiayaannya maka pengetahuan akan hal-hal yang ada hubungannya denan perhitungan biaya sangat diperlukan. Ada persyaratan-persyaratan yang harus dipenuhi untuk menyusun rencana anggaran biaya yaitu gambar konstruksi bangunan, harga bahan bangunan dan upah tenaga kerja

Tujuan dari rencana anggaran biaya adalah:

a. Menyediakan perkiraan biaya yang realistis dan akurat yang dibutuhkan untuk mencapai tujuan proyek.
b. Membantu dalam perencanaan pengeluaran dan alokasi sumber daya finansial yang efektif.
c. Memberikan dasar untuk mengendalikan dan memantau pengeluaran selama proyek berlangsung.
d. Membantu dalam pengembalian keputusan terkait pengeluaran dan prioritas anggran.
e. Menyediakan in*form*asi yang diperlukan untuk proses perolehan dana atau pembiayaan proyek.

Rencana anggaran biasanya disusun oleh menejer proyek atau tim pengelola proyek dengan melibatkan ahli keuangan atau akuntan. Dokumen ini dapat mengalami perubahan seiring berjalan proyek, dan perubahan biaya dapat direvisi dan di catat dalam rencana anggaran biaya yang diperbaruhi.

3. *Database*

Database adalah sistem terorganisasi yang dirancang untuk menyimpan, mengelola, dan mengakses data secara efisien. Data dalam

database dapat berupa in*form*asi berupa angka, teks, gambar, atau *form*at lain yang terstruktur untuk memudahkan pengguna atau perangkat lunak dalam memproses in*form*asi tersebut. *Database* umumnya digunakan dalam berbagai aplikasi, seperti sistem keuangan, *e-commerce*, manajemen pelanggan, hingga analitik data besar. Untuk mengelola database, digunakan perangkat lunak yang disebut *Database Management System (DBMS*), seperti *MySQL, PostgreSQL, Oracle Database, dan MongoDB*. *DBMS* memungkinkan pengguna untuk menambahkan, menghapus, memperbarui, dan mengambil data menggunakan perintah khusus seperti *Structured Query Language (SQL*).

Database memiliki berbagai jenis berdasarkan struktur dan penggunaannya, termasuk database relasional yang menyimpan data dalam tabel-tabel dengan hubungan antar tabel melalui *primary key* dan *foreign key, database NoSQL* yang cocok untuk data tidak terstruktur seperti dokumen dan grafik, serta database berbasis grafik yang dirancang untuk hubungan kompleks seperti jaringan sosial. Selain itu, database dapat diimplementasikan secara lokal atau di *cloud*, memberikan fleksibilitas dalam penyimpanan dan akses data.

Database memainkan peran penting dalam sistem modern karena fungsinya yang mencakup penyimpanan data yang aman, pengelolaan data yang terstruktur, analitik data untuk pengambilan keputusan, dan pengolahan data secara *real-time*. Dalam database relasional, data diatur dalam tabel-tabel yang memiliki kolom (*fields*) dan baris (*records*). Kolom mewakili atribut

dari data, sementara baris berisi nilai data itu sendiri. Hubungan antar tabel dibuat menggunakan *primary key* yang berfungsi sebagai pengenal unik dan foreign key yang menghubungkan data antar tabel. Selain itu, database juga dilengkapi dengan mekanisme keamanan untuk melindungi data dari akses yang tidak sah, termasuk autentikasi, otorisasi, dan enkripsi.

Keuntungan dari penggunaan database meliputi efisiensi dalam pengelolaan data, minimalisasi redundansi melalui proses normalisasi, kemampuan akses multi-pengguna tanpa konflik, dan fitur pemulihan data untuk mengatasi kehilangan data akibat kesalahan atau kerusakan sistem. Namun, penggunaan database juga menghadapi tantangan, seperti kompleksitas pengelolaan data dalam skala besar, kebutuhan akan sumber daya perangkat keras dan perangkat lunak yang mahal, serta risiko terhadap keamanan jika tidak dilindungi dengan baik. Database juga berperan penting dalam berbagai industri. Dalam aplikasi *e-commerce*, misalnya, database digunakan untuk menyimpan data produk, pelanggan, dan transaksi. Dalam sektor perbankan, database digunakan untuk mencatat in*form*asi akun pelanggan dan melacak transaksi secara *real-time*. Layanan media sosial memanfaatkan database untuk mengelola data pengguna, aktivitas, dan interaksi antar pengguna. Sementara itu, dalam analitik data besar, database mendukung pengolahan dan visualisasi data untuk membantu pengambilan keputusan strategis.

Sebagai tulang punggung teknologi in*form*asi modern, database terus berkembang untuk memenuhi kebutuhan yang semakin kompleks, termasuk

dukungan terhadap big data, kecerdasan buatan, dan sistem berbasis cloud. Pemahaman yang mendalam tentang konsep, struktur, dan pengelolaan database menjadi keterampilan yang sangat penting dalam dunia yang semakin digital ini, karena hampir semua sistem teknologi bergantung pada database untuk memastikan kelancaran operasionalnya.

4. *SQLite 3*

SQLite adalah salah satu solusi penyimpanan data yang paling ringan dan mudah digunakan, tapi tetap andal. Berbeda dengan sistem basis data yang kompleks seperti MySQL atau PostgreSQL, *SQLite* langsung bekerja dari satu file kecil, tanpa perlu server. Pengertian dari *SQLite* adalah sistem manajemen basis data atau database yang sangat ringan dan tidak membutuhkan server untuk menjalankannya. Berbeda dengan beberapa database lain seperti MySQL atau PostgreSQL yang memerlukan instalasi dan pengaturan server, *SQLite* dapat berjalan langsung di dalam aplikasi yang menggunakannya. Data disimpan dalam satu file yang bisa diakses kapan saja tanpa pengaturan tambahan. Karena kepraktisannya, *SQLite* banyak digunakan dalam aplikasi mobile, perangkat IoT, dan program desktop.

Pada dasarnya, *SQLite* adalah sebuah software atau sistem manajemen relational database yang bersifat open-*Source*, serverless, dan portabel yang digunakan untuk mempermudah akses dan mengelola penyimpanan data. Software ini didesain untuk tujuan utama dalam hal kecepatan, kemudahan, portabel, dan dapat diandalkan ketika kita butuh untuk menyimpan data hanya 1 kilobytes ataupun ratusan gigabytes.

Penggunanya dapat dengan mudah menggunakan *SQLite* tanpa perlu melakukan instalasi *environment* atau konfigurasi apapun karena *SQLite* merupakan stand-alone software yang memang didesain untuk disematkan (embedded) ke dalam sebuah aplikasi. Selain itu, beberapa produk sistem manajemen relational database yang cukup populer lainnya adalah *MySQL*, Oracle Database, *PostgreSQL* memiliki ukuran yang jauh lebih besar dibandingkan *SQLite*.

Disamping itu *SQLite* juga tidak menggunakan model arsitektur client-server seperti *MySQL* dan *PostgreSQL* karena seluruh programnya berisi library Bahasa pemrograman C yang menyematkan database ke dalam sebuah aplikasi. Ini berarti Anda tidak membutuhkan konfigurasi server yang rumit untuk dapat menjalankan sistem manajemen database pada aplikasi.

5. *Borland Delphi*

Delphi adalah Kompiler/penterjemah bahasa *Delphi* (awalnya dari pascal) yang merupakan bahasa tingkat tinggi sekelas dengan Basic, C. Bahasa pemprograman di *Delphi* disebut bahasa procedural artinya/sintaknya mengikuti urutan tertentu/prosedur. Ada jenis pemprograman non-prosedural seperti pemprograman untuk kecerdasan buatan seperti bahasa prolog. *Delphi* termasuk keluarga *Visual* sekelas *Visual Basic, Visual C*, artinya perintah perintah untuk membuat objek dapat dilakukan secara visual. Pemprograman tinggal memilih objek apa yang ingin dimasukkan ke dalam *From/Window*, lalu tingkah laku objek tersebut saat menerima *event*/aksi tinggal dibuat programnya.

Borland Delphi adalah sebuah lingkungan pengembangan terintegrasi (*IDE*) yang dirancang untuk mempercepat proses pembuatan aplikasi desktop, web, dan *mobile*. Dikembangkan pertama kali oleh *Borland* pada tahun 1995, *Delphi* menggunakan bahasa pemrograman *Object* Pascal yang merupakan pengembangan dari bahasa *Pascal*. *Delphi* terkenal karena pendekatan *Rapid Application Development* (*RAD*)-nya, yang memungkinkan pengembang untuk merancang antarmuka pengguna secara visual menggunakan komponen yang telah tersedia, seperti tombol, kotak teks, dan daftar. Komponen-komponen ini terintegrasi dalam pustaka *Visual Component Library* (*VCL*), yang mempermudah pengembangan aplikasi berbasis *GUI* (*Graphical User Interface*).

Selain itu, *Delphi* mendukung konsep Pemrograman Berorientasi Objek (OOP), seperti enkapsulasi, pewarisan, dan polimorfisme, sehingga mempermudah pengelolaan kode dalam proyek-proyek besar. *Delphi* juga memiliki dukungan kuat untuk pemrograman database, memungkinkan pengembang untuk mengintegrasikan aplikasi dengan berbagai jenis database, seperti *MySQL, SQLite*, dan *Microsoft SQL Server*, menggunakan komponen seperti *FireDAC*. Meskipun awalnya populer untuk pengembangan aplikasi *Windows*, *Delphi* kini telah berkembang menjadi alat lintas plat*form* yang mendukung pengembangan untuk sistem operasi lain seperti *macOS*, *iOS*, *Android*, dan *Linux*. Dukungan yang luas, kecepatan pengembangan, dan kemudahan penggunaan membuat *Delphi* tetap relevan dalam dunia pemrograman hingga saat ini.

6. *Microsoft Excel*

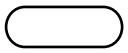
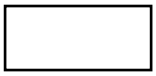
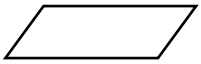
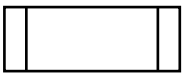
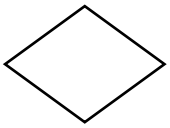
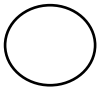
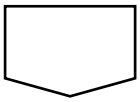
Menurut Syamsul Hadi dalam buku Mengenal *Microsoft Excel* untuk Pemula, berikut pengertian *Microsoft Excel*: "*Microsoft Excel* adalah aplikasi yang memungkinkan input data berupa tulisan, teks, angka, atau fungsi kompleks lainnya." Aplikasi ini juga dikenal dengan sebutan s*preadsheet* (lembar kerja elektronik). Program ini membantu pengguna untuk menghitung, menganalisis, juga mempresentasikan data.

Sementara itu, dikutip dari buku Praktis Belajar Komputer Seri *Microsoft Office* 2010 karya Endang Supriyadi, pengertian *Microsoft Excel* adalah: "*Microsoft Excel* merupakan program aplikasi milik *Microsoft* yang berbentuk spreadsheet. Sering digunakan untuk mengolah data yang kemudian dihimpun dalam tabel." Berdasarkan penjelasan di atas, kita bisa menarik kesimpulan bahwa *Microsoft Excel* adalah program aplikasi pengolah data yang dihimpun dalam bentuk tabel.

7. *Flowchart*

Menurut (Arifianto) "*Flowchart* adalah representasi grafis menggunakan simbol-simbol tertentu yang menunjukkan hubungan antara suatu instruksi dan operasi lain dalam suatu program serta urutan proses secara menyeluruh.

Tabel 2. 1 *Flowchart*

No	Simbol	Nama	Fungsi
1		*Terminator*	Permulaan/akhir program
2		Garis Alir (*Flow Line*)	Menyatakan bagaimana suatu prosedur berjalan
3		*Preparation*	Proses inisialisasi program/pemberian nilai awal
4		Proses	Proses perhitungan/proses pengolahan data
5		Input/Output Data	Proses input/output data, parameter, informasi
6		*Predefined process (sub program)*	Permulaan sub program/proses menjalankan sub program
7		*Decision*	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk data langkah selanjutnya
8		*On Page Connector*	Penghubung bagian-bagian *Flowchart* yang berada pada halaman berbeda
9		*Off Page Connector*	Penghubung bagian-bagian *Flowchart* yang pada halaman berbeda

8. *Unified modelling language*

UML adalah serangkaian alat yang digunakan untuk menyederhanakan suatu sistem atau perangkat lunak berbasis objek. Singkatan dari *Unified Modelling Language*, *UML* juga berperan dalam mempermudah pengembangan aplikasi secara berkelanjutan. Sistem atau aplikasi yang tidak didokumentasikan dengan baik seringkali menghambat proses pengembangan karena memaksa para pengembang untuk melakukan penyelidikan dan memahami kode program yang kompleks. *UML* juga

berfungsi sebagai alat untuk men-*transfer* pengetahuan tentang suatu sistem atau aplikasi dari satu pengembang ke pengembang lainnya. Keberadaan *UML* tidak hanya mempermudah komunikasi antara para pengembang, tetapi juga memungkinkan orang di luar lingkaran pengembangan, seperti pihak bisnis atau siapapun, untuk memahami suatu sistem dengan lebih baik.

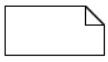
UML (*Unified Modelling Language*), sebagai standar industri, digunakan untuk menggambarkan, merencanakan, dan mendokumentasikan sistem perangkat lunak. *UML* menyediakan *form*at standar untuk merancang model sistem. Namun, karena *UML* memanfaatkan kelas dan operasi sebagai bagian dari konsep intinya, lebih sesuai digunakan untuk pengembangan perangkat lunak dengan bahasa berbasis objek seperti C++, Java, C#, atau VB.NET. Meskipun begitu, *UML* masih bisa digunakan untuk memodelkan aplikasi prosedural dalam bahasa seperti *VB* atau C.

UML mulai dikembangkan oleh *Object Management Group* sejak versi 1.0 dirilis pada bulan Januari 1997. Dalam pengembangan berbasis objek, terdapat beberapa prinsip kunci yang harus dipahami, seperti Objek, Kelas, Abstraksi, Enkapsulasi, Pewarisan, dan Polimorfisme. Adapun daftar simbol UML yaitu:

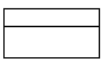
Tabel 2. 2 *Symbol Use Case Diagram*

No	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Mendefinisikan serangkaian tindakan pengguna saat berinteraksi dengan *use case.*
2		*Dependency*	Suatu hubungan di mana perubahan pada elemen tertentu mempengaruhi

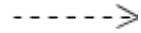
Lanjutan Tabel 2.6

No	GAMBAR	NAMA	KETERANGAN
3		*Generalization*	Hubungan dimana objek anak mewarisi perilaku dan struktur data dari objek yang berada di level hierarki di atasnya objek induk.
4		*Include*	Mengklarifikasi bahwa sumber *use case* secara jelas ditentukan.
5		*Extend*	Menjelaskan bahwa *use case* target menambahkan perilaku dari *use case* sumber pada titik tertentu.
6		*Association*	Yang menggabungkan dua objek menjadi satu.
7		*System*	Mengidentifikasi paket yang memiliki tampilan sistem terbatas.
8		*Use Case*	Penjelasan mengenai langkah-langkah yang diambil oleh sistem untuk menyebabkan seorang *actor* mengalami hasil yang dapat diukur
9		*Collaboration*	Cara hukum dan komponen lainnya berinteraksi untuk menghasilkan perilaku yang lebih kompleks daripada gabungan bagian-bagiannya (sinergi).
10		*Note*	Komponen fisik yang berfungsi sebagai sumber daya komputer dan hadir ketika suatu program dijalankan.

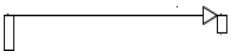
Tabel 2. 3 Symbol Class Diagram

NO.	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang

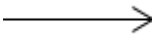
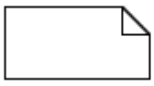
Lanjutan Tabel 2.3

NO.	GAMBAR	NAMA	KETERANGAN
			terukur bagi suatu *actor*
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

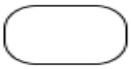
Tabel 2. 4 *Symbol Sequence Diagram*

No	GAMBAR	NAMA	KETERANGAN
1		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2		*Message*	Spesifikasi dari komunikasi antar objek yang memuat in*form*asi-in*form*asi tentang aktivitas yang terjadi
3		*Message*	Spesifikasi dari komunikasi antar objek yang memuat in*form*asi-in*form*asi tentang aktivitas yang terjadi

Tabel 2. 5 *Symbol State Chart Diagram*

No	GAMBAR	NAMA	KETERANGAN
1		*State*	Nilai atribut dan nilai *Link* pada suatu waktu tertentu, yang dimiliki oleh suatu objek.
2		*Initial Pseudo State*	Bagaimana objek dibentuk atau diawali
3		*Final State*	Bagaimana objek dibentuk dan dihancurkan
4		*Transition*	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya
5		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
6		*Node*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2. 6 *Symbol Activity Diagram*

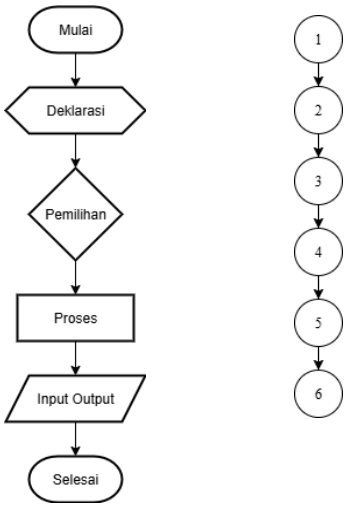
No.	GAMBAR	NAMA	KETERANGAN
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Activity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

9. *White Box Testing*

White box Testing disebut sebagai pemeriksaan struktur. Penguji dapat melihat dengan tepat program mana yang sedang diuji. Karena struktur internal kode diketahui dengan menguji setiap bagian yang dapat dievaluasi untuk mengidentifikasi masalah logis dalam kode sumber perangkat lunak, pengujian dibuat dari sudut pandang pengembang. Jovanovic dalam jurnal (Praniffa et al., 2023).

White box testing memiliki belebihan, seperti mampu menghapus bagian asing dari kode –kode yang tersembunyi, melakukan pengujian secara menyeluruh karena seluruh bagian dari struktur atau logika dieksplorasi, membantu dalam mengoptimalkan kode, dan pengujian *white box testing* dapat dimulai walaupun GUI masih dalam masa pengembangan(Praniffa et al., 2023).

Adapun contoh dari pengujian *white box* dapat kita lihat di bawah



a. Pertama membuat flowchart dan flowgram activity user

b. Kemudian menghitung *Cyclomatic Complexcity* V(G) dari *Edge* dan Node dengan rumus

c. V(G) = E – N + 2

d. Ket :

E = Edge

N = Node

e. Menentukan region berdasarkan *Cyclomatic Complexcity* V(G)

f. Membuat Independent Path Berdasarkan flowgraph

g. Terakhir membuat matriks aktivasi user berdasarkan Independent Path

Tabel 2. 7 Contoh Tabel Matriks

	1	**2**	**3**	**4**	**5**	**6**	**E-1**
1		1					1 - 1 = 0
2			1				1 – 1 = 0
3				1			1 – 1 = 0
4					1		1 – 1 = 0
5						1	1 – 1 = 0
6							0
	SUM (E + 1)						1 + 1 = 2

10. *Black box Testing*

Metode *Black box Testing* merupakan salah satu metode yang mudah digunakan karena hanya memerlukan batas bawah dan batas atas dari data yang di harapkan. Estimasi banyaknya data uji dapat dihitung melalui banyaknya field data entri yang akan diuji, aturan entri yang harus dipenuhi serta kasus batas atas dan batas bawah yang memenuhi. Dan dengan metode ini dapat diketahui jika fungsionalitas masih dapat menerima masukan data yang tidak diharapkan maka menyebabkan data yang disimpan kurang valid.

Pengujian adalah satu set aktifitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan. Pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian pada sistem menggunakann metode *Black Box*, tujuannya mengetahui kelemahan dari sistem agar data yang dihasilkan sesuai dengan data yang dimasukkan setelah data dieksekusi dan menghindari kekurangan dan kesalahan pada aplikasi sebelum digunakan oleh user

Tujuan *Black box* Testing adalah untuk mengevaluasi setiap fitur infrastruktur program aplikasi. *Black box Testing* bertujuan untuk menentukan apakah perangkat lunak beroperasi sebagaimana mestinya tanpa memiliki akses ke kode sumber yang sebenarnya (Ismail & Efendi, 2020). Contoh table *blackbox* sebagai berikut:

Tabel 2. 8 Contoh Tabel Black Box Test

Test Faktor	**Hasil**	**Kesimpulan**
Halaman yang diuji	Berhasil/gagal	Penjelesan jika berhasil atau tidak
Screenshoot		
***Source* Gambar atau Screenshoot Halaman yang diuji**		

B. Kajian Hasil Penelitian Terdahulu

Untuk dapat digunakan sebagai data pendukung, diperlukan berbagai penelitian sebelumnya. Studi sebelumnya yang berkaitan dengan masalah yang dibahas dalam penelitian ini merupakan salah satu in*form*asi pendukung yang peneliti butuhkan. Dalam hal ini, referensi yang berkaitan dengan tantangan teknologi in*form*asi dari studi sebelumnya digunakan. Oleh karena itu penelitian dilakukan secara *online* dengan menggunakan berbagai temuan penelitian yang disajikan dalam makalah akhir, tesis, atau publikasi.

(Suprianti, Tegar bayu Ginanjar, 2021) dengan judul "Analisa dan perancangan Sistem In*form*asi Akuntansi Go! Adventure Yogyakarta". Penelitian ini menghasilkan sistem in*form*asi akuntansi berbasis dekstop sebagai solusi untuk mempermudah admin dan kasir dalam mengelola dan memproes data peminjaman dan data pengembalian serta membantu mengurangi beban waktu yang lama pada admin dan kasir dalam pengolahan data dan proses pencarian data peminjaman.

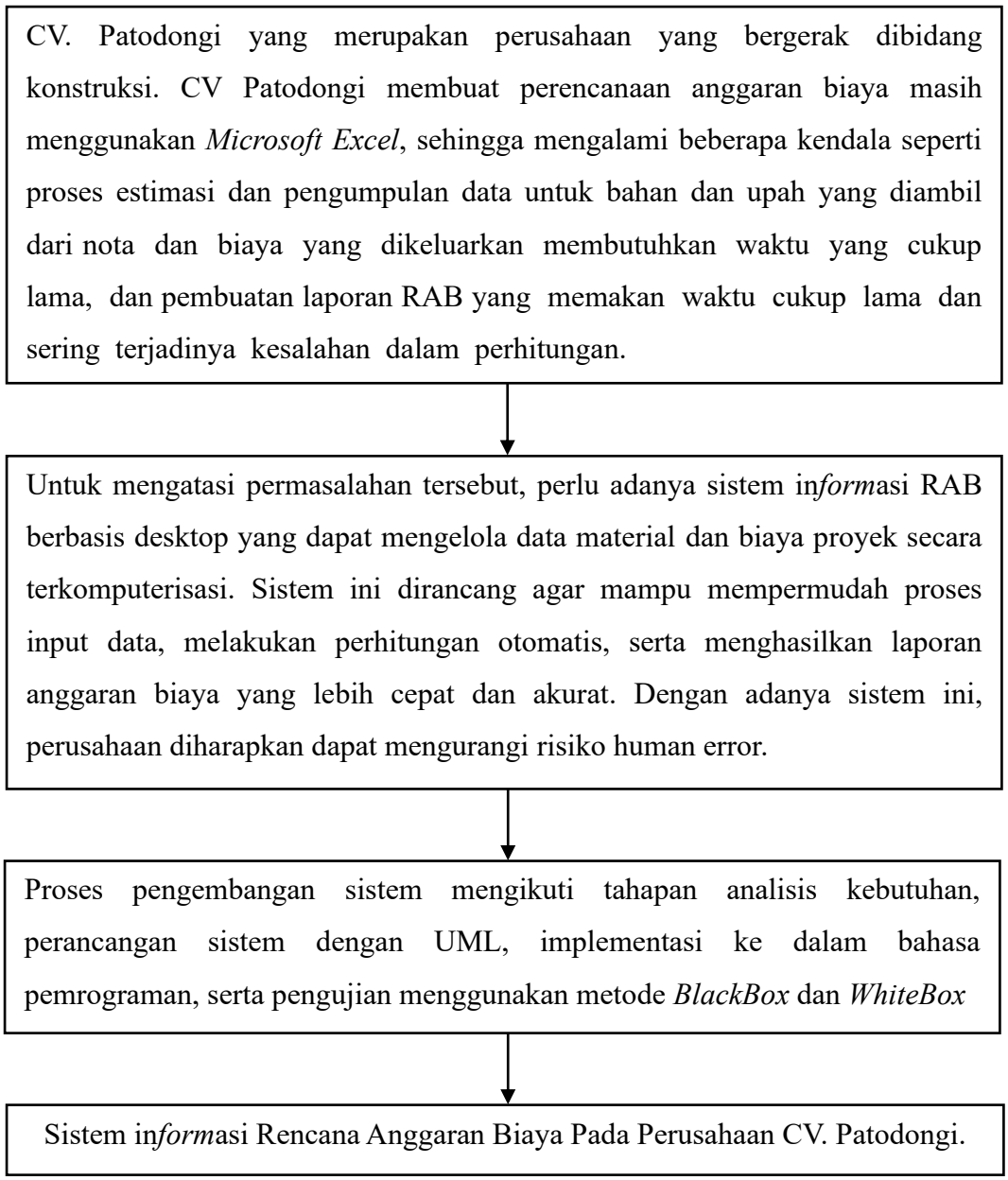
(Yulia Meliana, 2023) dengan judul "Analisa Peranan Dan Kegunaan Sistem In*form*asi Akuntansi Pada PT Kurnia Eka Nusa Yang Berbasis Accurate-Desktop". Penelitian ini bertujuan untuk mengetahui peranan dan kegunaan

Sistem In*form*asi Akuntansi yang digunakan dalam pengelolaan data keuangan agar lebih efektif di PT Kurnia Eka Nusa.

(Resti Lia Andharsaputri, 2021) dengan judul "Rancang Bangun Sistem In*form*asi Pengadaan Barang dan Jasa Berbasis Dekstop".Tujuan dari penelitian ini adalah membantu perusahaan dalam mengatasi permasalahan diatas agar pelaksanaan operasional perusahaan dapat berjalan lebih baik, lebih efektif dan efisien, mengurangi terjadinya human error, serta mengurangi beban kerja karyawan yaitu dengan sebuah sistem in*form*asi yang terkomputerisasi dan program aplikasi sebagai implementasinya.

(Ni komang Ayu Suarpurningsih, A.A Istri Ita Paramitha, 2021) dengan judul "Perancangan Sistem In*form*asi pengolaan Rencana anggaaran biaya (RAB) pada peusahaan konstruksi Bebasis Web (Studi Kasus Cv. Osa Karya Mandiri)". Penelitian ini bertujuan untuk merancang system pengelolaan rencana anggaran biaya berbasis web mengginakan metode System Development Life Cycle (SDLC). Hasil dari penelitian ini adalah sistem in*form*asi pengelolaan rencana anggaran biaya (rab) pada perusahaan konstruksi berbasis web (studi kasus cv. osa karya mandiri).

C. Kerangka Pikir



BAB III

METODE PENELITIAN

A. Jenis Penelitian

Metode penelitian yang digunakan adalah Research and Development (R&D) dengan pendekatan kuantitatif, karena sesuai dengan kebutuhan CV. Patodongi yang masih menghadapi kendala dalam penyusunan Rencana Anggaran Biaya (RAB) menggunakan Microsoft Excel. Melalui metode R&D, penelitian ini tidak hanya menganalisis permasalahan, tetapi juga menghasilkan produk berupa aplikasi berbasis desktop yang dapat membantu perusahaan dalam mengelola RAB secara lebih efisien dan akurat. Sementara itu, pendekatan kuantitatif digunakan untuk mengukur efektivitas sistem melalui penyebaran kuesioner kepada pengguna, sehingga diperoleh data terukur mengenai tingkat kepuasan, kecepatan, dan ketepatan aplikasi dalam mendukung kinerja perusahaan

B. Lokasi dan Waktu Penelitian

Tempat penelitian ini di perusahaan CV.Patodongi. Penelitian ini berlangsung selama ± 2 (dua) bulan di tahun 2025

C. Alat dan Bahan Penelitian

1. Perangkat keras

Bahan dan instrumen penelitian diperlukan untuk membantu kegiatan penelitian pada saat melakukan penelitian. Perlengkapan dan perlengkapan berikut ini diperlukan:

1. Alat

   a. Laptop acer aspire A314-23M dengan spesifikasi :

      1) *Processor* : AMD Ryzen 3 7320U with Raden Graphics

      2) *RAM* : 8GB

      3) *Harddisk* : 475GB

   b. Dalam membuat aplikasi, *software* yang digunakan adalah :

      1) *Windows 11 Home Single Language*

      2) *Delphi XE8*

      3) *AQLite 3*

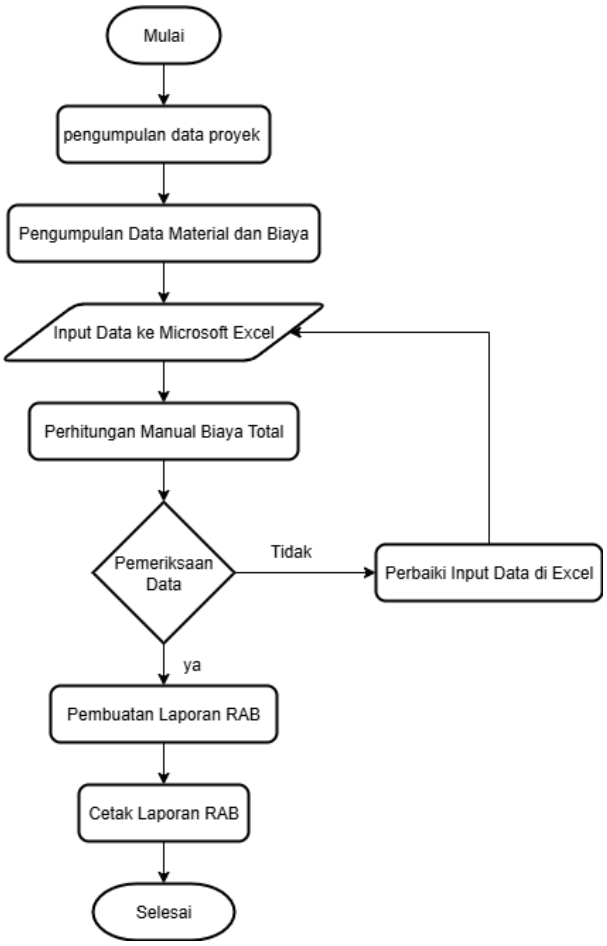
      4) *Microsoft excel*

2. Bahan

   Bahan penelitian yang digunakan berupa data proyek dari perusahaan CV. Patodongi, meliputi data harga material bangunan, data upah tenaga kerja serta perencanaan proyek yang relevan.

D. Rancangan Sistem

1. Sistem yang berjalan



Gambar 3. 1 *Flowchart* sistem yang berjalan

Flowchart sistem yang berjalan di CV Patodongi menggambarkan proses manual yang dilakukan untuk membuat Rencana Anggaran Biaya (RAB) menggunakan *Microsoft Excel*.

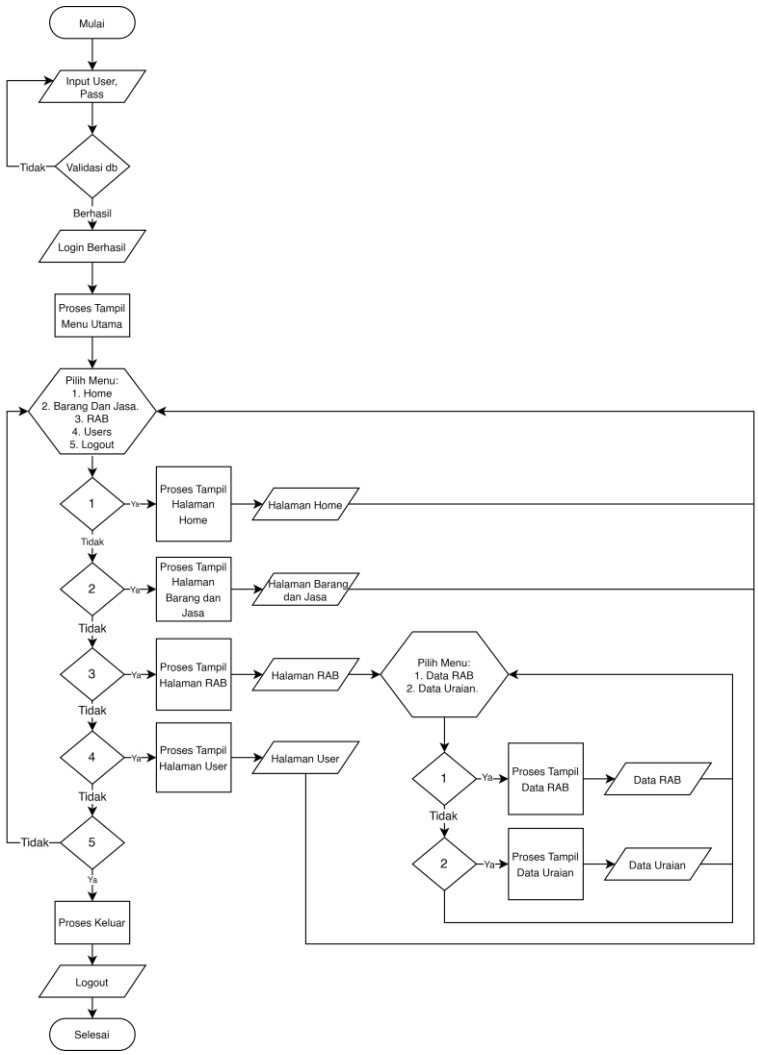
Proses dimulai dengan pengumpulan data proyek, seperti nama proyek,lokasi, tanggal mulai, dan in*form*asi lainnya yang relevan. Setelah itu, data material dan biaya, termasuk biaya tenaga kerja dan bahan bangunan, dikumpulkan secara manual dari nota atau sumber lainnya. Data yang telah

terkumpul kemudian dimasukkan ke dalam *Microsoft Excel* untuk dilakukan penghitungan.

Setelah data dimasukkan, proses penghitungan total biaya dilakukan secara manual menggunakan *form*ula di Excel. Kemudian, data yang telah dihitung diperiksa secara manual untuk memastikan tidak ada kesalahan. Jika terdapat kesalahan, pengguna harus kembali ke tahap input data dan memperbaiki kesalahan tersebut di Excel. Namun, jika data dinyatakan benar, proses dilanjutkan ke tahap pembuatan laporan RAB.

Laporan RAB yang telah selesai dibuat kemudian disimpan di *Microsoft Excel* dan dicetak untuk kebutuhan dokumentasi atau diberikan kepada klien. Proses ini mengakhiri alur sistem manual yang berjalan saat ini di CV Patodongi. Namun, sistem ini memiliki kelemahan, seperti memakan waktu yang lama, rentan terhadap kesalahan dalam perhitungan dan input data, serta kurang efisien dibandingkan dengan sistem yang terkomputerisasi secara khusus.

2. Sistem yang diusulkan



Gambar 3. 2 *Flowchart* yang diusulkan

a. Deskripsi flowchart

1) **Mulai**: Pengguna memulai dengan menginputkan ***username*** dan ***Password***.

2) **Validasi**: Setelah input dilakukan, sistem akan memvalidasi data ***username*** dan ***Password***.

a) Jika validasi berhasil, sistem akan mengarahkan pengguna ke halaman utama.

b) Jika gagal, pengguna akan diminta untuk mencoba lagi.

3) **Menu Utama**: Setelah berhasil *login*, pengguna akan disajikan dengan menu utama yang berisi pilihan:
   a) ***Home***
   b) **Barang dan Jasa**
   c) **RAB**
   d) ***Users***
   e) ***Logout***
4) **Pemilihan Menu**:
   a) Jika pengguna memilih ***Home***, sistem akan menampilkan halaman ***Home***.
   b) Jika memilih **Barang dan Jasa**, sistem akan menampilkan halaman **Barang dan Jasa**.
   c) Jika memilih **RAB**, sistem akan menampilkan halaman **RAB**.
   d) Jika memilih ***Users***, sistem akan menampilkan halaman ***Users***.
   e) Jika memilih ***Logout***, sistem akan keluar dan menyelesaikan sesi.
5) **Navigasi Lebih Lanjut**:
   a) Jika memilih menu **RAB** atau ***Users***, ada pilihan untuk melanjutkan dengan memilih **Data RAB** atau **Data Uraian**, masing-masing akan menampilkan data sesuai dengan pilihan.

E. Teknik Pengumpulan Data

1. Metode wawancara

Metode wawancara, yaitu metode yang dilakukan untuk mengumpulkan data dengan melakukan wawancara langsung dengan pengelola rencana anggaran biaya pada perusahaan Cv.Patodongi.

2. Observasi

Observasi merupakan teknik pengumpulan data dengan cara mengamati secara langsung maupun tidak langsung. Observais tidak langsung dilakukan dengan cara mengamati objek penelitian melalui media tertentu, seperti video atau foto.

3. Dokumentasi

Pada tahap ini dilakukan mengumpulkan data yang berkaitan dalam bentuk dokumen. Dokumen-dokumen tersebut dapat berupa dokumen tertulis, dokumen elektronik, atau dokumen audiovisual.

BAB IV

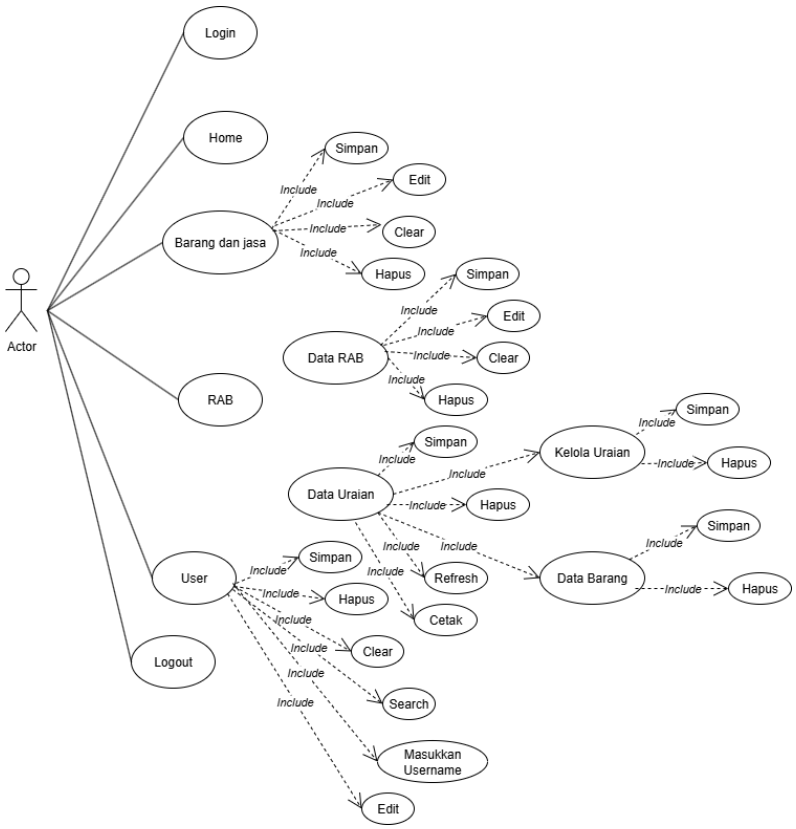
HASIL DAN PEMBAHASAN

A. Analisis Aliran Data Dengan UML

Proses pengembangan selanjutnya adalah proses desain pengembangan perangkat lunak. Desain sistem perangkat lunak yang dibuat nantinya akan dijadikan sebagai acuan pengembang dalam penulisan kode. Desain sistem perangkat lunak ini harus sesuai karakteristik perangkat lunak yang akan dibuat serta mampu mempermudah dan memperjelas pengembang dalam proses pembuatan perangkat lunak. Penelitian ini menggunakan desain sistem model *Unified Model Language* (UML) dikarenakan model ini paling sesuai digunakan Untuk mengembangkan sistem berionterasi.

1. *Use case diagram*

Use Case menggambarkan bagaimana seorang pengguna berinteraksi dengan sistem dengan cara menentukan langkahlangkah yang diperlukan untuk mencapai suatu tujuan tertentu. Variasi-variasi dalam urutan langkah yang terjadi antara aktor dan sistem merupakan sebuah skenario. Diagram *Use Case* memberikan suatu gambaran besar tentang fungsionalitas yang diharapkan dari sebuah sistem dilihat dari sudut pandang pengguna.



Gambar 4. 1 *Use Case Diagram User* Sistem yang diusulkan

Tabel 4. 1 Deskripsi *Use Case*

Use Case	Deskripsi
Login	Proses masuk ke dalam sistem dengan *username* dan *Password*.
Home	Halaman utama setelah berhasil *login*, sebagai pusat navigasi.
Barang dan Jasa	Menu untuk mengelola data barang dan jasa.
Simpan	Menyimpan data barang/jasa baru.
Edit	Mengubah data barang/jasa yang ada.
Hapus	Menghapus data barang/jasa yang dipilih.
Clear	Mengosongkan *form* input barang/jasa.
RAB	Menu untuk mengelola data Rencana Anggaran Biaya (RAB).
DATA RAB	Submenu untuk menampilkan data RAB.
Simpan	Menyimpan data RAB baru.

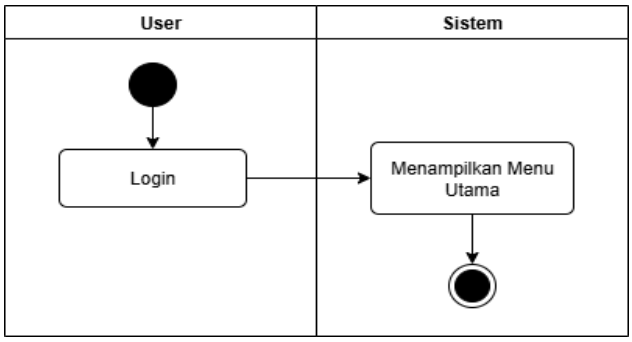
Lanjutan Tabel 4.1

Use Case	Deskripsi
Edit	Mengubah data RAB.
Hapus	Menghapus data RAB yang dipilih.
Clear	Mengosongkan *form* input RAB.
DATA URAIAN	Menu untuk mengelola uraian dalam RAB.
Simpan	Menyimpan data uraian baru.
Hapus	Menghapus data uraian.
Refresh	Memperbarui tampilan data uraian.
Cetak	Mencetak laporan uraian.
Kelola Uraian	Mengelola detail uraian dalam RAB.
Kelola Barang	Mengelola barang yang terkait dengan uraian.
Users	Menu untuk mengelola data pengguna sistem.
Simpan	Menambahkan user baru.
Edit	Mengubah data user.
Hapus	Menghapus user.
Clear	Mengosongkan *form* input user.
Search	Mencari data user.
Masukkan *Username*	Input field untuk memasukkan *username* saat mencari/menambah data user.
Logout	Keluar dari sistem.

2. *Activity diagram*

Activity Diagram ini menjelaskan tentang aktivitas-aktivitas yang terjadi dalam sebuah aliran proses pada Penelitian ini.

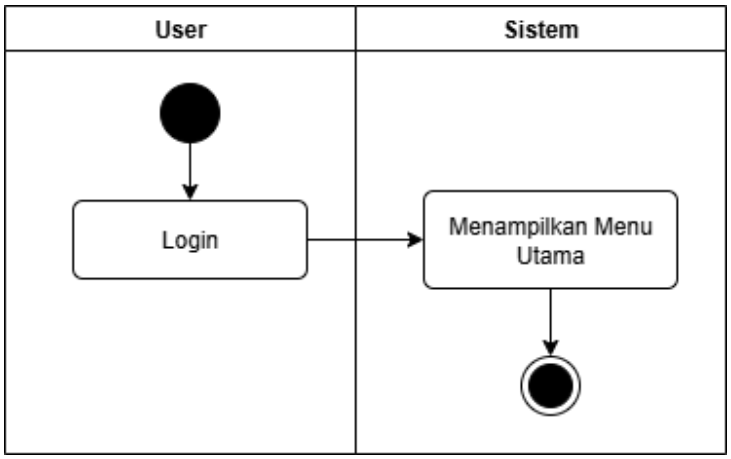
a. Activity diagram Login



***Gambar 4. 2** Activity Diagram Login*

Gambar diatas menjelaskan alur aplikasi pada menu *Login* untuk mengakses halaman utama.

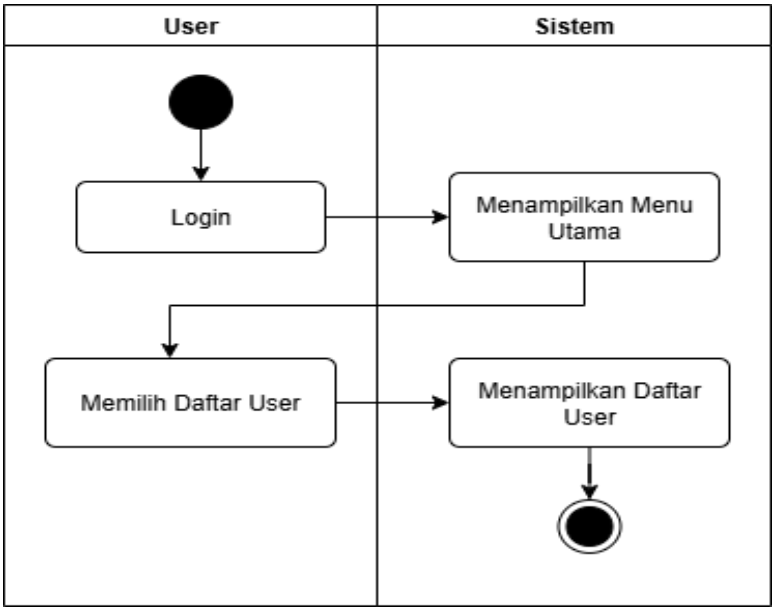
b. Activity diagram Menu Utama



Gambar 4. 3 *Activity* Diagram Tampilan Menu Simulasi

Gambar diatas menjelaskan alur aplikasi pada menu utama, Setelah *Login* akan menampilkan halaman utama.

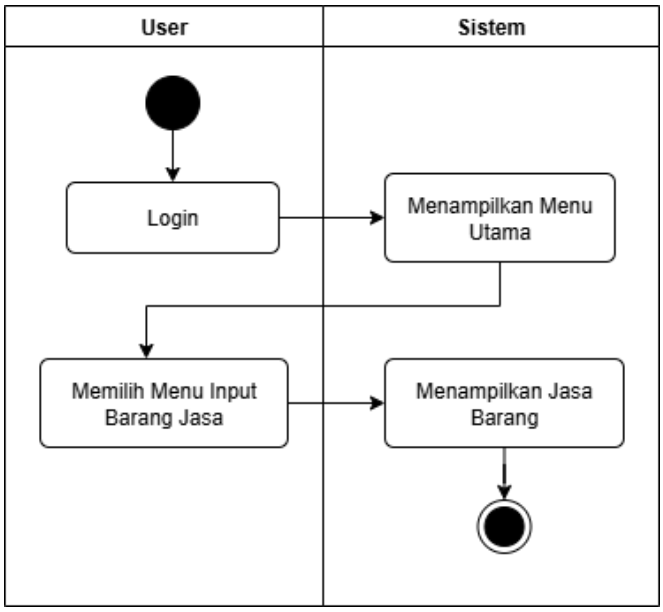
c. Activity diagram Menu Daftar User



Gambar 4. 4 *Activity* Diagram Menu Daftar *User*

Gambar diatas menjelaskan alur aplikasi pada menu Pendaftaran User, didalam menu utama terdapat menu pendaftaran User yang dapat di akses sebagai syarat untuk *login* pada sistem.

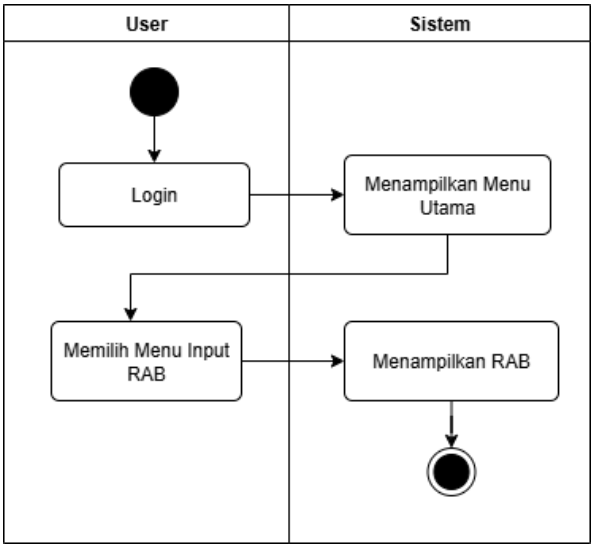
d. *Activity diagram* Menu Barang dan Jasa



Gambar 4. 5 *Activity* Diagram Menu Barang dan Jasa

Gambar diatas menjelaskan alur aplikasi pada menu Barang dan Jasa, didalam menu ini diarahkan untuk mengisi *form* untuk menghasilkan penginputan Barang dan Jasa.

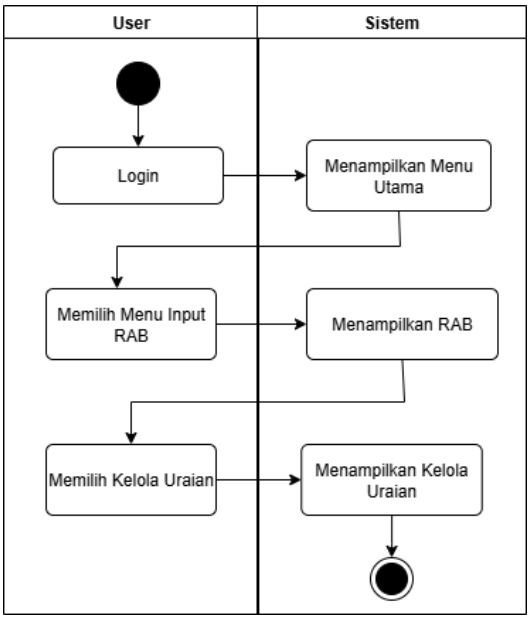
e. *Activity diagram* Menu Rencana Anggaran Biaya



Gambar 4. 6 *Activity* Diagram Menu Rencana Anggaran Biaya

Gambar diatas menjelaskan alur aplikasi pada menu Rencana Anggaran Biaya (RAB), didalam menu ini diarahkan untuk mengisi *form* untuk melanjutkan ke tahap selanjutnya yaitu Menu kelola Uraian dan Jasa.

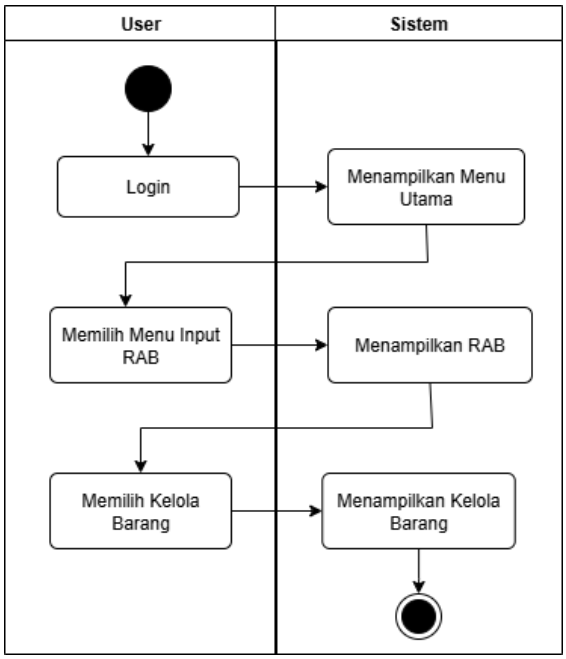
f. Activity diagram Menu Kelola Uraian



Gambar 4. 7 *Activity* Diagram Menu Kelola Uraian

Gambar diatas menjelaskan alur aplikasi pada menu kelola Uraian, setelah melakukan penginputan pada menu RAB akan di arahkan untuk mengisi penginputan pada menu Uraian.

g. Activity diagram Menu Kelola Barang



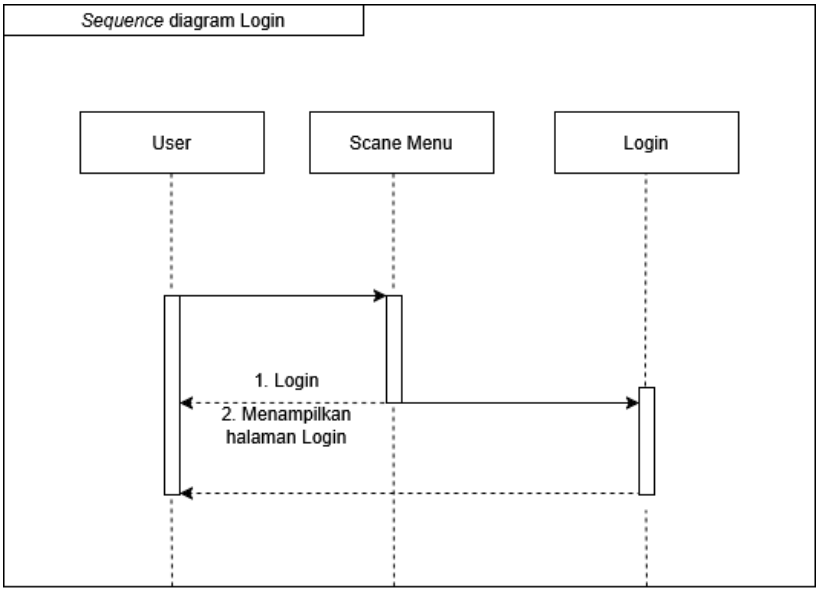
Gambar 4. 8 *Activity* Diagram Menu Kelola Barang

Gambar diatas menjelaskan alur aplikasi pada menu kelola Barang, setelah melakukan penginputan pada menu RAB akan di arahkan untuk mengisi penginputan pada menu Barang.

3. *Sequence Diagram*

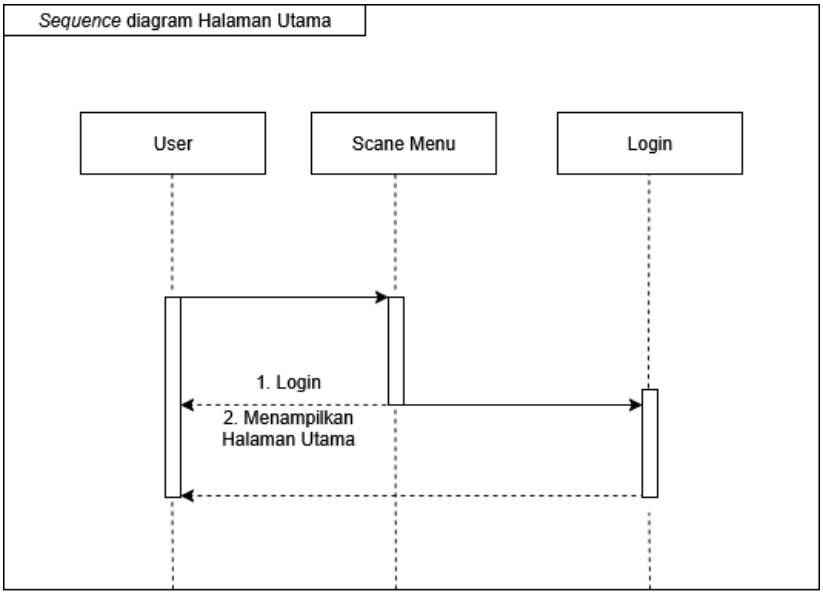
Sequence Diagram merupakan salah satu diagram *Interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan *message* (pesan) apa yang dikirim dan kapan pelaksanaannya.

A. *Sequence* diagram menu *Login*



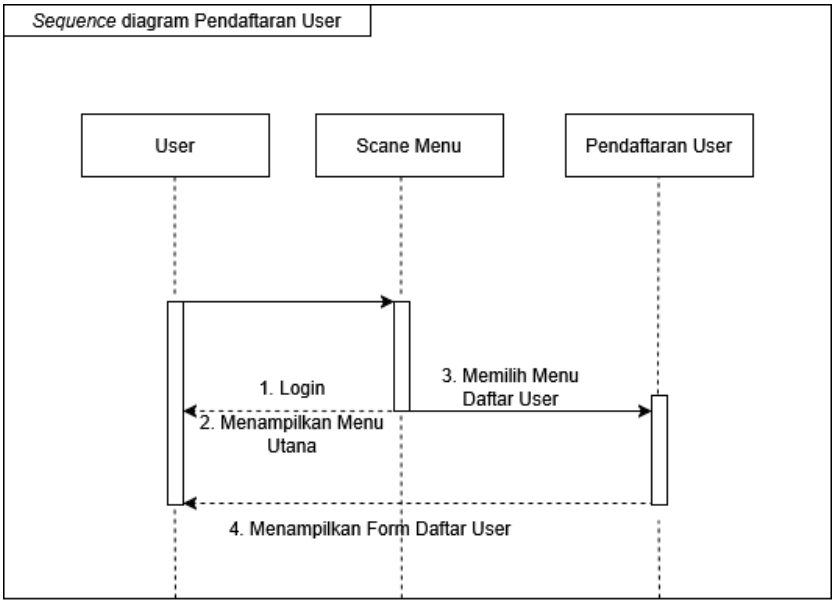
Gambar 4. 9 *Sequence* diagram menu *Login*

B. Sequence diagram menu Utama



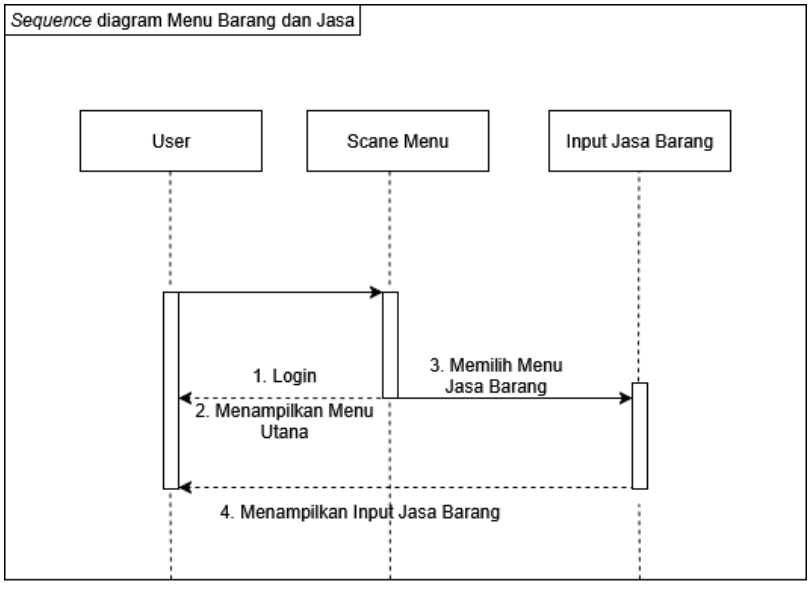
Gambar 4. 10 *Sequence* diagram Menu Utama

b. *Sequence* diagram menu daftar *User*



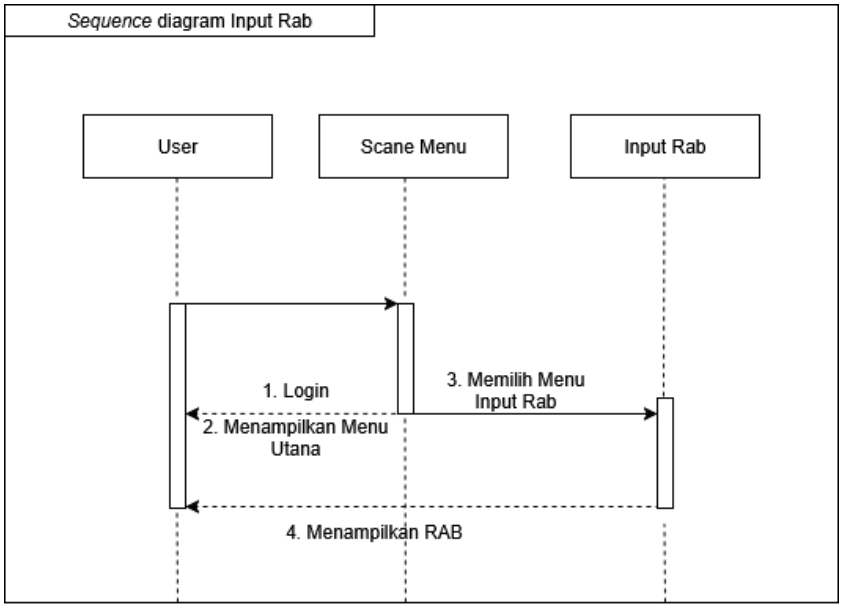
Gambar 4. 11 *Sequence* diagram Daftar *User*

c. *Sequence* diagram menu Barang dan Jasa



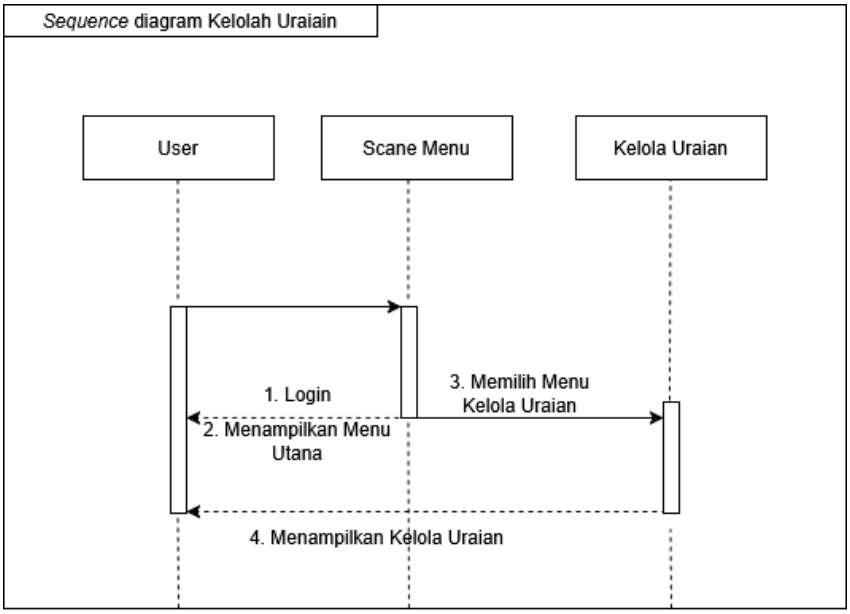
Gambar 4. 12 *Sequence* diagram Menu Barang dan Jasa

d. *Sequence* diagram menu Rencana Anggaran Biaya (RAB)



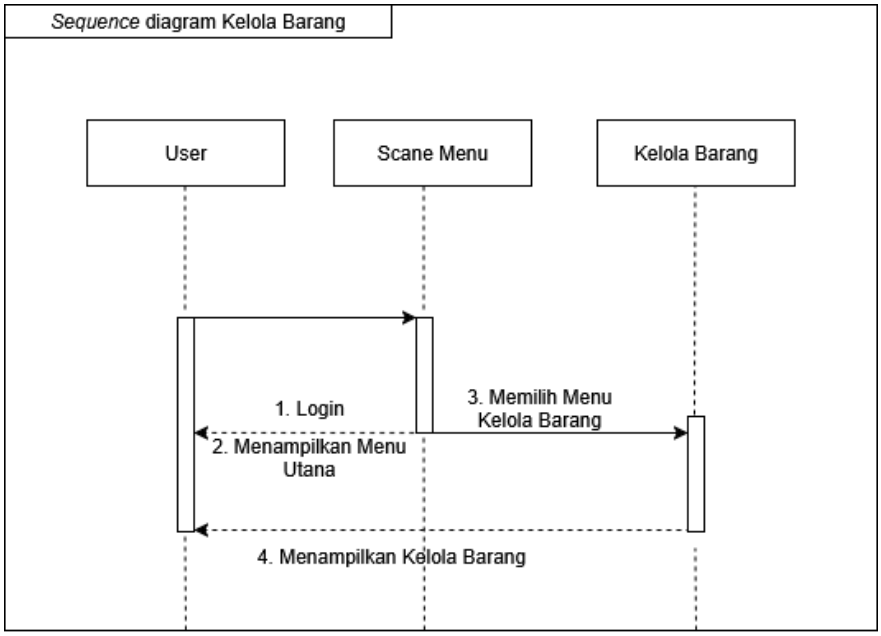
Gambar 4. 13 *Sequence* diagram Menu Rencana Anggaran Biaya (RAB)

e. *Sequence* diagram menu kelola Uraian



Gambar 4. 14 *Sequence* diagram Menu Kelola Uraian

f. *Sequence* diagram menu kelola Barang



Gambar 4. 15 *Sequence* diagram Menu kelola Barang

B. Perancangan Database

1. Tabel *user*

Tabel 4. 2 Tabel *User*

No	Nama	Jenis	Ukuran	Ket
1	id	Int		*PRIMARY KEY*
2	fullname	Text	100	
3	*Username*	Text	64	
4	*Password*	Text	64	

2. Tabel RAB

Tabel 4. 3 Tabel RAB

No	Nama	Jenis	Ukuran	Ket
1	id_rab	Int		*PRIMARY KEY*
2	Nama_rab	Text	100	

Lanjutan Tabel 4.2

No	Nama	Jenis	Ukuran	Ket
3	Kode	Int	11	
4	Anggaran	Int	50	

3. Tabel uraian

Tabel 4. 4 Uraian

No	Nama	Jenis	Ukuran	Ket
1	id_uraian	Int		*PRIMARY KEY*
2	id_rab	Int		*FOREIGN KEY*
3	Uraian	text	200	

4. Tabel barang

Tabel 4. 5 Tabel Barang

No	Nama	Jenis	Ukuran	Ket
1	id_barang	Int		*PRIMARY KEY*
2	Nama_barang	Text	100	
3	Satuan	Text	100	
4	Harga	Int	11	
5	Pajak	Int	2	

5. Tabel RAB detail

Tabel 4. 6 Tabel RAB Detail

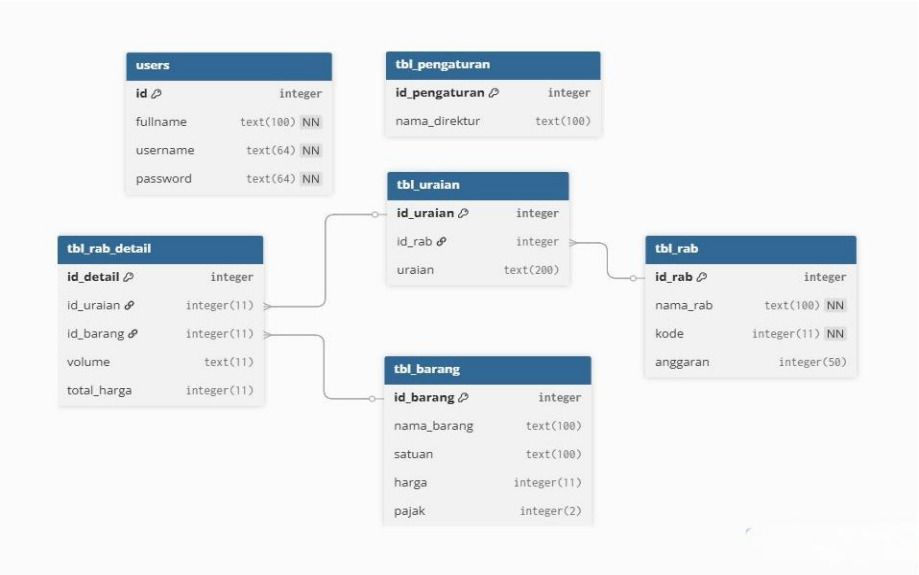
No	Nama	Jenis	Ukuran	Ket
1	id_detail	Int		*PRIMARY KEY*
2	id_uraian	Int	11	
3	id_barang	Int	11	
4	Volume	Text	11	
5	Total_harga	Int	11	

6. Tabel pengaturan

Tabel 4. 7 Tabel Pengaturan

No	Nama	Jenis	Ukuran	Ket
1	id_pengaturan	Int		*PRIMARY KEY*
2	Nama_direktur	Text	100	

7. Relasi Tabel Database



Gambar 4. 16 Relasi Database

Database pada ERD pada gambar 4.16 dirancang untuk mengelola sistem Rencana Anggaran Biaya (RAB), di mana setiap komponen dalam anggaran dikelola secara terstruktur. Terdapat lima tabel utama dalam sistem ini, yaitu *Users*, tbl_rab, tbl_uraian, tbl_barang, dan tbl_rab_detail, yang saling berhubungan untuk mendukung pengelolaan data RAB.

Tabel *Users* berfungsi untuk menyimpan in*form*asi pengguna yang memiliki akses ke dalam sistem. Setiap pengguna memiliki atribut seperti id sebagai kunci utama, serta fullname, *Username*, dan *Password* untuk

mengamankan akses mereka. Dengan adanya tabel ini, sistem dapat membedakan antara pengguna yang memiliki peran dalam mengelola dan mengakses data RAB.

Tabel pengaturan berfungsi untuk menyimpan data konfigurasi sistem yang bersifat fleksibel dan dapat diubah sesuai kebutuhan operasional perusahaan. Di dalam tabel ini tersimpan nama direktur perusahaan. Dengan adanya tabel pengaturan, pengguna dapat melakukan penyesuaian nama direktur perusahaan tanpa perlu mengubah struktur database atau kode program, sehingga sistem menjadi lebih dinamis dan mudah disesuaikan jika terjadi perubahan nama direktur perusahaan.

Selanjutnya, tabel tbl_rab menyimpan in*form*asi utama mengenai setiap RAB yang dibuat. Setiap RAB memiliki id_rab sebagai kunci utama, nama_rab untuk memberikan identitas pada RAB, serta kode sebagai nomor referensi atau identifikasi RAB, dan nilai anggaran proyek yang di jadikan dasar sebagai pembuatan RAB. Tabel ini menjadi pusat utama dalam sistem, karena semua in*form*asi terkait RAB akan dikaitkan dengannya.

Untuk memberikan rincian lebih lanjut mengenai suatu RAB, digunakan tabel tbl_uraian yang menyimpan deskripsi atau uraian dari masing-masing bagian dalam sebuah RAB. Setiap uraian memiliki id_uraian sebagai kunci utama dan terhubung ke id_rab sebagai referensi ke tabel tbl_rab. Dengan adanya tabel ini, setiap bagian dalam anggaran dapat memiliki deskripsi yang jelas dan terorganisir.

Dalam pengelolaan barang atau material yang digunakan dalam anggaran, sistem memiliki tabel tbl_barang. Tabel ini menyimpan in*form*asi mengenai

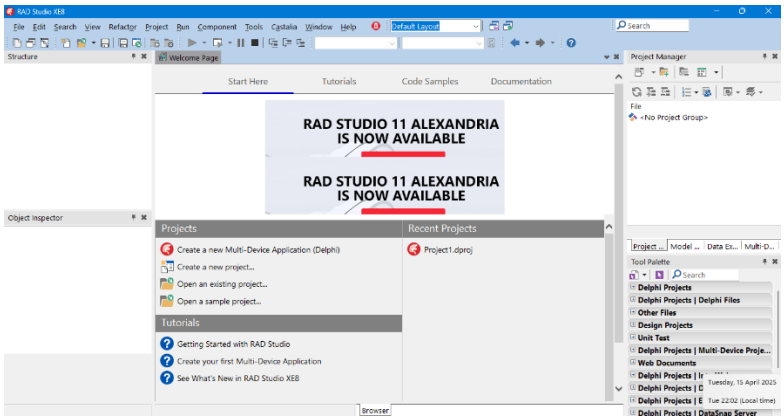
barang yang digunakan dalam RAB, seperti id_barang sebagai kunci utama, nama_barang, satuan, harga, dan pajak yang dikenakan pada barang tersebut. Dengan tabel ini, sistem dapat mencatat berbagai barang yang diperlukan dalam proyek RAB secara rinci.

Tabel terakhir, tbl_rab_detail, merupakan tabel penghubung antara tbl_uraian dan tbl_barang. Tabel ini berisi in*form*asi tentang barang yang digunakan dalam setiap uraian, termasuk id_detail sebagai kunci utama, serta id_uraian dan id_barang sebagai referensi ke tabel terkait. Selain itu, terdapat atribut volume yang menunjukkan jumlah barang yang digunakan, serta total_harga yang menunjukkan perhitungan total biaya berdasarkan volume barang yang digunakan. Dengan adanya tabel ini, sistem dapat menyusun detail dari setiap uraian dalam RAB dengan in*form*asi barang yang lebih spesifik.

C. Rancangan Sistem

Rancangan sistem adalah tahapan rancangan suatu aplikasi dari kebutuhan-kebutuhan fungsional hingga persiapan perancnagan

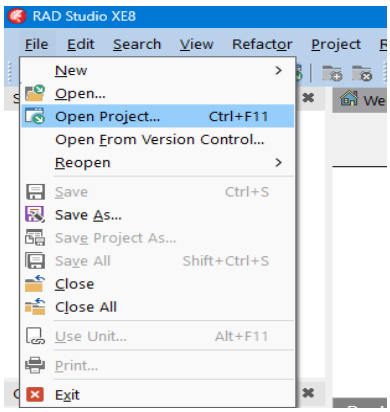
1. Tampilan aplikasi *delphi*



Gambar 4. 17 Tampilan Awal *delphi*

Tampilan awal *Delphi* diatas adalah tempat *project* di lakukan dengan berbgai fitur sehingga memudahkan pengguna mengakses fitur utama. Bagian atas berisi menu dan *toolbar* untuk navigasi cepat. Di tengah, terdapat opsi untuk membuat proyek baru, membuka proyek terakhir, serta terdapat rekomendasi pustaka dan komponen dari *GetIt Package Manager*.

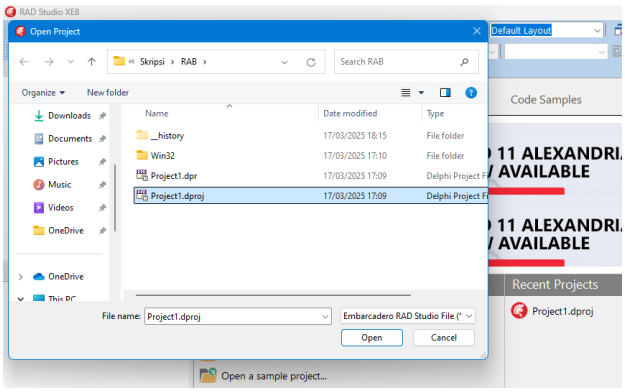
2. Open *project*



Gambar 4. 18 *Open Project*

Untuk mengakses *project* silahkan klik pada menu file dan memilih fitur Open *Project*, fitur ini berfungsi membuka file *project* yang telah di buat sebelumnya.

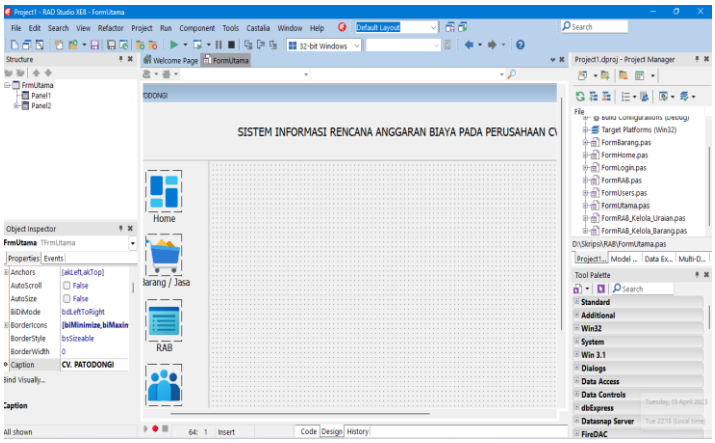
3. File *Project*



Gambar 4. 19 *File Project*

Setelah memilih open *project* pada menu file, kemudian akan diarahkan untuk mencari atau membuka file *project* yang telah disediakan.

4. Tampilan awal *project* pada aplikasi *delphi*



Gambar 4. 20 Tampilan awal *project* pada apliksi *delphi*

Setelah melewati beberapa tahap menuju tampilan awal *project*, pada tampilan diatas terdapat beberapa fitur yang diguankan sehingga menghasilkan gambaran seperti diatas, berikut beberapa tahapnya:

a. Desain *form* utama

Dengan mengubah caption menjadi CV. PATODONGI, mengatur Border style menjadi bsSizeable, dan mengaktifkan biMinize.

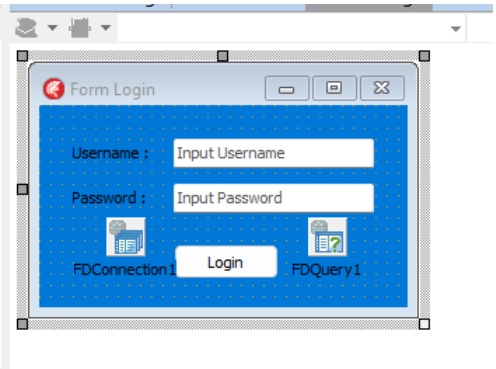
b. Menambahkan panel dan loyout

Tambahkan panel dari palatte dengan memilih standar dan Tpanel, Setalah itu tambahkan bagian *panel1* dan *panel2* untuk membagi *loyout* tampilan.

c. Menambahkan tombol navigasi

Gunakan Timage atau Tbutton untuk membuat tombol navigasi, setelah itu desain sesuai dengan rancangan *project* seperti gambar diatas meliputi: *Home*, Barang / jasa, RAB, *Users*, dan *logout*.

5. Rancangan *login*



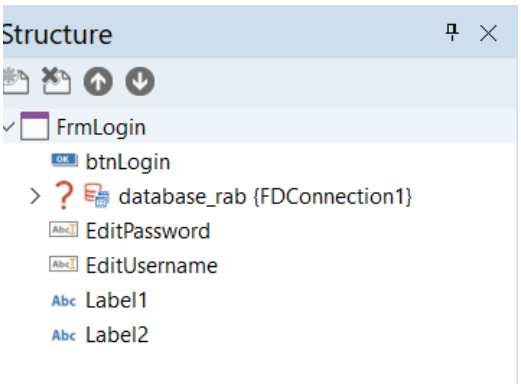
Gambar 4. 21 Rancangan *Login*

Beberapa tahap yang diperlukan dalam pembuatan rancangan *login* diatas yang meliputi:

a. *Form login*

1) Buat proyek baru di *Delphi* atau tambahkan *form* baru dengan nama *FrmLogin*
2) Tambahkan komponen dalam struktur berikut ini:
   a) (*EditUsername* & *EditPassword*) untuk input *Username* dan *Password*.
   b) *Username* (Label1) dan *Password* (Label2) untuk memberi .
   c) btn*Login* untuk tombol *login*.
   d) *Connection* untuk koneksi ke database berupa (database_ rab {FD*Connection*}

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan *form login* diatas:



Gambar 4. 22 Struktur *Form Login*

b. *Object Inspector*

Object Inspector adalah alat yang menampilkan dan mengelola properti dan gaya objek yang dipilih seperti berikut ini:

1) *Caption*

Untuk memilih nama yang muncul pada tampilan *form project* yaitu (*Form Login*).

2) *Color*

Color berfungsi untuk mengatur warna pada tampilan *project* untuk bagian *form login*.

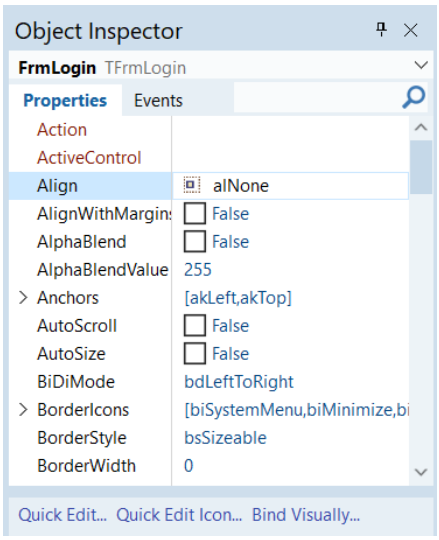
3) *Font*

Font berfungsi untuk mengatur gaya tulisan pada tampilan *project form login*.

4) *Name*

Name berfungsi untuk mengatur nama pemanggilan pada koneksi *project form login*.

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan *form login* diatas:



Gambar 4. 23 *Object Inspector Form Login*

c. Implementasi kode *login*

Klik pada bagian *code* untuk mendapatkan kode program pada tampilan *project form login*, berikut ini adalah bagian pembahasan mengenai kode implementasi *login* :

1) Deklarasi *form*

Mendefinisikan *form* bernama TFrm*Login*, turunan dari T*form*:

a) dit*Username* dan Edit*Password*: Input untuk *username* dan *Password*.

b) Label1, Label2: Label untuk menandai input.

c) btn*Login*: Tombol untuk men-trigger proses *login*.

d) FD*Connection*1: Komponen koneksi ke database *SQLite*.

e) FD*Query*1: Untuk menjalankan *query* ke database.

f) *Login*Status: Variabel boolean publik, digunakan untuk menyimpan status *login*.

2) Event: btn*loginClick*

a) Mengambil input user, dan menghapus spasi di awal/akhir string

b) Validasi Kosong (Jika salah satu kosong, tampilkan pesan error dan hentikan proses *login*)

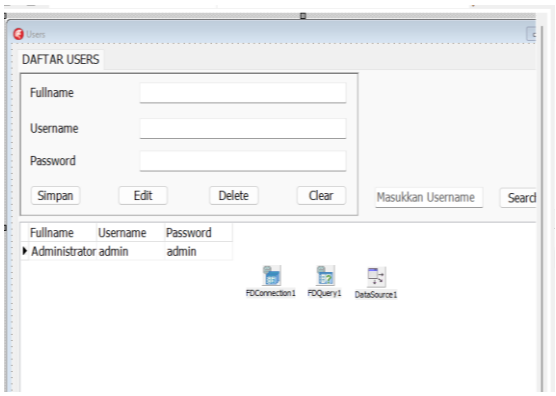
c) *Query* ke Database (Mengecek apakah kombinasi *username* dan *Password* ada di tabel)

d) Bind Parameter (Mengikat nilai parameter dengan input user)Eksekusi dan Cek Hasil (Jika hasil *query* (COUNT(*)) lebih dari 0, *login* berhasil, *form* akan ditutup (Close) dan *Login*Status diset True dan Jika tidak, tampilkan pesan gagal

3) Event: *Form*Create

Event ini dipanggil saat *form* dibuat, tetapi masih kosong. Biasanya bisa digunakan untuk inisialisasi seperti membuka koneksi database (FD*Connection*1.Connected := True), jika diperlukan.

6. Rancangan daftar user



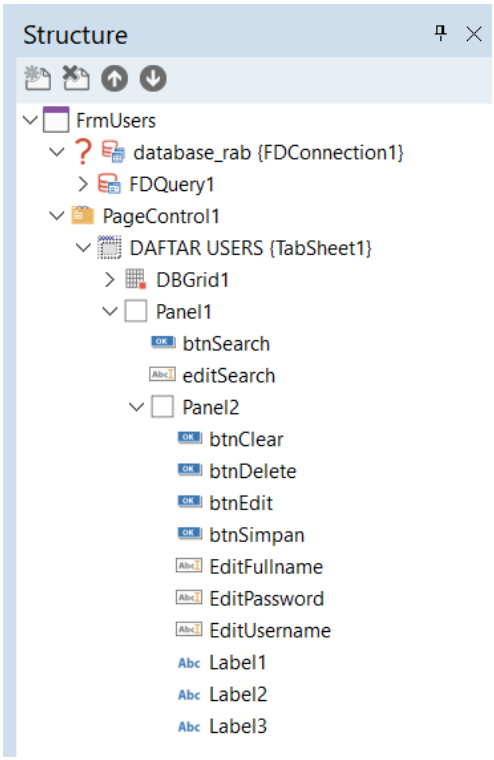
Gambar 4. 24. Rancangan Daftar *User*

Beberapa tahap yang diperlukan dalam pembuatan rancangan Daftar *User* diatas yang meliputi:

a. Buat *form* daftar *user*

1) Buat proyek baru di *Delphi* atau tambahkan *form* baru dengan nama Frm*Users*
2) Tambahkan komponen dalam struktur berikut ini:
   a) *Full name* (Label 3), *Username* (label 1), *Password* (label 2).
   b) *Button Fullname* (EditFullname), *Button Username* (Edit*Username*), dan *Button Password* (Edit*Password*)
   c) *Button* Simpan (btnsimpan), *Button Edit* (btnedit), *Button Delete* (btn*Delete*) dan *Button Clear* (btn*Clear*) .
   d) Masukkan *Username* (editSearch dan *Button Search* (btnSearch).
   e) *Connection* untuk koneksi ke database berupa (database_ rab {FD*Connection*}

   Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan Sturktur *Form User*:



Gambar 4. 25 Struktur *Form* Daftar *User*

b. Object Inspector

Object Inspector adalah alat yang menampilkan dan mengelola properti dan gaya objek yang dipilih seperti berikut ini:

1) *Caption*

Untuk memilih nama yang muncul pada tampilan *form project* yaitu (*Form User*).

2) *Color*

Color berfungsi untuk mengatur warna pada tampilan *project* untuk bagian *form User*.

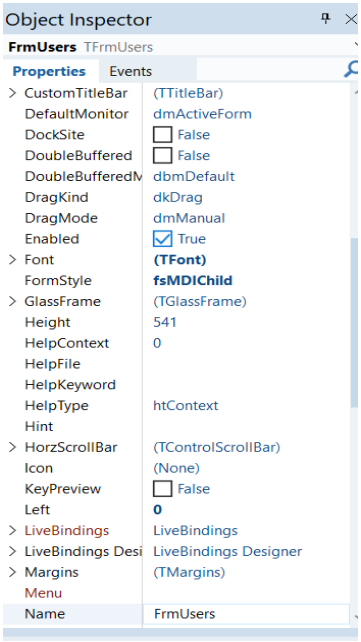
3) *Font*

Font berfungsi untuk mengatur gaya tulisan pada tampilan *project form User*

4) *Name*

Name berfungsi untuk mengatur nama pemanggilan pada koneksi *project form User*.

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan *form User* diatas:



Gambar 4. 26 *Object Inspector Form User*

c. Implementasi kode *form user*

Klik pada bagian *code* untuk mendapatkan kode program pada tampilan *project form User,* berikut ini adalah bagian pembahasan mengenai kode implementasi user :

1) Struktur umum

unit *FormUsers*;

Unit ini mendefinisikan sebuah *form* bernama TFrm*Users* yang digunakan untuk menangani data dari tabel *users* di database :

a) FD*Connection*1: Komponen koneksi ke database.

b) FD*Query*1: Untuk menjalankan *query* SQL.

c) Data*Source*1: Menghubungkan FD*Query*1 ke komponen visual seperti DBGrid1.

d) DBGrid1: Menampilkan data tabel *users* dalam bentuk grid.

e) EditFullname, Edit*Username*, Edit*Password*: Komponen input untuk menambah/edit data *user*.

f) btnSimpan, btnEdit, btn*Delete*, btnSearch, btn*Clear*: Tombol untuk aksi CRUD

2) Fungsi CRUD dan Fitur

a) BtnSearchClik(fitur pencarian)

procedure TFrm*Users*.btnSearch*Click*(Sender: TObject);

(1) *Query* dijalankan untuk mencari data user berdasarkan input pada editSearch.

(2) SQL LIKE digunakan agar pencarian bisa fleksibel (tidak harus persis).

(3) % digunakan sebagai wildcard untuk mencocokkan sebagian teks

b) btnSimpan*Click* (Menyimpan Data Baru)

***procedure* TFrm*Users*.btnSimpan*Click*(Sender: TObject);**

(1) FD*Query*1.Append digunakan untuk menambahkan data baru.

(2) Nilai dari inputan (fullname, *username*, *Password*) dimasukkan ke field *query*.

(3) Post menyimpan data ke database

c) btneEdit*Click* (Mengubah data yang dipilih)

***procedure* TFrm*Users*.btnEdit*Click*(Sender: TObject);**

(1) FD*Query*1.Edit masuk ke mode edit data.

(2) Kemudian field diubah sesuai nilai input.

(3) Post menyimpan perubahan ke database

3) Alur umum

a) *Form* dibuka, data user ditampilkan di DBGrid1

b) User dapat:

(1) Menambah user baru (btnSimpan)

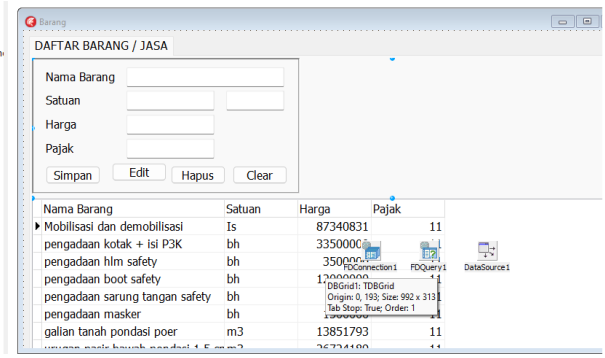
(2) Mencari user berdasarkan *username* (btnSearch)

(3) Mengedit data dengan klik pada grid lalu tekan btnEdit

(4) Menghapus data dengan btn*Delete*

(5) Membersihkan *form* input dengan btn*Clear*

7. Hasil rancangan barang dan jasa



Gambar 4. 27 Rancangan Barang dan Jasa

Beberapa tahap yang diperlukan dalam pembuatan rancangan Daftar *User* diatas yang meliputi:

a. Buat *form* barang dan jasa

1) Buat proyek baru di *Delphi* atau tambahkan *form* baru dengan nama FrmBarang

2) Tambahkan komponen dalam struktur berikut ini:

   a) Nama Barang (Label1), Satuan (label2), Harga (label3) dan Pajak (Label4)

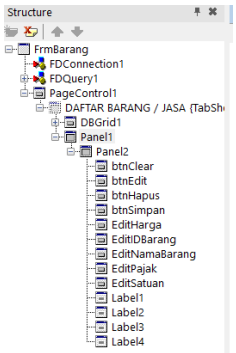
   b) Tedit nama barang (EditNamaBarang), Tedit Satuan (EditSatuan), Tedit Harga (EditHarga), dan Tedit Pajak (EditPajak).

   c) Button Simpan (btnSimpan), Button Edit (btnEdit), Button Hapus (btnHapus) dan Button *Clear* (btn*Clear*).

   d) DbGrind1 (adalah merupakan data dari hasil input) Nama Barang, Satuan, Haga dan Pajak.

   e) *Connection* untuk koneksi ke database berupa (database_ rab {FD*Connection*}

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan Sturktur *Form* Jasa dan Barang:



Gambar 4. 28 Struktur *Form* Jasa Barang

b. Object Inspector

Object Inspector adalah alat yang menampilkan dan mengelola properti dan gaya objek yang dipilih seperti berikut ini:

a) *Caption*

Untuk memilih nama yang muncul pada tampilan *form project* yaitu (*Form* Jasa Barang).

b) *Color*

Color berfungsi untuk mengatur warna pada tampilan *project* untuk bagian *form* Jasa Barang.

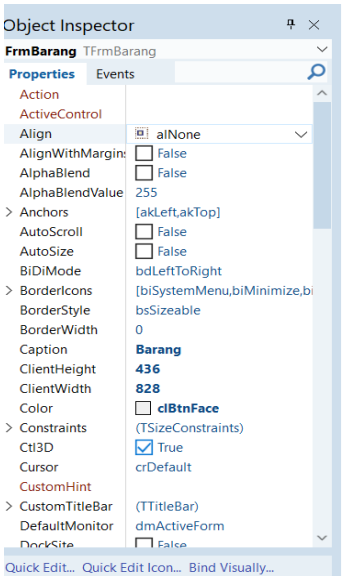
c) *Font*

Font berfungsi untuk mengatur gaya tulisan pada tampilan *project* Jasa Barang

d) *Name*

Name berfungsi untuk mengatur nama pemanggilan pada koneksi *project* Jasa Barang.

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan *form* Jasa Barang diatas:



Gambar 4. 29 *Object Inspector Form* Jasa Barang

c. Implementasi kode barang dan jasa

Klik pada bagian *code* untuk mendapatkan kode program pada tampilan *project form* jasa barang, berikut ini adalah bagian pembahasan mengenai kode implementasi rancangan Barang dan jasa :

1) btnSimpan*Click* (Menyimpan Barang Baru)

   a) *procedure* TFrmBarang.btnSimpan*Click*(Sender: TObject);

   b) Validasi bahwa semua field (NamaBarang, Satuan, Harga, Pajak) tidak kosong.

   c) Menyusun *query* INSERT INTO tbl_barang untuk menambahkan data baru ke database.

   d) Mengisi parameter *query* dari input pengguna.

   e) Menjalankan *query* dengan *ExecSQL*.

   f) Menampilkan pesan sukses atau pesan error.

   g) Membersihkan *form* input.

h) Memuat ulang data melalui *LoadData*

2) btnHapusClik (Menghapus data barang)

***procedure* TFrmBarang.btnHapus*Click*(Sender: TObject);**

a) Memastikan EditIDBarang tidak kosong (artinya data dipilih).

b) Menampilkan konfirmasi menggunakan MessageDlg.

c) Menyusun *query DELETE* FROM tbl_barang WHERE id_barang = :id_barang.

d) Menjalankan *ExecSQL* jika dikonfirmasi.

e) Membersihkan *form* input dan memuat ulang data.

3) loadData (Menampilkan Data)

***procedure* TFrmBarang.LoadData;**

a) Menjalankan *query* untuk mengambil semua data barang dari tabel tbl_barang.

b) Digunakan untuk refresh isi grid setelah simpan atau hapus.

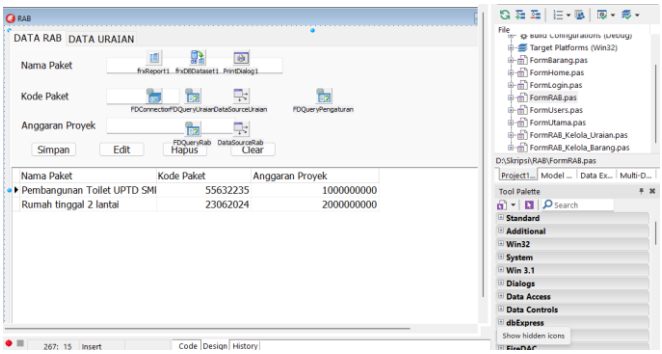
4) DBGGrid1Cell*Click* (Menyalin data ke *Form*)

***procedure* TFrmBarang.DBGrid1Cell*Click*(Column: TColumn);**

a) Ketika user klik salah satu baris di DBGrid1, data dari baris tersebut akan ditampilkan kembali ke *form* input.

Berguna untuk melihat detail atau menghapus berdasarkan id_barang.

8. Rancangan Rencana Anggaran Biaya (RAB)



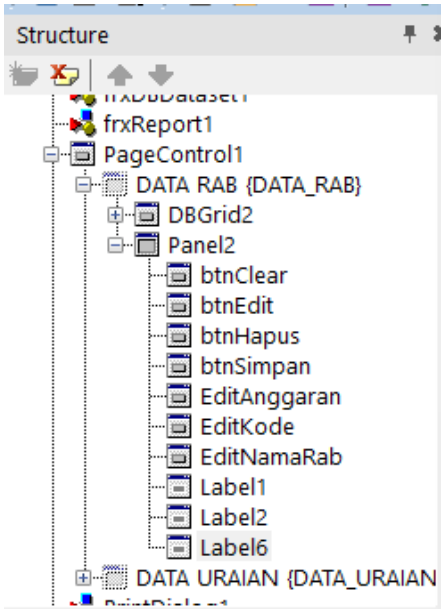
Gambar 4. 30 Rancangan Rencana Anggaran Biaya (RAB)

Beberapa tahap yang diperlukan dalam pembuatan rancangan Rencana Anggaran Biaya (RAB) diatas yang meliputi:

a. Buat *form* RAB

1) Buat proyek baru di *Delphi* atau tambahkan *form* baru dengan nama FrmRAB
2) Tambahkan komponen dalam struktur berikut ini:
   a) Nama Paket (Label1) dan Kode Paket (Label2) dan anggaran (Label6)
   b) Tedit Nama Paket (EditNamaRab) dan Tedit Kode Paket (EditKode), Tedit Anggaran (EditAnggaran)
   c) *Button* Simpan (btnSimpan), *Button* Edit (btnEdit), *Button* Hapus (btnHapus) dan *Button Clear* (btn*Clear*).
   d) DbGrind2 (adalah merupakan data dari hasil *input*) Nama Paket, dan Kode Paket, anggaran
   e) *Connection* untuk koneksi ke database berupa (database_ rab {FD*Connection*1}

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan Sturktur *Form* Jasa dan Barang:



Gambar 4. 31 Struktur *Form* RAB

b. Object Inspector

Object Inspector adalah alat yang menampilkan dan mengelola properti dan gaya objek yang dipilih seperti berikut ini:

1) Caption

Untuk memilih nama yang muncul pada tampilan *form project* yaitu (*Form* RAB).

2) Color

Color berfungsi untuk mengatur warna pada tampilan *project* untuk bagian *form* RAB.

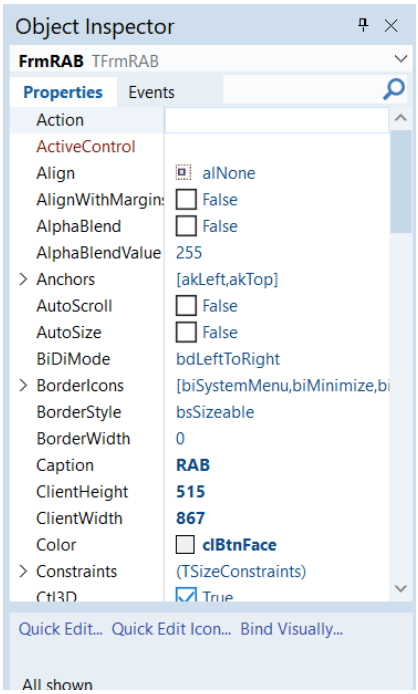
3) Font

Font berfungsi untuk mengatur gaya tulisan pada tampilan *project* RAB

4) *Name*

Name berfungsi untuk mengatur nama pemanggilan pada koneksi *project* RAB.

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan *form* RAB diatas:



Gambar 4. 32 *Object Inspector Form* RAB

c. Implementasi Kode *Login*

Klik pada bagian *code* untuk mendapatkan kode program pada tampilan *project form* RAB, berikut ini adalah bagian pembahasan mengenai kode implementasi rancangan Rencana Anggaran Biaya (RAB) :

1) *procedure* TFrmRAB.DATA_RABShow(Sender: TObject);

Saat tab DATA_RAB ditampilkan, *query* FD*Query*Rab dijalankan untuk mengambil semua data dari tabel tbl_rab.

2) *procedure* TFrmRAB.btnSimpan*Click*(Sender: TObject);

   Digunakan untuk menambahkan RAB baru ke tabel tbl_rab.

3) *procedure* TFrmRAB.btnEdit*Click*(Sender: TObject);

   Memungkinkan pengguna untuk mengedit data RAB yang dipilih.

4) *procedure* TFrmRAB.btnHapus*Click*(Sender: TObject);

   Menampilkan dialog konfirmasi sebelum menghapus data RAB yang dipilih.

5) *procedure* TFrmRAB.btn*ClearClick*(Sender: TObject);

   Membersihkan isian input RAB (kode dan nama).

6) *procedure* TFrmRAB.DBGrid2Cell*Click*(Column: TColumn);

   Saat pengguna klik data di grid, nilai nama dan kode akan dimuat ke input.

7) *procedure* TFrmRAB.DATA_URAIANShow(Sender: TObject);

   Menampilkan semua RAB dalam ComboBox1 saat tab DATA_URAIAN dibuka.

8) *procedure* TFrmRAB.ComboBox1Change(Sender: TObject);

   Saat pengguna pilih salah satu RAB di combobox, uraian + barang ditampilkan berdasarkan ID RAB terpilih.

   a) Ambil ID dari objek combobox.

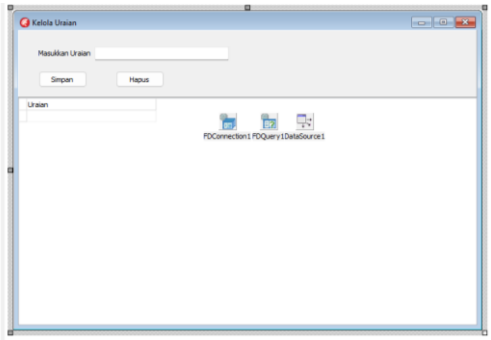
   b) *Query* data RAB (kode, dll).

c) Jalankan *query* gabungan CTE (Common Table Expression) untuk menampilkan:

(1) Baris “Tambah Barang” (berdasarkan uraian)

(2) Baris barang (dari tbl_rab_detail + tbl_barang)

9. Rancangan Kelola Uraian



Gambar 4. 33 Rancangan Kelola Uraian

Beberapa tahap yang diperlukan dalam pembuatan rancangan kelola Uraian diatas yang meliputi:

a. Buat *form* KelolaUraian

1) Buat proyek baru di *Delphi* atau tambahkan *form* baru dengan nama FrmKelolaUraian

2) Tambahkan komponen dalam struktur berikut ini:

a) Masukkan Uraian (Label1)

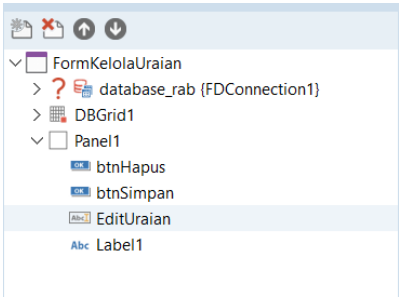
b) Tedit Masukan Uraian (EditUraian).

c) Button Simpan (btnSimpan) dan Button Hapus (btnHapus)

d) DbGrind1 (adalah merupakan data dari hasil input) Uraian.

e) *Connection* untuk koneksi ke database berupa (database_ rab {FD*Connection*1}

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan Sturktur *Form* Kelola Uraian:



Gambar 4. 34 Struktur *Form* Kelola Uraian

b. Object inspector

Object Inspector adalah alat yang menampilkan dan mengelola properti dan gaya objek yang dipilih seperti berikut ini:

1) Caption

Untuk memilih nama yang muncul pada tampilan *form project* yaitu (*Form* Kelola Uraian).

2) Color

Color berfungsi untuk mengatur warna pada tampilan *project* untuk bagian *form* Kelola Uraian.

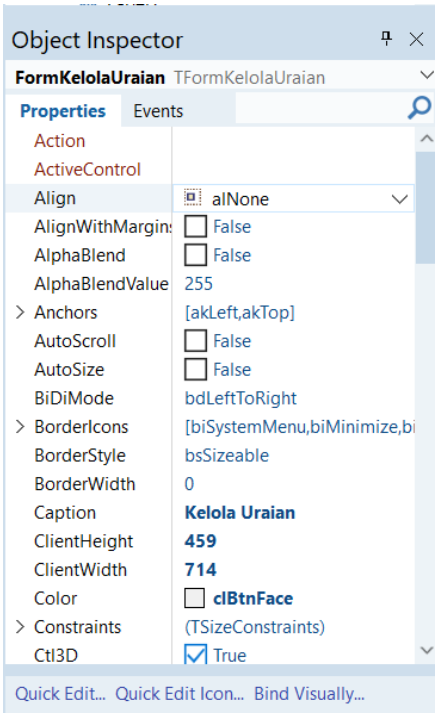
3) Font

Font berfungsi untuk mengatur gaya tulisan pada tampilan *project* Kelola Uraian.

4) Name

Name berfungsi untuk mengatur nama pemanggilan pada koneksi *project* Kelola Uraian.

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan *form* Kelola Uraian diatas:



Gambar 4. 35 *Object Inspector Form* Kelola Uraian

c. Implementasi kode kelola uraian

Klik pada bagian *code* untuk mendapatkan kode program pada tampilan *project form* Kelola Uraian, berikut ini adalah bagian pembahasan mengenai kode implementasi rancangan Kelola Uraian :

1) *procedure* LoadData(ASelectedID: Integer);

   FSelectedID := ASelectedID; (Simpan ID RAB ke variabel internal), **FD*Query*1.SQL.Text := 'SELECT * FROM tbl_uraian WHERE id_rab = :id'**; (*Query* semua uraian berdasarkan RAB yang dipilih).

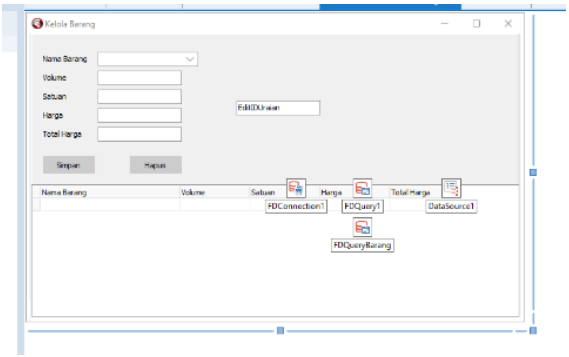
2) btnSimpan*Click* (Digunakan untuk menambahkan uraian baru)

if Trim(EditUraian.Text) = '' then ... (Validasi agar input tidak kosong), FD*Query*1.SQL.Text := 'SELECT * FROM tbl_uraian WHERE id_rab = :id'; (Refresh data setelah berhasil menyimpan).

3) btnHapus*Click* (Digunakan untuk menghapus uraian yang dipilih dari grid).

if FD*Query*1.IsEmpty then ... (Cegah penghapusan jika tidak ada baris yang dipilih), FD*Query*1.*Delete*; (Hapus baris aktif dari dataset FD*Query*1 (yang ditampilkan di DBGrid1).

10. Rancangan Kelola Barang



Gambar 4. 36 Rancangan Kelola Barang

Beberapa tahap yang diperlukan dalam pembuatan rancangan kelola Barang diatas yang meliputi:

a. Buat *form* kelolabarang

1) Buat proyek baru di *Delphi* atau tambahkan *form* baru dengan nama FrmKelolaBarang

2) Tambahkan komponen dalam struktur berikut ini:

f) Nama Barang (Label1), *Volume* (Label2), Satuan (Label3), Harga (Label4) dan Total Harga (Label5).

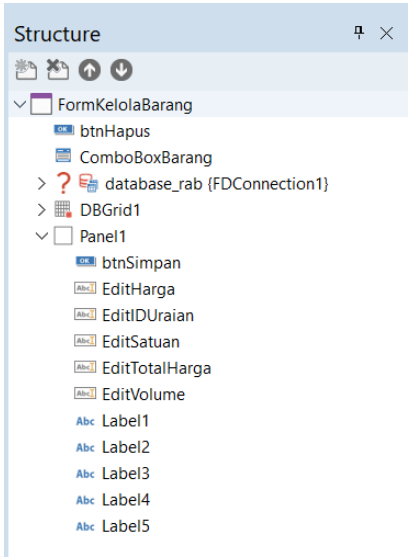
g) Tedit Nama Barang (EditNamaBarang), Tedit Volume (EditVolume), Tedit Satuan (EditSatuan), Tedit Harga (EditHarga) dan Tedit Total Harga (EditTotalHarga).

h) *Button* Simpan (btnSimpan) dan *Button* Hapus (btnHapus)

i) DbGrind1 (adalah merupakan data dari hasil *input*) Nama Barang, *Volume*, Satuan, Harga dan Total Harga.

j) *Connection* untuk koneksi ke database berupa (database_ rab {FD*Connection*1}

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan Sturktur *Form* Kelola Barang:



Gambar 4. 37 Struktur *Form* Kelola Barang

b. Object Inspector

Object Inspector adalah alat yang menampilkan dan mengelola properti dan gaya objek yang dipilih seperti berikut ini:

1) *Caption*

Untuk memilih nama yang muncul pada tampilan *form project* yaitu (*Form* Kelola Barang).

2) *Color*

Color berfungsi untuk mengatur warna pada tampilan *project* untuk bagian *form* Kelola Barang.

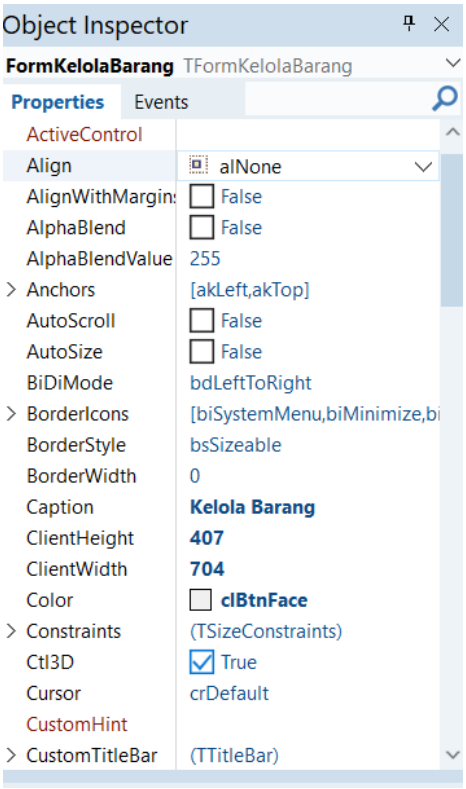
3) *Font*

Font berfungsi untuk mengatur gaya tulisan pada tampilan *project* Kelola Barang.

4) *Name*

Name berfungsi untuk mengatur nama pemanggilan pada koneksi *project* Kelola Barang.

Gambar berikut ini adalah bentuk struktur yang telah dijelaskan melalui proses pembuatan *form* Kelola Barang diatas:



Gambar 4. 38 *Object Inspector Form* Kelola Barang

c. Implementasi Kode Kelola Barang

Klik pada bagian *code* untuk mendapatkan kode program pada tampilan *project form* Kelola Barang, berikut ini adalah bagian pembahasan mengenai kode implementasi rancangan Kelola Uraian :

1) Variabel dan Properti

FSelectedID: Integer; (Menyimpan nilai id_uraian yang aktif dari *form* sebelumnya).

2) *procedure* LoadData(ASelectedID: Integer);

a) Menerima id_uraian sebagai parameter dan menyimpan ke FSelectedID.

b) Menampilkan data detail barang dari tbl_rab_detail yang sesuai.

c) Mengisi EditIDUraian agar bisa digunakan saat insert data.

3) *Form*Show

ComboBoxBarang.Items.AddObject(...);

a) Di-trigger saat *form* muncul.

b) Mengisi ComboBoxBarang dengan data dari tabel tbl_barang.

c) Menyimpan id_barang ke dalam Objects tiap item di combobox.

4) ComboBoxBarangChange

a) Mengambil `id_barang` dari `ComboBox`.

b) Menampilkan satuan dan harga dari barang yang dipilih.

c) Memanggil HitungTotal untuk update total harga langsung.

5) HitungTotal

Total := Volume * Harga;

EditTotalHarga.Text := Total.ToString;

a) Mengambil nilai volume dan harga.

b) Menghitung total harga dan menampilkannya.

6) EditVolumeKeyUp

a) Setiap user mengetik di EditVolume, maka total harga otomatis dihitung ulang.

7) btnSimpan*Click* (Menyimpan barang ke dalam detail uraian (tbl_rab_detail).

a) Validasi apakah barang sudah dipilih dan input numerik valid.

b) Menggunakan StartTransaction dan Commit untuk memastikan integritas data.

c) Insert data ke tbl_rab_detail.

d) Refresh data grid setelah simpan.

8) btnHapus*Click* (Menghapus satu baris data dari grid).

a) Ambil id_detail dari baris yang dipilih di grid.

b) Tampilkan konfirmasi penghapusan.

c) Jalankan *DELETE* FROM tbl_rab_detail WHERE id_detail = :id_detail. Refresh grid setelah hapus.

D. Desain Sistem

1. Tampilan *Login*

Tampilan *login* menggambarkan pintu masuk sistem. *Form* input untuk *username* dan *Password* serta tombol *login* memberi kesan sistematis dan profesional. Tampilan ini menegaskan bahwa hanya pihak yang berwenang yang dapat mengakses fitur-fitur penting dalam sistem.



Gambar 4. 39 Tampilan *Login*

```
      var
        FrmLogin: TFrmLogin;
      implementation
      {$R *.dfm}
      procedure TFrmLogin.btnLoginClick(Sender: TObject);
      var
        username, password: string;
      begin
        username := Trim(EditUsername.Text);
        password := Trim(EditPassword.Text);
        // Pastikan input tidak kosong
        if (username = '') or (password = '') then
        begin
          ShowMessage('Username dan password harus diisi!');
          Exit;
        End;
        //  Query  ke  database  untuk  mengecek  username  dan
      password
        FDQuery1.Close;
        FDQuery1.SQL.Text  :=  'SELECT  COUNT(*)  FROM  users  WHERE
      username = :uname AND password = :pass';
        FDQuery1.ParamByName('uname').AsString := username;
        FDQuery1.ParamByName('pass').AsString := password;
        FDQuery1.Open;
        if FDQuery1.Fields[0].AsInteger > 0 then
        begin
          LoginStatus := True;
          Close;
        end
        else
        begin
          ShowMessage('Username atau password salah!');
          LoginStatus := False;
        end;
      end;
end.
```

2. Tampilan *User*



Gambar 4. 40 Tampilan *User*

```
      implementation
      procedure TFrmUsers.btnSearchClick(Sender: TObject);
      begin
```

```
        FDQuery1.SQL.Clear;
        FDQuery1.Open('SELECT * FROM users WHERE username LIKE
      :username', ['%'+ editSearch.Text +'%']);
        FDQuery1.Active := True;
      end;
      procedure TFrmUsers.btnSimpanClick(Sender: TObject);
      begin
        FDQuery1.Append;
        FDQuery1.FieldByName('fullname').AsString             :=
      EditFullname.Text;
        FDQuery1.FieldByName('username').AsString             :=
      EditUsername.Text;
        FDQuery1.FieldByName('password').AsString             :=
      EditPassword.Text;
        FDQuery1.Post;
      end;
      procedure TFrmUsers.btnEditClick(Sender: TObject);
      begin
        FDQuery1.Edit;
        FDQuery1.FieldByName('fullname').AsString             :=
      EditFullname.Text;
        FDQuery1.FieldByName('username').AsString             :=
      EditUsername.Text;
        FDQuery1.FieldByName('password').AsString             :=
      EditPassword.Text;
        FDQuery1.Post;
      end;
      procedure TFrmUsers.btnClearClick(Sender: TObject);
      begin
        EditFullname.Text := '';
        EditUsername.Text := '';
        EditPassword.Text := '';
      End;
      procedure TFrmUsers.btnDeleteClick(Sender: TObject);
      begin
        if  MessageDlg('Yakin  akan  menghapus  data  ini?',
      mtConfirmation, [mbYes, mbNo], 0) = mrYes then
        FDQuery1.Delete;
      end;
      procedure TFrmUsers.DBGrid1CellClick(Column: TColumn);
      begin
        EditFullname.Text                                     :=
      FDQuery1.FieldByName('fullname').AsString;
        EditUsername.Text                                     :=
      FDQuery1.FieldByName('username').AsString;
        EditPassword.Text                                     :=
      FDQuery1.FieldByName('password').AsString;
      end;
end.
```

Tampilan *form* user adalah simbol manajemen dalam sistem. Dengan layout *form* di sebelah kiri dan tabel data di kanan, desain ini menyiratkan dualitas fungsi: input data baru dan pengelolaan data lama. *Form* ini terdiri

dari fullname, *username* dan *Password*. Tampilan ini mencerminkan pentingnya mengatur siapa saja yang memiliki hak akses ke sistem.

3. Tampilan *home*



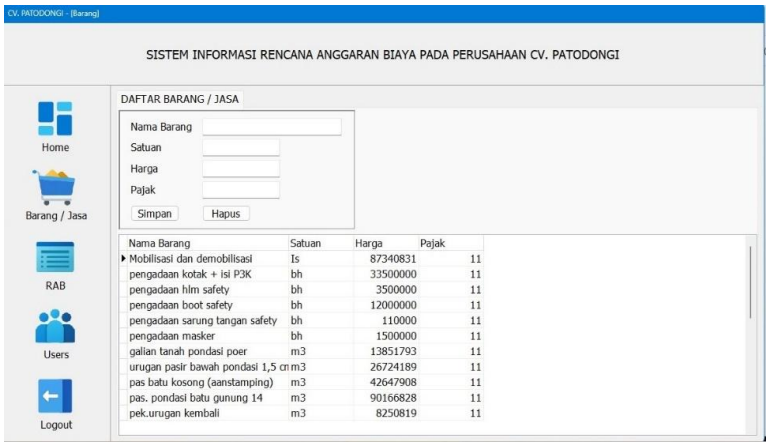
Gambar 4. 41 Menu *Home*

```
      interface
      uses
        Winapi.Windows,     Winapi.Messages,     System.SysUtils,
      System.Variants, System.Classes, Vcl.Graphics,
        Vcl.Controls,   Vcl.Forms,   Vcl.Dialogs,   Vcl.StdCtrls,
      Vcl.ComCtrls,
        Vcl.Imaging.jpeg, Vcl.ExtCtrls;
      type
        TFrmHome = class(TForm)
          PageControl1: TPageControl;
          HOME: TTabSheet;
          Image1: TImage;
        private
          { Private declarations }
        public
          { Public declarations }
        end;
      var
        FrmHome: TFrmHome;
      implementation
      {$R *.dfm}
end.
```

Tampilan *home* menjadi beranda utama atau pusat navigasi sistem. Terdapat tombol/menu utama seperti *Home*, Barang dan Jasa, RAB, *Users*, *Logout*, yang memudahkan pengguna memilih fitur yang ingin digunakan.

Layout-nya yang rapi dan terstruktur menandakan sistem ini dirancang untuk kenyamanan dan efisiensi penggunaan.

4. Tampilan Barang dan Jasa



Gambar 4. 42 Tampilan Barang dan Jasa

```
implementation
procedure TFrmBarang.btnSimpanClick(Sender: TObject);
begin
  if Trim(EditNamaBarang.Text) = '' then
  begin
    ShowMessage('Nama barang harus diisi!');
    Exit;
  end;
  if Trim(EditSatuan.Text) = '' then
  begin
    ShowMessage('Satuan harus diisi!');
    Exit;
  end;
  if Trim(EditHarga.Text) = '' then
  begin
    ShowMessage('Harga harus diisi!');
    Exit;
  end;
  if Trim(EditPajak.Text) = '' then
  begin
    ShowMessage('Pajak harus diisi!');
    Exit;
  end;
  FDQuery1.Close;
  FDQuery1.SQL.Text :=
    'INSERT INTO tbl_barang (nama_barang, satuan, harga,
pajak) ' +
    'VALUES (:nama_barang, :satuan, :harga, :pajak)';
  FDQuery1.ParamByName('nama_barang').AsString          :=
EditNamaBarang.Text;
  FDQuery1.ParamByName('satuan').AsString               :=
```

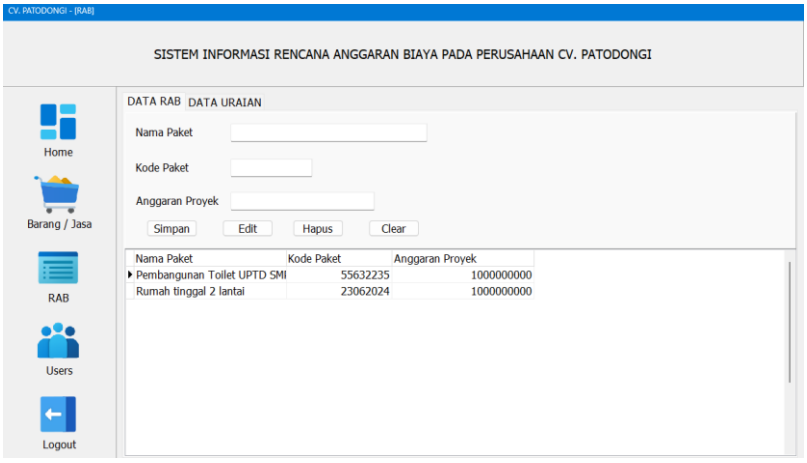
```
      EditSatuan.Text;
        FDQuery1.ParamByName('harga').AsInteger               :=
      StrToIntDef(EditHarga.Text, 0);
        FDQuery1.ParamByName('pajak').AsInteger               :=
      StrToIntDef(EditPajak.Text, 0);
        try
          FDQuery1.ExecSQL;
          ShowMessage('Barang berhasil disimpan.');
          EditNamaBarang.Clear;
          EditSatuan.Clear;
          EditHarga.Clear;
          EditPajak.Clear;
          LoadData;
        except
          on E: Exception do
            ShowMessage('Gagal menyimpan barang: ' + E.Message);
        end;
      end;
      procedure TFrmBarang.DBGrid1CellClick(Column: TColumn);
      begin
        if not FDQuery1.IsEmpty then
        begin
          EditIDBarang.Text                                   :=
      FDQuery1.FieldByName('id_barang').AsString;
          EditNamaBarang.Text                                 :=
      FDQuery1.FieldByName('nama_barang').AsString;
          EditSatuan.Text                                     :=
      FDQuery1.FieldByName('satuan').AsString;
          EditHarga.Text                                      :=
      FDQuery1.FieldByName('harga').AsString;
          EditPajak.Text                                      :=
      FDQuery1.FieldByName('pajak').AsString;
        end;
      end;
      procedure TFrmBarang.btnClearClick(Sender: TObject);
      begin
        EditIDBarang.Clear;
        EditNamaBarang.Clear;
        EditSatuan.Clear;
        EditHarga.Clear;
        EditPajak.Clear;
        EditNamaBarang.SetFocus;
      end;
      procedure TFrmBarang.btnEditClick(Sender: TObject);
      begin
        if Trim(EditIDBarang.Text) = '' then
        begin
          ShowMessage('Pilih data yang akan diedit!');
          Exit;
        end;
        if Trim(EditNamaBarang.Text) = '' then
        begin
          ShowMessage('Nama barang harus diisi!');
          Exit;
        end;
        if Trim(EditSatuan.Text) = '' then
```

```
  begin
    ShowMessage('Satuan harus diisi!');
    Exit;
  end;
  if Trim(EditHarga.Text) = '' then
  begin
    ShowMessage('Harga harus diisi!');
    Exit;
  end;
  if Trim(EditPajak.Text) = '' then
  begin
    ShowMessage('Pajak harus diisi!');
    Exit;
  end;
  FDQuery1.Close;
  FDQuery1.SQL.Text :=
    'UPDATE tbl_barang SET nama_barang = :nama_barang,
satuan = :satuan, ' +
    'harga = :harga, pajak = :pajak WHERE id_barang =
:id_barang';
  FDQuery1.ParamByName('nama_barang').AsString          :=
EditNamaBarang.Text;
  FDQuery1.ParamByName('satuan').AsString               :=
EditSatuan.Text;
  FDQuery1.ParamByName('harga').AsInteger               :=
StrToIntDef(EditHarga.Text, 0);
  FDQuery1.ParamByName('pajak').AsInteger               :=
StrToIntDef(EditPajak.Text, 0);
  FDQuery1.ParamByName('id_barang').AsInteger           :=
StrToIntDef(EditIDBarang.Text, 0);
  try
    FDQuery1.ExecSQL;
    ShowMessage('Data barang berhasil diperbarui.');
    LoadData;
    btnClearClick(Sender);
  except
    on E: Exception do
      ShowMessage('Gagal memperbarui barang: ' +
E.Message);
  end;
end;
procedure TFrmBarang.btnHapusClick(Sender: TObject);
begin
  if Trim(EditIDBarang.Text) = '' then
  begin
    ShowMessage('Pilih data terlebih dahulu!');
    Exit;
  end;
  if MessageDlg('Apakah Anda yakin ingin menghapus barang
ini?', mtConfirmation, [mbYes, mbNo], 0) = mrNo then
    Exit;
  FDQuery1.Close;
  FDQuery1.SQL.Text := 'DELETE FROM tbl_barang WHERE
id_barang = :id_barang';
  FDQuery1.ParamByName('id_barang').AsInteger           :=
StrToIntDef(EditIDBarang.Text, 0);
```

```
        try
          FDQuery1.ExecSQL;
          ShowMessage('Barang berhasil dihapus.');
          EditIDBarang.Clear;
          EditNamaBarang.Clear;
          EditSatuan.Clear;
          EditHarga.Clear;
          EditPajak.Clear;
          LoadData;
        except
          on E: Exception do
            ShowMessage('Gagal menghapus barang: ' + E.Message);
        end;
      end;
      procedure TFrmBarang.LoadData;
      begin
        FDQuery1.Close;
        FDQuery1.SQL.Text  :=  'SELECT  id_barang,  nama_barang,
      satuan, harga, pajak FROM tbl_barang';
        FDQuery1.Open;
      end;
end.
```

Tampilan ini merupakan representasi data sumber daya proyek. Melalui *form* input Nama Barang, Satuan, Harga, Pajak dan tabel, pengguna bisa menambah dan melihat data barang/material atau jasa. Hal ini mencerminkan pentingnya pengelolaan in*form*asi bahan untuk menghasilkan perhitungan anggaran yang tepat.

5. Tampilan Rencana Anggaran Biaya



Gambar 4. 43 Rencana Anggaran Biaya (RAB)

```
uses FormRAB_Kelola_Uraian, FormRAB_Kelola_Barang;
procedure TFrmRAB.FormShow(Sender: TObject);
begin
  EditSisa.Text := IntToStr(
    StrToIntDef(EditRABUraianAnggaran.Text, 0) -
    StrToIntDef(EditTotal.Text, 0)
  );
  FDQueryPengaturan.Close;
  FDQueryPengaturan.SQL.Text := 'SELECT nama_direktur FROM
tbl_pengaturan WHERE id_pengaturan = 1';
  FDQueryPengaturan.Open;
  if not FDQueryPengaturan.IsEmpty then
      EditNamaDirektur.Text                                    :=
FDQueryPengaturan.FieldByName('nama_direktur').AsString;
end;
procedure TFrmRAB.DATA_RABShow(Sender: TObject);
begin
  FDQueryRab.Close;
  FDQueryRab.SQL.Text := 'SELECT * FROM tbl_rab';
  FDQueryRab.Open;
end;
procedure TFrmRAB.btnSimpanClick(Sender: TObject);
begin
  FDQueryRab.Append;
  FDQueryRab.FieldByName('nama_rab').AsString              :=
EditNamaRab.Text;
  FDQueryRab.FieldByName('kode').AsString := EditKode.Text;
  FDQueryRab.FieldByName('anggaran').AsString              :=
EditAnggaran.Text;
  FDQueryRab.Post;
end;
procedure TFrmRAB.btnEditClick(Sender: TObject);
begin
  FDQueryRab.Edit;
  FDQueryRab.FieldByName('nama_rab').AsString              :=
EditNamaRab.Text;
  FDQueryRab.FieldByName('kode').AsString := EditKode.Text;
  FDQueryRab.FieldByName('anggaran').AsString              :=
EditAnggaran.Text;
  FDQueryRab.Post;
end;
procedure TFrmRAB.btnHapusClick(Sender: TObject);
begin
  if   MessageDlg('Yakin   akan   menghapus   data   ini?',
mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  FDQueryRab.Delete;
end;
procedure TFrmRAB.btnClearClick(Sender: TObject);
begin
  EditNamaRab.Text := '';
  EditKode.Text := '';
  EditAnggaran.Text := '';
end;
procedure TFrmRAB.DBGrid2CellClick(Column: TColumn);
begin
  EditNamaRab.Text                                         :=
```

```
FDQueryRab.FieldByName('nama_rab').AsString;
  EditKode.Text := FDQueryRab.FieldByName('kode').AsString;
  EditAnggaran.Text                                          :=
FDQueryRab.FieldByName('anggaran').AsString;
end;
procedure TFrmRAB.EditTotalChange(Sender: TObject);
begin
  EditSisa.Text := IntToStr(
    StrToIntDef(EditRABUraianAnggaran.Text, 0) -
    StrToIntDef(EditTotal.Text, 0)
  );
end;
procedure  TFrmRAB.frxUserDataSet1CheckEOF(Sender:  TObject;
var Eof: Boolean);
begin
end;
procedure TFrmRAB.HitungTotal;
var
  total, nilai: Double;
  i: Integer;
begin
  total := 0;
  for i := 0 to DBGrid1.DataSource.DataSet.RecordCount - 1
do
  begin
    DBGrid1.DataSource.DataSet.RecNo := i + 1;
    nilai := StrToFloatDef(DBGrid1.Fields[7].AsString, 0);
    total := total + nilai;
  end;
  EditTotal.Text := FloatToStr(total);
end;
procedure TFrmRAB.DATA_URAIANShow(Sender: TObject);
begin
  ComboBox1.Items.Clear;
  FDQueryRab.Open;
  FDQueryRab.First;
  while not FDQueryRab.Eof do
  begin
ComboBox1.Items.AddObject(FDQueryRab.FieldByName('nama_rab')
.AsString,
TObject(FDQueryRab.FieldByName('id_rab').AsInteger));
    FDQueryRab.Next;
  end;
end;
procedure TFrmRAB.ComboBox1Change(Sender: TObject);
begin
  if ComboBox1.ItemIndex = -1 then Exit;
  SelectedID                                                 :=
Integer(ComboBox1.Items.Objects[ComboBox1.ItemIndex]);
  FDQueryRab.Close;
  FDQueryRab.SQL.Text := 'SELECT * FROM tbl_rab WHERE id_rab
= :id';
  FDQueryRab.ParamByName('id').AsInteger := SelectedID;
  FDQueryRab.Open;
  FDQueryUraian.Close;
  FDQueryUraian.SQL.Text :=
```

```
          'WITH DataUraian AS ( ' +
          '    SELECT ' +
          '        NULL AS id_detail, ' +
          '        u.id_uraian, ' +
          '        u.uraian, ' +
          '        NULL AS nama_barang, ' +
          '        NULL AS volume, ' +
          '        NULL AS satuan, ' +
          '        NULL AS harga, ' +
          '        NULL AS total_harga, ' +
          '        NULL AS pajak, ' +
          '        NULL AS total_sebelum_pajak, ' +
          '        NULL AS total_setelah_pajak, ' +
          '        ''Tambah Barang'' AS aksi, ' +
          '        0 AS sort_order ' +
          '    FROM tbl_uraian u ' +
          '    WHERE u.id_rab = :id_rab ' +
          '    UNION ALL ' +
          '    SELECT ' +
          '        d.id_detail, ' +
          '        u.id_uraian, ' +
          '        NULL AS uraian, ' +
          '        b.nama_barang, ' +
          '        d.volume, ' +
          '        b.satuan, ' +
          '        b.harga, ' +
          '        d.total_harga, ' +
          '        b.pajak, ' +
          '        (d.total_harga) AS total_sebelum_pajak, ' +
          '           (d.total_harga + (d.total_harga * b.pajak /
      100)) AS total_setelah_pajak, ' +
          '        ''Hapus Barang'' AS aksi, ' +
          '        1 AS sort_order ' +
          '    FROM tbl_uraian u ' +
          '       LEFT  JOIN  tbl_rab_detail  d  ON  u.id_uraian  =
      d.id_uraian ' +
          '    LEFT JOIN tbl_barang b ON d.id_barang = b.id_barang
      ' +
          '    WHERE u.id_rab = :id_rab ' +
          ') ' +
          'SELECT * FROM DataUraian ' +
          'ORDER BY id_uraian, sort_order;';
        FDQueryUraian.ParamByName('id_rab').DataType := ftInteger;
        FDQueryUraian.ParamByName('id_rab').AsInteger           :=
      SelectedID
        FDQueryUraian.Open;
        if not FDQueryRab.IsEmpty then
          begin
            EditRABUraianKode.Text                              :=
      FDQueryRab.FieldByName('kode').AsString;
            EditRABUraianAnggaran.Text                          :=
      FDQueryRab.FieldByName('anggaran').AsString;
          end
        else
          begin
            EditRABUraianKode.Clear;
```

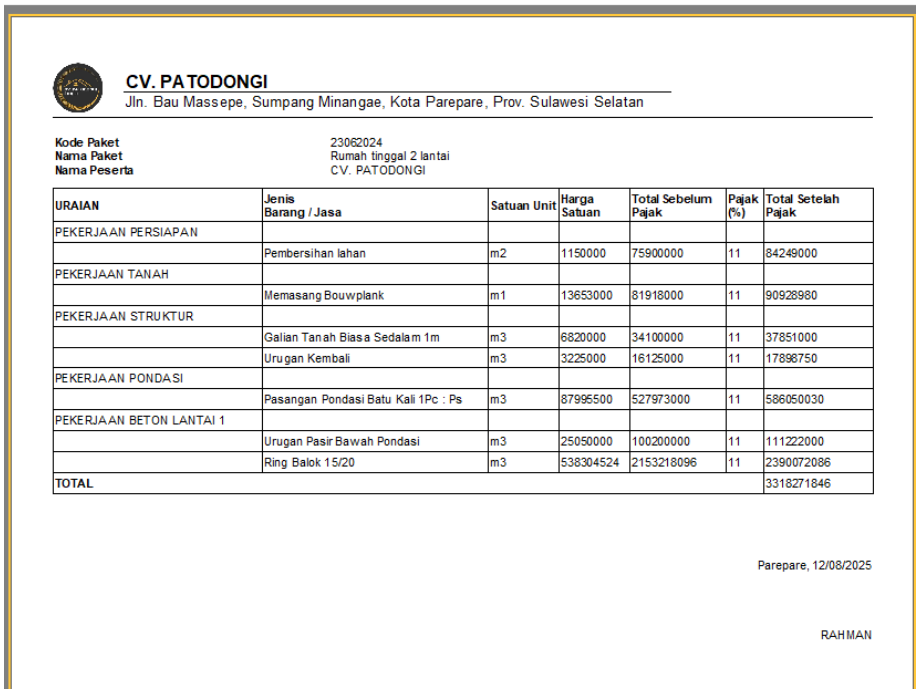
```
          EditRABUraianAnggaran.Clear;
        end;
      HitungTotal;
      EditSisa.Text := IntToStr(
        StrToIntDef(EditRABUraianAnggaran.Text, 0) -
        StrToIntDef(EditTotal.Text, 0)
      );
    end;
    procedure TFrmRAB.btnKelolaUraianClick(Sender: TObject);
    begin
      if SelectedID = 0 then
      begin
        ShowMessage('Silakan pilih RAB terlebih dahulu.');
        Exit;
      end;
      FormKelolaUraian := TFormKelolaUraian.Create(Self);
      try
        FormKelolaUraian.LoadData(SelectedID);
        FormKelolaUraian.ShowModal;
      finally
        FormKelolaUraian.Free;
      end;
    end;
    procedure TFrmRAB.btnRefreshClick(Sender: TObject);
    begin
      FDQueryUraian.Close;
      FDQueryUraian.SQL.Text :=
        'WITH DataUraian AS ( ' +
        '    SELECT ' +
        '        NULL AS id_detail, ' +
        '        u.id_uraian, ' +
        '        u.uraian, ' +
        '        NULL AS nama_barang, ' +
        '        NULL AS volume, ' +
        '        NULL AS satuan, ' +
        '        NULL AS harga, ' +
        '        NULL AS total_harga, ' +
        '        NULL AS pajak, ' +
        '        NULL AS total_sebelum_pajak, ' +
        '        NULL AS total_setelah_pajak, ' +
        '        ''Tambah Barang'' AS aksi, ' +
        '        0 AS sort_order ' +
        '    FROM tbl_uraian u ' +
        '    WHERE u.id_rab = :id_rab ' +
        '    UNION ALL ' +
        '    SELECT ' +
        '        d.id_detail, ' +
        '        u.id_uraian, ' +
        '        NULL AS uraian, ' +
        '        b.nama_barang, ' +
        '        d.volume, ' +
        '        b.satuan, ' +
        '        b.harga, ' +
        '        d.total_harga, ' +
        '        b.pajak, ' +
        '        (d.total_harga) AS total_sebelum_pajak, ' +
```

```
    '          (d.total_harga + (d.total_harga * b.pajak /
100)) AS total_setelah_pajak, ' +
    '        ''Hapus Barang'' AS aksi, ' +
    '        1 AS sort_order ' +
    '    FROM tbl_uraian u ' +
    '       LEFT JOIN tbl_rab_detail d ON u.id_uraian =
d.id_uraian ' +
    '    LEFT JOIN tbl_barang b ON d.id_barang = b.id_barang
' +
    '    WHERE u.id_rab = :id_rab ' +
    ') ' +
    'SELECT * FROM DataUraian ' +
    'ORDER BY id_uraian, sort_order;';
  FDQueryUraian.ParamByName('id_rab').DataType := ftInteger;
  FDQueryUraian.ParamByName('id_rab').AsInteger          :=
SelectedID
  FDQueryUraian.Open;
  HitungTotal;
end;
procedure TFrmRAB.DBGrid1DblClick(Sender: TObject);
var
  idDetail: Integer;
  idUraian: Integer;
  aksi: string;
begin
  if FDQueryUraian.IsEmpty then Exit;
  idUraian                                                :=
FDQueryUraian.FieldByName('id_uraian').AsInteger;
  aksi := FDQueryUraian.FieldByName('aksi').AsString;
  if aksi = 'Tambah Barang' then
  begin
    FormKelolaBarang := TFormKelolaBarang.Create(Self);
    try
      FormKelolaBarang.LoadData(idUraian);
      FormKelolaBarang.ShowModal;
    finally
      FormKelolaUraian.Free;
    end;
  end
  else if aksi = 'Hapus Barang' then
  begin
    if  MessageDlg('Yakin  ingin  menghapus  barang  ini?',
mtConfirmation, [mbYes, mbNo], 0) = mrYes then
    begin
      idDetail                                            :=
FDQueryUraian.FieldByName('id_detail').AsInteger;
      if idDetail > 0 then
      begin
        FDQueryRab.Close;
        FDQueryRab.SQL.Text := 'DELETE FROM tbl_rab_detail
WHERE id_detail = :id_detail';
        FDQueryRab.ParamByName('id_detail').AsInteger     :=
idDetail;
        FDQueryRab.ExecSQL;
        btnRefresh.Click;
        ShowMessage('Barang berhasil dihapus.');
```

```
            end
            else
              ShowMessage('ID Detail tidak ditemukan, data tidak
      dapat dihapus.');
          end;
        end;
      end;
      procedure TFrmRAB.btnCetakClick(Sender: TObject);
      var
        KodeRab, NamaRab, TotalRab, NamaDirektur: string;
      begin
        KodeRab := EditRABUraianKode.Text;
        NamaRab := ComboBox1.Text;
        TotalRab := EditTotal.Text;
        NamaDirektur := EditNamaDirektur.Text;
        frxReport1.Variables['KodeRab'] := QuotedStr(KodeRab); //
      Mengirim KodeRab
        frxReport1.Variables['NamaRab'] := QuotedStr(NamaRab);
        frxReport1.Variables['TotalRab'] := QuotedStr(TotalRab);
        frxReport1.Variables['NamaDirektur']                    :=
      QuotedStr(NamaDirektur);
        frxReport1.ShowReport();
      end;
      procedure  TFrmRAB.ButtonSimpanPengaturanUraianClick(Sender:
      TObject);
      begin
        if Trim(EditNamaDirektur.Text) = '' then
        begin
          ShowMessage('Nama direktur tidak boleh kosong.');
          Exit;
        end;
        FDQueryPengaturan.Close;
        FDQueryPengaturan.SQL.Text :=
          'UPDATE tbl_pengaturan ' +
          'SET nama_direktur = :nama ' +
          'WHERE id_pengaturan = 1';
        FDQueryPengaturan.ParamByName('nama').AsString          :=
      EditNamaDirektur.Text;
        FDQueryPengaturan.ExecSQL;
        ShowMessage('Nama direktur berhasil diperbarui.');
      end;
end.
```

Tampilan ini adalah jantung utama sistem. Di sinilah proyek didefinisikan dengan memasukkan data paket RAB. Ini adalah tempat menyusun perencanaan biaya berdasarkan data yang sudah dimasukkan sebelumnya. Desainnya memperlihatkan bahwa sistem ini tidak hanya menyimpan data, tetapi juga menghitung dan menyusun perencanaan proyek secara utuh dan *form* input menggunakan nama paket dan kode paket.

Dibawah ini adalah cetakan dari laporan RAB CV. PATODONGI:



CV. PATODONGI
Jln. Bau Massepe, Sumpang Minangae, Kota Parepare, Prov. Sulawesi Selatan

Kode Paket 23062024
Nama Paket Rumah tinggal 2 lantai
Nama Peserta CV. PATODONGI

URAIAN	Jenis Barang / Jasa	Satuan Unit	Harga Satuan	Total Sebelum Pajak	Pajak (%)	Total Setelah Pajak
PEKERJAAN PERSIAPAN						
	Pembersihan lahan	m2	1150000	75900000	11	84249000
PEKERJAAN TANAH						
	Memasang Bouwplank	m1	13653000	81918000	11	90928980
PEKERJAAN STRUKTUR						
	Galian Tanah Biasa Sedalam 1m	m3	6820000	34100000	11	37851000
	Urugan Kembali	m3	3225000	16125000	11	17898750
PEKERJAAN PONDASI						
	Pasangan Pondasi Batu Kali 1Pc : Ps	m3	87995500	527973000	11	586050030
PEKERJAAN BETON LANTAI 1						
	Urugan Pasir Bawah Pondasi	m3	25050000	100200000	11	111222000
	Ring Balok 15/20	m3	538304524	2153218096	11	2390072086
TOTAL						3318271846

Parepare, 12/08/2025

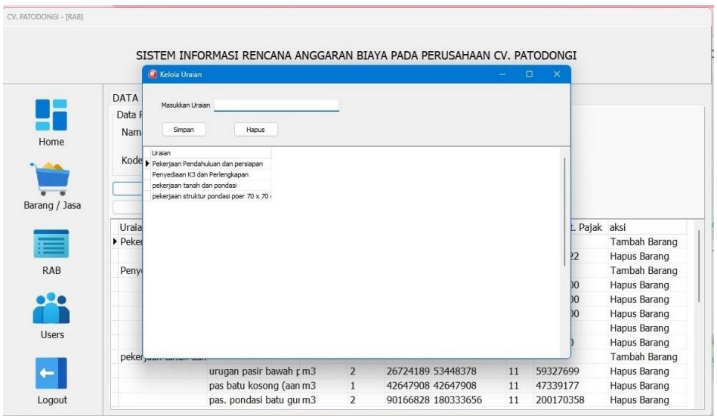
RAHMAN

Gambar 4. 44 Cetakan Laporan Rencana Biaya Anggaran (RAB)

Gambar diatas merupakan output sistem dalam bentuk laporan RAB proyek konstruksi, yang ditampilkan dalam *form*at tabel terstruktur. Tampilan laporan Rencana Anggaran Biaya (RAB) merupakan hasil keluaran dari sistem yang menyajikan in*form*asi lengkap mengenai rincian pekerjaan dalam suatu proyek. Di dalamnya terdapat uraian pekerjaan, jenis barang atau jasa yang digunakan, satuan unit, harga satuan, total biaya sebelum pajak, persentase pajak. Setiap item disusun secara terstruktur mulai dari pekerjaan pendahuluan, penyediaan alat keselamatan kerja (K3), pekerjaan tanah dan pondasi, hingga pekerjaan struktur. Tampilan ini dirancang untuk memberikan kemudahan dalam membaca dan memahami detail anggaran proyek secara menyeluruh. Selain menampilkan data secara jelas dan sistematis, laporan RAB ini juga berfungsi sebagai bagian dari dokumen penawaran yang

menunjukkan rincian biaya pekerjaan secara transparan, dan juga sebagai dokumen pendukung dalam pelaporan dan pertanggungjawaban biaya kepada pihak terkait.

6. Tampilan Kelola Uraian



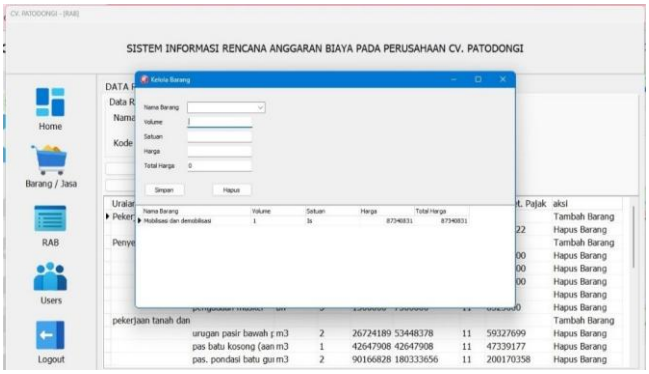
Gambar 4. 45 Kelola Uraian

```
implementation
procedure          TFormKelolaUraian.btnSimpanClick(Sender:
TObject);
begin
  if Trim(EditUraian.Text) = '' then
  begin
    ShowMessage('Uraian tidak boleh kosong!');
    Exit;
  end;
  try
    FDQuery1.Close;
    FDQuery1.SQL.Text := 'INSERT INTO tbl_uraian (id_rab,
uraian) VALUES (:id_rab, :uraian)';
    FDQuery1.ParamByName('id_rab').AsInteger           :=
FSelectedID;
    FDQuery1.ParamByName('uraian').AsString            :=
Trim(EditUraian.Text);
    FDQuery1.ExecSQL;
    ShowMessage('Data berhasil disimpan!');
    FDQuery1.Close;
    FDQuery1.SQL.Text := 'SELECT * FROM tbl_uraian WHERE
id_rab = :id';
    FDQuery1.ParamByName('id').AsInteger := FSelectedID;
    FDQuery1.Open;
  except
    on E: Exception do
      ShowMessage('Error: ' + E.Message);
  end;
```

```
      end;
      procedure          TFormKelolaUraian.btnHapusClick(Sender:
      TObject);
      begin
        if FDQuery1.IsEmpty then
        begin
          ShowMessage('Tidak   ada   data   yang   dipilih   untuk
      dihapus!');
          Exit;
        end;
        if MessageDlg('Apakah Anda yakin ingin menghapus uraian
      ini?', mtConfirmation, [mbYes, mbNo], 0) = mrYes then
        begin
          try
            FDQuery1.Delete;  //  Menghapus  baris  yang  dipilih
      dalam dataset
            ShowMessage('Data berhasil dihapus!');
          except
            on E: Exception do
              ShowMessage('Error   saat   menghapus   data:   '   +
      E.Message);
          end;
        end;
      end;
      procedure          TFormKelolaUraian.LoadData(ASelectedID:
      Integer);
      begin
        FSelectedID := ASelectedID;
        if FSelectedID = 0 then
        begin
          ShowMessage('ID RAB tidak valid!');
          Exit;
        end;
        if not FDConnection1.Connected then
          FDConnection1.Connected := True;
        FDQuery1.Close;
        FDQuery1.SQL.Text  :=  'SELECT  *  FROM  tbl_uraian  WHERE
      id_rab = :id';
        FDQuery1.ParamByName('id').AsInteger := FSelectedID;
        FDQuery1.Open;
      end;
end.
```

Tampilan ini menggambarkan perincian pekerjaan atau kegiatan dalam proyek. Uraian pekerjaan berperan besar dalam menentukan struktur anggaran. Dengan memasukkan uraian desain ini memfasilitasi pengguna untuk membuat daftar pekerjaan yang jelas dan terstruktur, sehingga alur pekerjaan dalam proyek dapat dihitung dan diawasi dengan baik.

7. Tampilan Kelola Barang



Gambar 4. 46 Kelola Barang

```
implementation
procedure TFormKelolaBarang.btnSimpanClick(Sender: TObject);
var
  BarangID, IDUraian: Integer;
  Volume, Harga, Total: Double;
  FS: TFormatSettings;
begin
  FS := TFormatSettings.Create;
  FS.DecimalSeparator := '.';
  if ComboBoxBarang.ItemIndex = -1 then
  begin
    ShowMessage('Silakan pilih barang terlebih dahulu.');
    Exit;
  end;
  BarangID                                                    :=
Integer(ComboBoxBarang.Items.Objects[ComboBoxBarang.ItemInde
x]);
  if  not  TryStrToFloat(EditVolume.Text,  Volume,  FS)  or
(Volume <= 0) then
  begin
    ShowMessage('Masukkan volume yang valid.');
    Exit;
  end;
  if not TryStrToFloat(EditHarga.Text, Harga, FS) or (Harga
<= 0) then
  begin
    ShowMessage('Masukkan harga yang valid.');
    Exit;
  end;
  Total := Volume * Harga;
  IDUraian := StrToIntDef(EditIDUraian.Text, 0);
  FDConnection1.StartTransaction;
  try
    FDQuery1.Close;
    FDQuery1.SQL.Text   :=   'INSERT   INTO   tbl_rab_detail
(id_uraian, id_barang, volume, total_harga) ' +
                         'VALUES   (:id_uraian,   :id_barang,
:volume, :total_harga)';
```

```
        FDQuery1.ParamByName('id_uraian').AsInteger := IDUraian;
        FDQuery1.ParamByName('id_barang').AsInteger := BarangID;
        FDQuery1.ParamByName('volume').AsFloat := Volume;
        FDQuery1.ParamByName('total_harga').AsFloat := Total;
        FDQuery1.ExecSQL;
        FDConnection1.Commit;
        ShowMessage('Data berhasil disimpan.');
        LoadData(IDUraian);
      except
        on E: Exception do
        begin
          FDConnection1.Rollback;
          ShowMessage('Terjadi kesalahan saat menyimpan data: '
+ E.Message);
        end;
      end;
    end;
    procedure TFormKelolaBarang.btnHapusClick(Sender: TObject);
    var
      IDDetail: Integer;
    begin
      if DBGrid1.DataSource.DataSet.IsEmpty then
      begin
        ShowMessage('Tidak ada data yang dipilih.');
        Exit;
      end;
      IDDetail                                                  :=
    DBGrid1.DataSource.DataSet.FieldByName('id_detail').AsIntege
    r;
      if MessageDlg('Apakah Anda yakin ingin menghapus data
    ini?', mtConfirmation, [mbYes, mbNo], 0) = mrYes then
      begin
        FDQuery1.Close;
        FDQuery1.SQL.Text := 'DELETE FROM tbl_rab_detail WHERE
    id_detail = :id_detail';
        FDQuery1.ParamByName('id_detail').AsInteger := IDDetail;
        FDQuery1.ExecSQL;
        ShowMessage('Data berhasil dihapus.');
        LoadData(FSelectedID); // Refresh data
      end;
    end;
    procedure TFormKelolaBarang.HitungTotal;
    var
      Volume, Harga, Total: Double;
      FS: TFormatSettings;
    begin
      FS := TFormatSettings.Create;
      FS.DecimalSeparator := '.';
      if not TryStrToFloat(EditVolume.Text, Volume, FS) then
        Volume := 0;
      if not TryStrToFloat(EditHarga.Text, Harga, FS) then
        Harga := 0;
      Total := Volume * Harga;
      EditTotalHarga.Text := Total.ToString;
    end;
    procedure     TFormKelolaBarang.ComboBoxBarangChange(Sender:
```

```
      TObject);
      var BarangID: Integer;
      begin
        if ComboBoxBarang.ItemIndex = -1 then Exit;
        BarangID                                                  :=
      Integer(ComboBoxBarang.Items.Objects[ComboBoxBarang.ItemInde
      x]);
        FDQueryBarang.Close;
        FDQueryBarang.SQL.Text := 'SELECT * FROM tbl_barang WHERE
      id_barang = :id';
        FDQueryBarang.ParamByName('id').AsInteger := BarangID;
        FDQueryBarang.Open;
        if not FDQueryBarang.IsEmpty then
        begin
          EditSatuan.Text                                         :=
      FDQueryBarang.FieldByName('satuan').AsString;
          EditHarga.Text                                          :=
      FDQueryBarang.FieldByName('harga').AsString;
          HitungTotal;
        end
        else
        begin
          EditSatuan.Clear;
          EditHarga.Clear;
        end;
      end;
      procedure TFormKelolaBarang.EditVolumeKeyUp(Sender: TObject;
      var Key: Word;
        Shift: TShiftState);
      begin
        HitungTotal;
      end;
      procedure TFormKelolaBarang.FormShow(Sender: TObject);
      begin
        ComboBoxBarang.Items.Clear;
        FDQueryBarang.Open;
        FDQueryBarang.First;
        while not FDQueryBarang.Eof do
        begin
      ComboBoxBarang.Items.AddObject(FDQueryBarang.FieldByName('na
      ma_barang').AsString,
      TObject(FDQueryBarang.FieldByName('id_barang').AsInteger));
          FDQueryBarang.Next;
        end;
      end;
      procedure TFormKelolaBarang.LoadData(ASelectedID: Integer);
      begin
        FSelectedID := ASelectedID;
        if FSelectedID = 0 then
        begin
          ShowMessage('ID RAB tidak valid!');
          Exit;
        end;
        if not FDConnection1.Connected then
          FDConnection1.Connected := True;
        FDQuery1.Close;
```

```
        FDQuery1.SQL.Text :=
          'SELECT d.id_detail, b.nama_barang, d.volume, b.satuan,
      b.harga, d.total_harga ' +
          'FROM tbl_rab_detail d ' +
          'LEFT JOIN tbl_barang b ON d.id_barang = b.id_barang ' +
          'WHERE d.id_uraian = :id';
        FDQuery1.ParamByName('id').AsInteger := FSelectedID;
        FDQuery1.Open;
        EditIDUraian.Text := FSelectedID.ToString();
      end;
end.
```

Tampilan ini memperlihatkan bagaimana barang-barang dikelola secara rinci dalam hubungannya dengan uraian pekerjaan. *Form* yang digunakan adalah Nama Barang, Volume, Satuan, Harga, dan Total Harga. Dengan mengelola barang secara langsung dari uraian, sistem ini menciptakan konsistensi dan akurasi dalam pembentukan anggaran proyek

E. Implementasi

Implementasi system merupakan tahap penerapan dari suatu teknologi yang di desain untuk siap di operasikan. Tahap ini merupakan terjemahan perancangan dari bab hasil analisis sebelumnya dalam suatu Bahasa pemrograman dalam membangun sebuah Sistem In*form*asi Rencana Anggaran Biaya Pada Perusahaan CV. PATODONGI.

1. Kebutuhan Perangkat Keras

Spesifikasi minimum perangkat keras sebagai berikut:

Tabel 4. 8 *Kebutuhan Perangkat Keras*

Jenis	Spesifikasi
Laptop/PC	*ACER*
Processor	*AMD Ryzen 3 7320U with Raden Graphics*
Memory	8 GB

2. Kebutuhan Perangkat Lunak.

Spesifikasi minimum perangkat lunak sebagai berikut :

Tabel 4. 9 Kebutuhan Perangkat Lunak

Jenis	Spesifikasi
Sistem Operasi	*Windows 10*
Aplikasi	*Delphi*

F. Pengujian Sistem

1. *BlackBox*

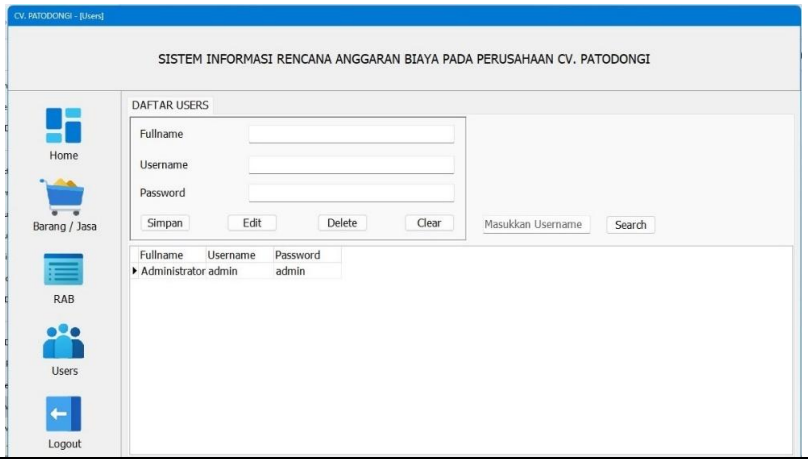
Pengujian *system* dilakukan dengan cara pengujian *BlackBox. Blackox* adalah metode pengujian perangkat lunak yang menguji.

fungsionalitas aplikasi tanpa mengintip ke dalam struktur atau cara kerja internalnya. Metode pengujian ini dapat diterapkan secara virtual ke setiap tingkat pengujian perangkat lunak: unit, integrasi, system, dan penerimaan.

Tabel 4. 10 Hasil *BlackBox test* Menu *Login*

Test Faktor	Hasil	Keterangan
Login	✓	Berhasil menampilkan menu *Login*, Jika halaman *login* sesuai *Username* dan *Password* berhasil dan jika *Username* dan *Password* salah maka *login* gagal dan tidak masuk kehalaman selanjutnya.
Screen Shoot		
		

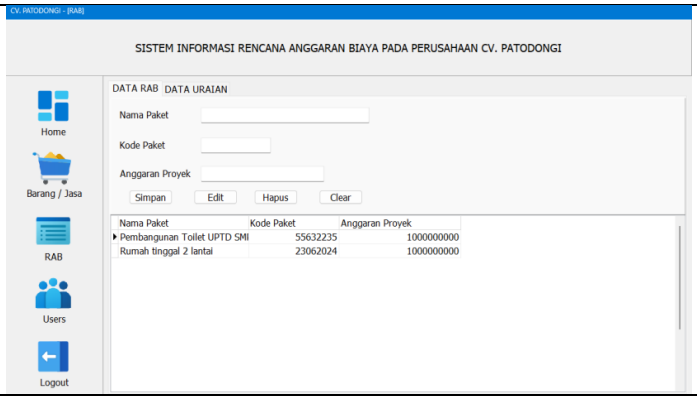
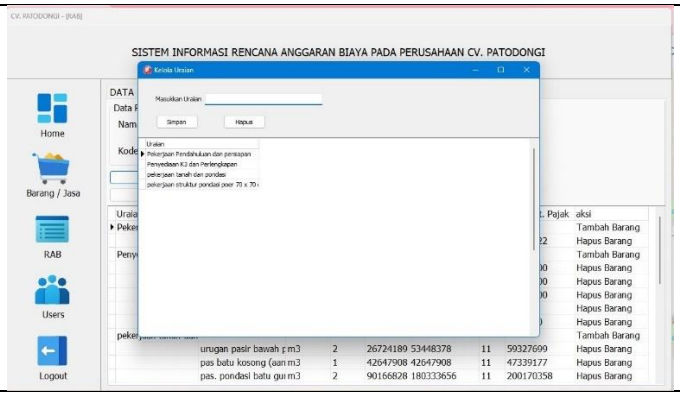
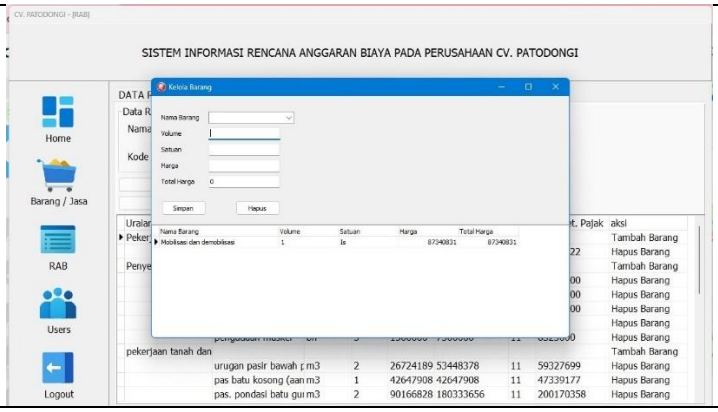
Tabel 4. 11 Hasil BlackBox test menu Daftar User

Test Faktor	Hasil	Keterangan
Menu Daftar *User*	✓	Berhasil menampilkan menu Daftar *User*, Halaman ini berisi *form fullname*, *Username*, *Password*. Jika semua sesuai maka berhasil daftar *User* dan jika tidak sesuai maka gagal melakukan pendaftaran *User*.
Screen Shoot		
		

Tabel 4. 12 Hasil *BlackBox test Home*

Test Faktor	Hasil	Keterangan
Menu *Home*	✓	Berhasil menampilkan menu *Home* berupa penjelesan singkat dari perusahaan.
Screen Shoot		
		

Tabel 4. 13 Hasil *BlackBox test* menu Jasa Barang

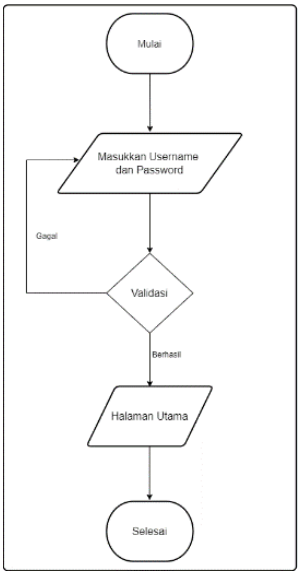
Test Faktor	Hasil	Keterangan
Menu Jasa Barang	✓	Berhasil menampilkan menu Jasa Barang, Halaman ini berisi nama paket dan kode paket serta anggaran proyek. Jika data sesuai maka akan menghasilkan halaman Data Uraian dan Data Barang sebagai hasil Rencana Anggaran Biaya.
Screen Shoot		
		
Screen Shoot		
		
Sreen Shoot		
		

2. *WhiteBox*

Pengujian ini dilakukan dengan cara pngujian *WhiteBox*. *WhiteBox* adalah metode pengujian perangkat lunak yang menguji struktur internal atau cara kerja aplikasi yang bertentangan dengan fungsinya. Dalam pengujian kotak putih, perspektif internal sistem digunakan untuk merancang kasus uji.

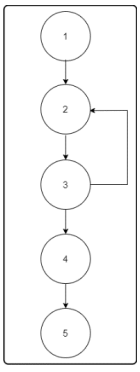
a. *White Box login* admin

1) *Flowchart*



Gambar 4. 47 *Flowchart Login*

2) *Flowgraph*



Gambar 4. 48 *Flowgraph Login*

Berdasarakan gambar di atas dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *Cyclomatic Complexcity* dari *flowgraph* di atas memiliki *Region* = 2

(3) *Independent path* pada *flowgraph* tersebut yakni:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

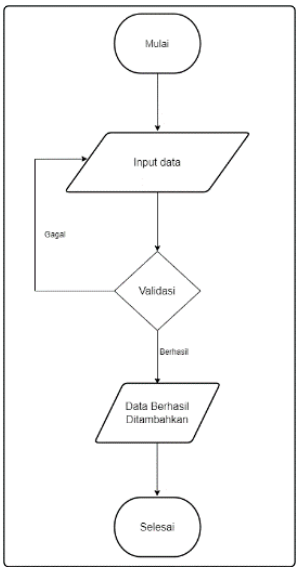
(4) Grafik matriks *login*

Tabel 4. 14 *Matriks Login*

	1	**2**	**3**	**4**	**5**	**E-1**
1		1				1 – 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

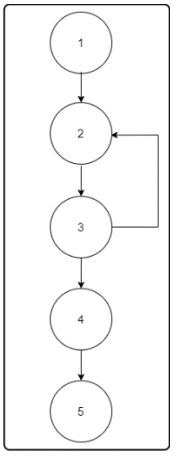
b. *White Box tambah user admin*

1) *Flowchart*



Gambar 4. 49 *Flowchart* tambah *user admin*

2) *Flowgraph*



Gambar 4. 50 *Flowgraph* tambah *user admin*

Berdasarkan gambar diatas dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *Cyclomatic Complexcity* dari *flowgraph* di atas memiliki *Region* = 2

(3) *Independent path* pada *flowgraph* tersebut yakni:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

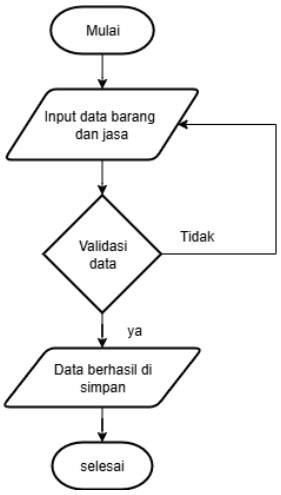
(4) Grafik matriks tambah user admin

Tabel 4. 15 *Grafik matriks* tambah *user admin*

	1	**2**	**3**	**4**	**5**	**E-1**
1		1				1 – 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

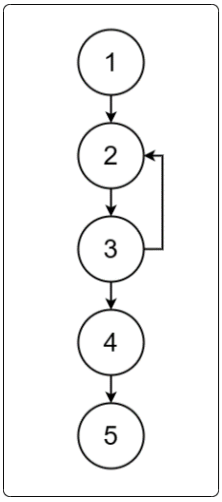
c. *White Box* barang dan jasa

1) *Flowchart*



Gambar 4. 51 *Flowchart* barang dan jasa

2) *Flowgruph*



Gambar 4. 52 *Flowgruph* barang dan jasa

Berdasarkan gambar 4. 59 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexity* V(G) pada *edge* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexity* dari *Flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *Flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

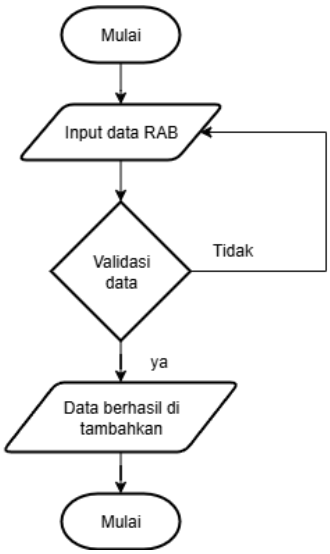
(4) Grafik matriks barang dan jasa

Tabel 4. 16 *Grafik matriks* barang dan jasa

	1	**2**	**3**	**4**	**5**	**E-1**
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1+ 1 = 2

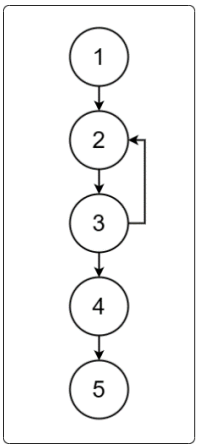
d. *White Box* rencana barang dan biaya

1) *Flowchart*



Gambar 4. 53 *Flowchart* RAB

2) *Flowgraph*



Gambar 4. 54 *Flowgruph* RAB

Berdasarkan gambar 4. 61 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexity* V(G) pada *edge* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexity* dari *Flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *Flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

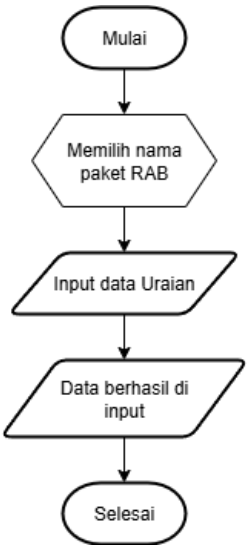
(4) Grafik matriks RAB

Tabel 4. 17 *Grafik matriks* RAB

	1	**2**	**3**	**4**	**5**	**E-1**
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1+ 1 = 2

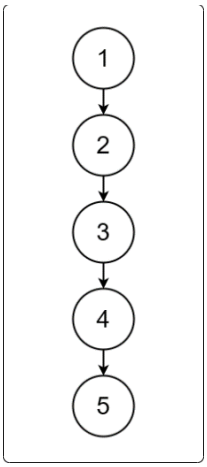
e. *White Box* Kelola Uraian

1) *Flowchart*



Gambar 4. 55 *Flowchart* kelola uraian

2) *Flowgraph*



Gambar 4. 56 *Flowgraph* kelola uraian

Berdasarkan gambar 4. 63 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexity* V(G) pada *edge* dan *node.*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *cyclomatic complexity* dari *Flowgraph* diatas memiliki *region = 1*

(3) *Independent path* pada *Flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 4 – 5

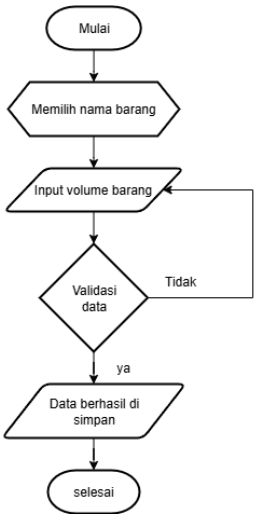
(4) Grafik matriks halaman kelola uraian

Tabel 4. 18 *Grafik matriks* kelola uraian

	1	**2**	**3**	**4**	**5**	**E-1**
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3				1		1 – 1 = 0
4					1	1 – 1 = 0
5						
	SUM (E + 1)					0+ 1 = 1

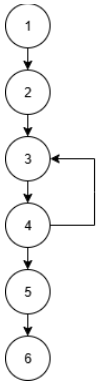
f. *White Box* Kelola Barang

1) *Flowchart*



Gambar 4. 57 *Flowchart* kelola barang

2) *Flowgraph*



Gambar 4. 58 *Flowgrph* kelola barang

Berdasarkan gambar 4. 65 diatas, dilakukan perhitungan sebagai berikut:

(5) Menghitung *cyclomatic complexity* V(G) pada *edge* dan *node.*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 6

N (*node*) = 6

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 6 – 6 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(6) Berdasarkan perhitungan *cyclomatic complexity* dari *Flowgraph* diatas memiliki *region = 2*

(7) *Independent path* pada *Flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 4 – 3

Path 2 = 1– 2 – 3 – 4 – 5 – 6

(8) Grafik matriks halaman kelola barang

Tabel 4. 19 *Grafik matriks* kelola barang

	1	**2**	**3**	**4**	**5**	**6**	**E-1**
1		1					1 - 1 = 0
2			1				1 – 1 = 0
3				1			1 – 1 = 0
4			1		1		2 – 1 = 1
5						1	1 – 1 = 0
6							
	SUM (E + 1)						1+ 1 = 2

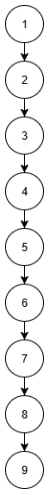
g. *White Box* Cetak Laporan

1) *Flowchart*



Gambar 4. 59 *Flowchart* cetak laporan

2) *Flowgraph*



Gambar 4. 60 *Flowgraph* cetak laporan

Berdasarkan gambar 4. 67 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexity* V(G) pada *edge* dan *node.*

Pada rumus : V(G) = E – N + 2
E (*edge*) = 8
N (*node*) = 9
P (Predikat *node*) = 0
Penyelesaian :
V(G) = E – N + 2
= 8 – 9 + 2
= 1
Predikat (P) = P + 1
= 0 + 1
= 1

(2) Berdasarkan perhitungan *cyclomatic complexity* dari *Flowgraph* diatas memiliki *region = 1*

(3) *Independent path* pada *Flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9

(4) Grafik matriks halaman kelola uraian

Tabel 4. 20 *Grafik matriks* cetak laporan

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**E-1**
1		1								1 - 1 = 0
2			1							1 – 1 = 0
3				1						1 – 1 = 0
4					1					1 – 1 = 0
5						1				1– 1 = 0
6							1			1– 1 = 0
7								1		1– 1 = 0
8									1	1– 1 = 0
9										
	SUM (E + 1)									0 + 1 = 1

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa sistem in*form*asi Rencana Anggaran Biaya (RAB) berbasis desktop yang dikembangkan untuk CV. Patodongi mampu meningkatkan efisiensi dalam proses perencanaan anggaran biaya proyek konstruksi. Sebelumnya, perhitungan dan penyusunan RAB dilakukan secara manual menggunakan *Microsoft Excel*, yang sering kali menyebabkan kesalahan perhitungan, memakan waktu lama, serta kurang efektif dalam pengelolaan data.

Aplikasi ini dilengkapi dengan sejumlah fitur penting yang mendukung pengelolaan RAB. Fitur *login* dan manajemen user memastikan keamanan data dengan membatasi akses hanya kepada pihak yang berwenang. Fitur input data proyek memungkinkan pencatatan detail proyek seperti nama lokasi, serta anggaran proyek. Selain itu, terdapat fitur input data material dan jasa, yang memudahkan pencatatan kebutuhan material, barang dan jasa lengkap dengan harga satuan. Sistem kemudian melakukan penghitungan otomatis RAB sehingga proses perhitungan lebih cepat dan minim kesalahan.

Fitur lain yang menjadi nilai tambah adalah total dana dan sisa dana, yang memberikan in*form*asi penting dalam mengendalikan anggaran agar tidak melampaui batas biaya yang tersedia. Fitur laporan RAB menyajikan rincian biaya proyek secara detail dan transparan, sehingga tidak hanya bermanfaat

sebagai alat pengendalian internal, tetapi juga dapat digunakan sebagai dokumen pendukung dalam proses pengajuan tender. Selain itu, terdapat fitur pengaturan (konfigurasi sistem) yang bersifat fleksibel, sehingga perusahaan dapat menyesuaikan parameter sesuai kebutuhan operasional, mulai dari kategori pekerjaan hingga pengaturan hak akses pengguna.

B. Saran

Pada penelitian ini penulis menyadari bahwa masih ada bebrapa kekurangan yang sangat diperlukan perbaikan dan pengembangan di penelitian selanjutnya. Oleh karena itu, penulis memiliki bebrapa saran untuk pengembangan selanjutnya, sebagai berikut:

1. Untuk meningkatkan fungsionalitas sistem, disarankan agar sistem in*form*asi Rencana Anggaran Biaya (RAB) dikembangkan lebih lanjut dengan fitur tambahan seperti integrasi dengan sistem manajemen proyek, analisis prediksi biaya, serta laporan keuangan yang lebih mendetail.
2. Mengingat sistem ini mengelola data anggaran perusahaan, perlu diterapkan sistem keamanan yang lebih ketat, seperti enkripsi data, otorisasi pengguna berbasis peran (role-based access control), serta fitur backup otomatis untuk mencegah kehilangan data.

DAFTAR PUSTAKA

Andharsaputri, R. L. (2021). Rancang Bangun Sistem In*form*asi Pengadaan Barang Dan Jasa Berbasis Dekstop. *Jurnal Ilmiah Teknologi Informasi Asia*, *15*(1), 1-12.

Asmara , D. R., Munawaroh, S., & Permatasari , H. (2023). Pengujian Interface Tambah Barang Pada Aplikasi Point Of Sales Berbasis Web Menggunakan Metode White Box Testing dan Black Box Testing. *Prosiding Seminar Nasional Teknologi Informasi dan Bisnis*

Dini, R. *Jurnal Sains dan Informatika.*

Eska, J., Hidayatullah, H., & Sari, A. N. (2023). Pelatihan Aplikasi Rencana Anggaran Biaya Proyek Pada PT. Archivelo Group. *Jurnal Pemberdayaan Sosial dan Teknologi Masyarakat, 3(1), 1-6.*

Ginanjar, T. B. (2021). Analisa Dan Perancangan Sistem In*form*asi Akuntansi Pada GO! Adventure Yogyakarta Berbasis Dekstop. In*form*ation System Journal, 4(1), 25-31.

Hendri, H., Saputra, R. N., & Kisbianty, D. (2019). Perancangan Aplikasi Rencana Anggaran Biaya Pada PT. Zaki Putra Andalas Jambi. *Jurnal Ilmiah Media Sisfo, 13(2), 115-126.*

Hermawan, D. (2020). Rancang Bangun Sistem In*form*asi Renvana Anggaran Biaya (RAB) Rumah Minimalis Berbasis Android Di Kabupaten (Doctoral dissertation, UNISNU Jepara).

Jismar (Journal of In*form*ation System, Applied, Management, Accounting and Researh). (2019). *journal.stmikjayakarta.ac.id/index.php/jisamar.*

Lumbantobing, J., Perwito, P., & Aditiarno, R. (2023). Perancangan Sistem In*form*asi Akuntansi RAB Pemeliharaan Peralatan Dengan Microsoft Visual Studio 2010 Dan Database MySQL: Studi Deskriptif Pada Dinas ATR/BPN Kota Bandung. Coopetition: *Jurnal Ilmiah Manajemen*, 14(2), 221-234.

Meliana, Y., & Hwihanus, H. (2023). Analisis Peranan dan Kegunaan Sistem In*form*asi Akuntansi Pada PT Kurnia Eka Nusa Yang Berbasis Accurate-Dekstop. *Inisiatif: Jurnal Ekonomi, Akuntansi dan Manajemen*, *2*(1), 362-374.

Perdana, I., Akbar, M., Wulandari, R., Yuda, P., Yermadona, H., Bastian, E., Program, M., Sipil, S. T., Teknik, F., Sumatera Barat, U. M., & Program, D. (2023). Perancangan Sistem In*form*asi Rencana Anggaran Biaya (RAB) Rumah Sederhana Berbasis Web. Ensiklopedia Research and Community Service Review, 2(2). *http://jurnal.ensiklopediaku.org*

Perencanaan Bangunan Berbasis *Web* Menggunakan Metode Framework *Code*igniter, B., Triayudi, A., Minggu, P., Jakarta Selatan, K., & Khusus Ibukota Jakarta, D. (n.d.). Sistem In*form*asi Perhitung Rencana Anggaran.

Pramuditho, A. A. (2022). Sistem In*form*asi Manajemen Anggaran Biaya Perumahan PT Duva Ajeng Sentosa Berbasis *Website.*

Pratama, A. D., & Noviana, W. (2023). Perancangan Sistem In*form*as Aplikasi E-Booking Berbasis Website Menggunakan Metode Extreme Programming Pada PT Indonesia Comnets Plus. LOGIC: *Jurnal Ilmu Komputer dan Pendidikan,* 1(6), 1532-1535

Putri, M. P., & Bobby, B. (2020). Sistem In*form*asi Manajemen Proyek PT. Samudera Perkasa Konstruksi Berbasis *Web*. Matrik : *Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*

Ramadiyanti, S., & Purwani, F. (2021). Sistem In*form*asi Rencana Anggaran Biaya Berbasis Web Pada Dinas Perhubungan Kota Palembang. In Prosiding Seminar Nasional Sains dan Teknologi Terapan (Vol. 4, Issue 1).

Suarpur, N. K. A. S. A., & Paramitha, A. I. I. (2021). Perancangan Sistem In*form*asi Pengolaan Rencana Anggaran Biaya (RAB) Pada Perusahaan Konstruksi Berbasis Web (Study Kasus Cv. Osa Karya Mandiri). *Smart Techno (Smart Technology, Informatics and Technopreneurship)*, *3*(2), 37-45.

Sugiyono. 2019, Metode Penelitian Kuantitatif dan Kualitatif. CV.Alfabeta: Bandung

Syawaldi, N., & Siswanto, E. H. (2019). Rencana Anggaran Biaya.

Utama, D. W., Putra, I. G. L. A. R., & Satwika, I. P. (2019). Rancang bangun sistem in*form*asi pemesanan custom furniture berbasis website. J-SAKTI (*Jurnal Sains Komputer dan Informatika*), 3(1), 53-67.

Yana, S., Gunawan, R. D., & Budiman, A. (2020). Sistem In*form*asi Pelayanan Distribusi Keuangan Desa Untuk Pembangunan (Study Kasus: Dusun Srikaya). *Jurnal Informatika Dan Rekayasa Perangkat Lunak*, 1(2), 254-263.

Yanuardi, Y., & Permana, A. A. (2019). Rancang Bangun Sistem In*form*asi Keuangan Pada Pt. Secret Discoveries Travel and Leisure Berbasis Web. JIKA (*Jurnal Informatika*), 2(2).