TUGAS UJIAN AKHIR SEMENTER
SISTEM INFORMASI AKUNTANSI
APLIKASI SISTEM PEMBELIAN SECARA KREDIT

Dosen pengampu:

Dr. Totok Dewayanto, S.E.,M.Si., Akt.

Disusun oleh:

Ivana Carmella Nugroho (12030124130089)

PRODI AKUNTANSI

FAKULTAS EKONOMIKA DAN BISNIS

UNIVERSITAS DIPONEGORO

2025

Analisis Arsitektur, Konfigurasi, dan Alur Kerja Sistem Informasi Pembelian Kredit

1.0 Pendahuluan: Memahami Konsep Dasar Sistem

Sebagai Analis Sistem dan Arsitek Perangkat Lunak, kita memahami bahwa setiap kegiatan dalam perusahaan, mulai dari yang paling sederhana hingga kompleks seperti proses pembelian secara kredit, memerlukan sistem untuk mengaturnya. Dalam analisis ini, kita akan membedah arsitektur, konfigurasi, dan alur kerja yang mendasari sebuah sistem informasi, dengan mengambil contoh konkret dari proses pembelian kredit. Tanpa sistem yang terstruktur, proses ini akan menjadi tidak efisien dan sulit dikendalikan. Secara fundamental, sebuah sistem adalah kumpulan dari berbagai elemen yang bekerja sama untuk mencapai suatu tujuan.

Beberapa ahli mendefinisikan sistem sebagai berikut:

"Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu." — (Agustiani & Junaidi, 2024)

"Sistem dapat didefinisikan sebagai suatu kumpulan elemen atau komponen yang saling berinteraksi dan bekerja sama untuk mencapai tujuan tertentu, di mana elemen-elemen tersebut dapat berupa manusia, mesin, prosedur, atau perangkat keras dan perangkat lunak yang terintegrasi dalam suatu struktur yang terorganisir." — (Maydianto & Ridho, 2021)

"Sistem merupakan kumpulan komponen yang memiliki hubungan serta bekerja bersama sama dalam mencapai suatu tujuan yang ditentukan." — (Agustina, 2024)

Laporan ini akan menguraikan konsep dasar yang membentuk sebuah sistem, mulai dari anatomi dasarnya, arsitektur perancangan, alur kerja pengembangan (*System Development Life Cycle*), hingga proses *deployment* di lingkungan produksi. Semua konsep ini akan dijelaskan dalam konteks Sistem Informasi Akuntansi Pembelian secara kredit untuk memberikan gambaran yang lebih praktis.

2.0 Anatomi Sebuah Sistem: Input, Proses, dan Output

Setiap sistem, baik manual maupun terkomputerisasi, dirancang untuk memproses masukan (*input*) menjadi keluaran (*output*) melalui serangkaian mekanisme tertentu. Ketiga komponen ini adalah fondasi dari semua sistem informasi.

Input

Input merupakan data mentah atau instruksi yang menjadi pemicu alur kerja dalam sistem. Komponen ini adalah bahan baku yang akan diolah untuk menghasilkan informasi bernilai.

- **Contoh Konkret:** Sebuah departemen mengajukan **informasi permintaan pembelian barang** kepada bagian pengadaan. Dokumen ini, beserta dokumen pendukung lainnya, menjadi input bagi sistem akuntansi pembelian.

Proses

Di sinilah logika bisnis dieksekusi untuk mentransformasikan data mentah menjadi informasi yang bernilai. Sistem menjalankan serangkaian aturan, kalkulasi, dan validasi untuk mencapai tujuan yang telah ditentukan.

- **Contoh Konkret:** Sistem **memverifikasi permintaan pembelian**, memeriksa ketersediaan persediaan barang di gudang, dan kemudian **membuat dokumen pencatatan** yang sistematis untuk diserahkan kepada pemasok.

Output

Output adalah hasil akhir yang terstruktur dan dapat ditindaklanjuti, yang dihasilkan oleh sistem setelah melalui tahap pemrosesan. Output ini harus selaras dengan tujuan bisnis yang mendasari pembuatan sistem.

- **Contoh Konkret:** Sistem menghasilkan **dokumen pesanan pembelian** yang terstruktur untuk dikirim ke pemasok dan **laporan persediaan yang diperbarui** untuk manajemen internal.

3.0 Arsitektur dan Perancangan Sistem

Setelah kebutuhan dianalisis, tahap perancangan bertujuan untuk merepresentasikan hasil analisis tersebut ke dalam model yang lebih teknis. Proses ini memanfaatkan berbagai alat bantu seperti *Entity Relationship Diagram* (ERD) untuk merancang basis data dan *Unified Modeling Language* (UML) untuk memvisualisasikan alur kerja sistem.

3.1 Analisis Kebutuhan Sistem

Tahap awal dalam pengembangan perangkat lunak adalah analisis kebutuhan, di mana tim mengidentifikasi fungsi, fitur, dan karakteristik yang diperlukan oleh pengguna dan sistem itu sendiri. Kebutuhan ini dibagi menjadi dua kategori utama:

- **Kebutuhan Fungsional** Ini adalah fitur-fitur spesifik yang *harus dapat dilakukan* oleh sistem untuk memenuhi tujuan bisnisnya.
    - Pengguna dapat mengajukan permintaan pembelian barang baru.
    - Admin dapat mengelola data pemasok, termasuk menambah, mengubah, dan menghapus informasi.
    - Sistem dapat menghasilkan laporan pembelian bulanan secara otomatis.
- **Kebutuhan Non-Fungsional** Ini adalah karakteristik atau kualitas yang *harus dimiliki* oleh sistem, yang menentukan seberapa baik sistem menjalankan fungsinya.
    - Sistem harus memiliki waktu respons yang cepat, misalnya di bawah 2 detik per permintaan.
    - Sistem harus aman dari serangan siber umum seperti *SQL Injection* untuk melindungi data perusahaan.

3.2 Perancangan Basis Data dengan ERD

Entity Relationship Diagram (ERD) adalah representasi grafis yang digunakan untuk merancang struktur logis dari sebuah basis data. ERD membantu memvisualisasikan entitas (objek data), atribut-atributnya, dan hubungan antar entitas tersebut.

Berikut adalah komponen utama dalam sebuah ERD:

Komponen ERD	Deskripsi
Entitas (Entity)	Objek atau konsep di dunia nyata yang datanya dapat disimpan, seperti "Pemasok" atau "Barang". Digambarkan dengan persegi panjang.
Atribut	Karakteristik atau informasi yang melekat pada suatu entitas, seperti "Nama Pemasok" atau "Harga Barang".
Primary Key (PK)	Atribut unik yang berfungsi sebagai pengidentifikasi eksklusif untuk setiap baris data dalam sebuah entitas.
Relasi (Relationship)	Keterkaitan logis antara dua atau lebih entitas. Contohnya, satu "Pemasok" dapat menyediakan "banyak" "Barang".

3.3 Perancangan Alur Kerja dengan UML

Unified Modeling Language (UML) adalah bahasa pemodelan visual standar yang digunakan untuk merancang, mendokumentasikan, dan memvisualisasikan berbagai aspek dari sistem perangkat lunak. Untuk merancang alur kerja, dua diagram berikut sangat penting:

- **Use Case Diagram** Diagram ini menggambarkan skenario interaksi antara pengguna (disebut *aktor*) dengan sistem. Ini menunjukkan fungsi apa saja yang dapat dilakukan oleh setiap jenis pengguna. Misalnya, seorang "Staf Gudang" dapat mengajukan permintaan barang, sementara seorang "Manajer Keuangan" dapat menyetujui permintaan tersebut.
- **Activity Diagram** Diagram ini memodelkan urutan tindakan atau alur kerja (*workflow*) dalam sebuah proses bisnis. Ini sangat berguna untuk memvisualisasikan langkah-langkah yang terjadi dari awal hingga akhir, termasuk percabangan kondisi dan keputusan.

4.0 Alur Kerja Pengembangan Perangkat Lunak (SDLC)

System Development Life Cycle (SDLC) adalah sebuah kerangka kerja atau proses terstruktur yang digunakan untuk mengembangkan sistem informasi secara berurutan. Salah satu model SDLC yang paling fundamental adalah model **Waterfall**, di mana setiap tahapan diselesaikan secara penuh sebelum melanjutkan ke tahap berikutnya.

Berikut adalah 5 tahapan utama dalam model Waterfall:

1. **Analisis Kebutuhan** Pada tahap ini, tim pengembang mengumpulkan dan menganalisis semua kebutuhan dari pengguna dan pemangku kepentingan untuk memahami fungsi dan fitur apa yang diperlukan oleh sistem.
2. **Perancangan Sistem** Berdasarkan hasil analisis, arsitek sistem merancang cetak biru teknis. Ini termasuk merancang arsitektur perangkat lunak, struktur basis data (menggunakan ERD), dan alur kerja sistem (menggunakan UML).
3. **Pengembangan (Implementasi)** Ini adalah tahap di mana para *developer* mulai menulis kode program untuk membangun sistem sesuai dengan spesifikasi desain yang telah dibuat.

4. **Pengujian (Testing)** Setelah sistem selesai dibangun, tim *quality assurance* (QA) melakukan serangkaian pengujian untuk memastikan semua fitur berjalan sesuai spesifikasi, bebas dari *bug* atau kesalahan, dan memenuhi kebutuhan non-fungsional.
5. **Deployment** Setelah lolos pengujian, sistem dipasang (*install*) di lingkungan produksi (server) sehingga dapat diakses dan digunakan oleh pengguna akhir.

5.0 Konfigurasi Deployment: Mengenal Vercel

Deployment adalah proses membuat aplikasi atau sistem perangkat lunak tersedia untuk digunakan. Ini melibatkan pemasangan aplikasi di server, konfigurasi basis data, dan pengaturan jaringan agar dapat diakses melalui internet. Platform modern telah menyederhanakan proses ini secara signifikan.

Apa Itu Vercel?

Vercel adalah platform *cloud* yang dirancang untuk mempermudah proses *deployment* aplikasi web modern, terutama yang dibangun dengan *framework* seperti Next.js, React, atau Vue.js. Dengan Vercel, pengembang dapat meng-online-kan situs web dalam hitungan detik tanpa perlu repot mengatur server secara manual.

Berikut adalah beberapa keuntungan utama menggunakan Vercel:

- **Proses Cepat:** Memungkinkan *deployment* yang sangat cepat, sering kali hanya dalam hitungan detik setelah kode diunggah ke *repository*.
- **Solusi All-in-One:** Menyediakan berbagai fitur terintegrasi yang diperlukan untuk membangun dan mengelola aplikasi modern, mulai dari *hosting* hingga *analytics*.
- **Meningkatkan Profesionalisme:** Memudahkan penggunaan *custom domain* (misalnya, namaperusahaan.com), yang membantu memperkuat citra merek dan kredibilitas di mata publik.

Saat ini, proses *deployment* sering kali diotomatisasi menggunakan praktik **Continuous Integration/Continuous Deployment (CI/CD)**. Praktik CI/CD ini mengotomatisasi tahap **Pengujian (Testing)** dan **Deployment** dari model Waterfall, memungkinkan rilis yang lebih cepat dan andal dibandingkan pendekatan manual. Dengan alur kerja CI/CD, setiap perubahan kode yang diunggah ke *repository* secara otomatis akan dibangun, diuji, dan di-*deploy* ke server, sehingga mempercepat siklus rilis dan mengurangi risiko kesalahan manusia.

6.0 Studi Kasus: Analisis Sistem Pembelian Kredit pada Koperasi

Sebuah analisis pada **Koperasi Jasa Gunung Madu** meneliti sistem pengendalian internal mereka untuk proses pembelian kredit persediaan barang dagang. Hasil analisis menunjukkan bahwa sistem yang diterapkan sudah cukup sehat dan efisien. Hal ini ditandai dengan adanya dokumen-dokumen dan proses pencatatan yang cukup sistematis, yang merupakan indikator dari arsitektur sistem yang baik.

Namun, studi tersebut menemukan satu kelemahan krusial dalam pengendalian internal mereka:

Adanya bagian penerimaan yang merangkap tugas sebagai bagian penyimpanan barang.

Mengapa ini penting? Pemisahan fungsi adalah salah satu elemen kunci dalam sistem pengendalian internal yang efektif. Ketika satu orang atau departemen memiliki kendali atas beberapa tahap kritis dalam sebuah proses (dalam kasus ini, menerima *dan* menyimpan barang), hal itu membuka peluang untuk:

- **Kesalahan yang tidak terdeteksi:** Tidak ada pihak kedua yang memverifikasi jumlah atau kondisi barang yang diterima.
- **Potensi penyalahgunaan atau kecurangan:** Seseorang dapat dengan mudah memanipulasi catatan inventaris untuk keuntungan pribadi.

Kasus ini menunjukkan bahwa bahkan sistem yang secara teknis efisien masih memerlukan evaluasi berkelanjutan dari sisi pengendalian dan organisasi untuk memastikan integritas dan keamanannya.

7.0 Kesimpulan

Perjalanan membangun sebuah sistem informasi, seperti untuk mengelola pembelian kredit, adalah proses yang terstruktur dan berlapis. Dari ide awal hingga menjadi aplikasi yang fungsional, setiap tahap memiliki peran krusial dalam menentukan keberhasilan sistem tersebut.

Berikut adalah tiga poin kesimpulan utama:

1. Sebuah sistem, seperti untuk pembelian kredit, adalah kumpulan komponen terstruktur (**input**, **proses**, **output**) yang bekerja sama untuk mencapai tujuan bisnis. Memahami anatomi ini adalah langkah pertama untuk merancang solusi yang efektif.
2. Pengembangan sistem yang baik mengikuti alur kerja terstruktur (**SDLC**) yang mencakup tahapan analisis, perancangan arsitektur (menggunakan alat seperti **ERD** dan **UML**), implementasi kode, dan pengujian yang teliti untuk memastikan kualitas.
3. Platform modern seperti **Vercel** menyederhanakan proses *deployment*, memungkinkan sistem untuk diakses oleh pengguna dengan cepat dan efisien. Namun, setelah sistem berjalan, analisis berkelanjutan terhadap **pengendalian internal**, seperti pemisahan fungsi, tetap krusial untuk menjaga keamanan dan integritas operasional.

Link Apps: https://siakredit-a4cu.vercel.app/

Link Github: https://github.com/ivanacarmellaic-bot/siakredit