BAB III. PERMASALAHAN PADA PENGADILAN TINGGI TANJUNG KARANG

3.1 Permasalahan yang Dihadapi Pengadilan Tinggi Tanjung Karang

Dari latar belakangan yang telah disampaikan sebelumnya telah dijelaskan bahwa pada pengadilan tinggi tanjung karang, sistem informasi mengenai surat perintah perjalanan dinas masih dilakukan secara manual, mulai dari pembuatan, pengarsipan, hingga proses pertanggungjawaban biaya perjalanan dinas. proses-proses tersebut sering kali memakan waktu yang tidak sebentar dan karena proses tersebut dilakukan dengan cara yang konvesional maka proses tersebut akan rentan terhadap kesalahan administratif. proses tersebut dilakukan oleh bagian keuangan melalui proses input data dan penyimpanan arsip fisik

3.1.1. Temuan Masalah

Dari Analisa permasalahan sebelumnya maka dapat di ketahui permasalahan yang terjadi pada pengadilan tinggi tanjung karang terkait dengan Surat Perintah Perjalanan Dinas adalah mekanisme dokumentasi dan pengelolaan surat perintah perjalanan dinas

3.1.2. Perumusan Masalah

Dari temuan masalah yang telah disampaikan dapat di lihat bahwa proses dokumentasi dan pengelolaan pada dasarnya menjadi permasalahan utama

3.1.3. Kerangka Pemecahan Masalah

Untuk dapat menyelesaikan permasalahan terkait porses dokumentasi dan pengelolaan Surat Perintah Perjalanan Dinas perlu dilakukan analisis kebutuhan dimulai dari kebutuhan SDM dan teknologi yang akan di gunakan selanjutnya dilakukan perancangan sistem dengan pendekatan objek oriented dan terakhir melakukan implementasi rancangan tesebut menjadi sistem yang dapat di akses multiplatform

3.2. Landasan Teori

3.2.1. Pengertian Framework

Sebuah kerangka kerja atau yang umum bisasa di sebut (Framework) merupakan sekumpulan kode yang terstruktur dengan tujuan untuk memfasilitasi pembangunan aplikasi. *Framework* ini dapat memberikan struktur dan panduan dalam membangun aplikasi, sehingga proses pengembangan aplikasi menjadi lebih efisien. CodeIgniter adalah sebuah framework open source untuk pengembangan aplikasi web menggunakan bahasa pemrograman PHP. Framework ini menerapkan arsitektur Model-View-Controller yang memisahkan logika bisnis, tampilan, dan kontrol aplikasi menjadi komponen-komponen yang terpisah. Dengan menggunakan CodeIgniter, pengembangan aplikasi web menjadi lebih efisien dan terstruktur.

Berikut adalah beberapa karakteristik dan manfaat utama dari penggunaan framework dalam PHP:

1. **Struktur yang Terorganisir:** Framework menyediakan struktur yang jelas dan terorganisir untuk pengembangan aplikasi. Ini mencakup pembagian antara logika bisnis, tampilan, dan data, biasanya melalui pola arsitektur seperti MVC (Model-View-Controller).

2. **Efisiensi Pengembangan:** Dengan framework, pengembang dapat menggunakan kembali kode dan komponen yang sudah ada, yang mempercepat proses pengembangan dan mengurangi kebutuhan untuk menulis kode dari awal.

3. **Keamanan:** Framework PHP biasanya dilengkapi dengan fitur keamanan bawaan seperti proteksi terhadap SQL Injection, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), dan validasi input, yang membantu mengamankan aplikasi dari berbagai jenis serangan.

4. **Pemeliharaan dan Skalabilitas:** Framework mempermudah pemeliharaan kode dengan struktur yang konsisten dan modular. Ini juga

mendukung pengembangan aplikasi yang skalabel, yang dapat berkembang seiring dengan pertumbuhan kebutuhan bisnis.

5. **Komunitas dan Dukungan:** Sebagian besar framework PHP memiliki komunitas pengguna yang besar dan aktif, yang menyediakan berbagai sumber daya seperti dokumentasi, tutorial, plugin, dan dukungan melalui forum.

6. **Komponen dan Alat Bantu:** Framework PHP sering kali menyertakan komponen dan alat bantu untuk berbagai tugas umum seperti routing URL, manajemen sesi, autentikasi, manajemen cache, pengelolaan database, dan lainnya.

Beberapa contoh framework populer dalam PHP antara lain:

- **Laravel:** Dikenal dengan sintaks yang elegan dan fitur lengkap seperti ORM (Eloquent), migrasi database, dan sistem templating.
- **CodeIgniter:** Framework yang ringan dan cepat dengan konfigurasi minim, cocok untuk pengembangan aplikasi sederhana.
- **Symfony:** Framework yang kuat dan fleksibel, sering digunakan untuk proyek skala besar dengan kebutuhan khusus.
- **Yii:** Framework yang mendukung pengembangan aplikasi yang cepat dan memiliki performa tinggi.

3.2.2. Pengertian Codeigniter

Codeigniter merupakan salah satu *Framework* MVC (*model-view-controller*) pada bahasa pemrograman PHP yang sering digunakan untuk membangun aplikasi web. (Subari et al., 2021) Penggunaan codeigniter dalam pembangunan web dapat mempercepat proses pengembangan, karena codeigniter menyediakan berbagai library dan fungsi yang dapat memudahkan pengembang dalam membangun aplikasi web . framework ini dipilih karena memiliki fitur-fitur yang

dapat memudahkan pengembangan aplikasi, seperti routing, database abstraction, form validation, dan lain-lain Welcome to CodeIgniter¶ (2016)(Welcome to CodeIgniter, 2023)(Build Your First Application, 2023).

Beberapa karakteristik utama dari CodeIgniter adalah:

1. **Ringan dan Cepat:** CodeIgniter dikenal karena ukurannya yang kecil dan kinerja yang cepat. Ini membuatnya ideal untuk aplikasi yang membutuhkan performa tinggi dengan konsumsi sumber daya yang minimal.

2. **Mudah Dipelajari:** CodeIgniter memiliki dokumentasi yang sangat baik dan sintaks yang mudah dipahami, sehingga cocok bagi pengembang pemula maupun berpengalaman.

3. **Konfigurasi Minim:** Framework ini dirancang agar mudah digunakan dengan sedikit konfigurasi awal, memungkinkan pengembang untuk fokus langsung pada pengembangan aplikasi tanpa harus khawatir tentang pengaturan yang rumit.

4. **Kompatibilitas yang Luas:** CodeIgniter mendukung berbagai versi PHP dan kompatibel dengan berbagai macam database seperti MySQL, PostgreSQL, dan lainnya.

5. **Keamanan:** CodeIgniter memiliki berbagai fitur keamanan bawaan, seperti proteksi terhadap serangan SQL Injection, XSS (Cross-Site Scripting), dan CSRF (Cross-Site Request Forgery).

6. **Ekstensi dan Komunitas:** Framework ini memiliki sistem plugin yang memungkinkan pengembang untuk menambah fungsionalitas dengan mudah. Selain itu, terdapat komunitas yang aktif dan banyak sumber daya yang tersedia, seperti tutorial, forum, dan modul tambahan.

3.2.3. Pengertian Rapid Application Development (RAD)

Rapid Application Development adalah sebuah metodologi pengembangan sistem informasi yang berfokus pada pembuatan aplikasi secara cepat. RAD

merupakan model proses pengembangan perangkat lunak yang bersifat inkremental dan iteratif, dengan menekankan pada kolaborasi erat antara pengembang dan pengguna untuk menghasilkan aplikasi yang sesuai dengan kebutuhan dalam waktu yang relatif singkat.

Dalam pendekatan RAD, pengembangan aplikasi dilakukan melalui penggunaan teknik-teknik pemrograman yang efisien, seperti prototyping dan CASE tools, serta menekankan pada pengujian yang intensif untuk meminimalisir adanya kesalahan program. Hal ini memungkinkan pengembangan aplikasi dapat dilakukan dengan cepat dan fleksibel, serta dapat merespon perubahan kebutuhan pengguna dengan lebih baik dibandingkan dengan metode pengembangan tradisional seperti waterfall.

Sedangkan menurut Bentley, L.D. dan Whitten, J.L., Rapid Application Development adalah sebuah model proses pengembangan perangkat lunak yang bersifat inkremental dan iteratif, dengan fokus utama pada memenuhi kebutuhan pengguna dalam waktu yang relatif singkat. RAD menekankan pada penggunaan prototipe dan kolaborasi yang erat antara pengembang dan pengguna untuk menghasilkan aplikasi yang sesuai dengan kebutuhan secara cepat. Berbeda dengan model pengembangan tradisional seperti waterfall, RAD mendorong penggunaan teknik-teknik pemrograman yang efisien dan pengujian yang intensif untuk mempercepat siklus pengembangan dan mengurangi risiko kegagalan proyek.

Pendekatan Rapid Application Development menawarkan keuntungan dalam hal percepatan pengembangan aplikasi dibandingkan dengan metode tradisional seperti waterfall.

Beberapa karakteristik utama dari RAD adalah:

1. **Prototyping:** RAD mengutamakan pembuatan prototipe yang cepat, di mana pengembang dan pengguna bekerja sama untuk menghasilkan versi awal dari perangkat lunak. Prototipe ini digunakan untuk mengeksplorasi persyaratan dan mendapatkan umpan balik secara langsung dari pengguna.

2. **Iterasi Cepat:** RAD mendorong pengembangan iteratif, di mana perangkat lunak dikembangkan dalam siklus pendek dan berulang. Setiap iterasi menghasilkan versi baru yang semakin mendekati produk akhir.

3. **Keterlibatan Pengguna:** Pengguna akhir terlibat secara aktif dalam seluruh proses pengembangan, memberikan umpan balik yang cepat untuk memastikan bahwa perangkat lunak memenuhi kebutuhan mereka.

4. **Komponen yang Dapat Digunakan Kembali:** RAD sering menggunakan komponen perangkat lunak yang sudah ada atau yang dapat digunakan kembali, yang mempercepat proses pengembangan.

5. **Pengembangan Tim yang Kolaboratif:** Tim pengembang, yang sering kali terdiri dari berbagai disiplin ilmu, bekerja secara kolaboratif untuk mencapai tujuan pengembangan dalam waktu yang lebih singkat.

RAD cocok untuk proyek yang membutuhkan pengembangan cepat, memiliki persyaratan yang dinamis, atau membutuhkan keterlibatan pengguna yang tinggi. Namun, metodologi ini mungkin kurang cocok untuk proyek besar yang membutuhkan dokumentasi yang sangat detail dan pengendalian yang ketat atas setiap tahap pengembangan.

3.2.4. Perancangan Beroirientasi Objek (OOAD)

Perancangan Berorientasi Objek (Object-Oriented Design, OOD) adalah pendekatan dalam pengembangan perangkat lunak yang berfokus pada pemodelan sistem berdasarkan objek-objek nyata atau konsep-konsep yang ada dalam dunia nyata. Setiap objek merepresentasikan entitas tertentu yang memiliki atribut (data) dan metode (fungsi atau perilaku) yang berhubungan dengan objek tersebut.
Pendekatan pengembangan perangkat lunak yang berorientasi objek merupakan suatu metodologi yang berfokus pada pemodelan sistem melalui objek-objek, di mana setiap objek mewakili entitas tertentu dengan atribut dan metode yang saling terkait. Metodologi ini memungkinkan pembangunan sistem perangkat lunak secara sistematis berdasarkan konsep-konsep dan entitas nyata dalam dunia nyata.

Dalam OOD, prinsip-prinsip utama yang diterapkan meliputi:

1. **Encapsulation (Enkapsulasi):** Menyembunyikan detail internal dari sebuah objek dan hanya menyediakan akses terhadap data melalui metode yang telah ditentukan.
2. **Inheritance (Pewarisan):** Memungkinkan objek baru untuk mewarisi atribut dan metode dari objek yang sudah ada, yang disebut sebagai kelas induk (superclass).
3. **Polymorphism (Polimorfisme):** Kemampuan untuk menggunakan metode yang sama pada objek yang berbeda, dimana setiap objek dapat merespons dengan cara yang berbeda-beda.
4. **Abstraction (Abstraksi):** Menyederhanakan kompleksitas dengan memfokuskan hanya pada aspek penting dari objek dan menyembunyikan detail yang tidak relevan.

Tujuan dari perancangan berorientasi objek adalah untuk menciptakan sistem yang modular, dapat dipelihara, dan mudah untuk dikembangkan lebih lanjut. Metodologi ini sangat efektif dalam pengembangan sistem yang kompleks, karena memungkinkan pengembang untuk memecah sistem menjadi bagian-bagian yang lebih kecil dan lebih mudah diatur.

3.3. Metode Rapid Application Development (RAD)

Tahapan penerapan ***Rapid Application Development (RAD)*** untuk perancangan **Sistem Surat Perintah Perjalanan Dinas (SPPD)** di Pengadilan Tanjung Karang,

3.3.1. Perencanaan Persyaratan (Requirements Planning)

a. **Identifikasi Tujuan:**

   - Tujuan utama adalah mengembangkan sistem yang mempermudah dan mempercepat proses pembuatan, persetujuan, dan pelaporan Surat Perintah Perjalanan Dinas (SPPD).

b. **Pengumpulan Kebutuhan:**

- Mengadakan pertemuan dengan pemangku kepentingan di Pengadilan Tanjung Karang, termasuk bagian kepegawaian, keuangan, dan pejabat yang berwenang.
- Diskusi fokus pada alur kerja saat ini, tantangan yang dihadapi, dan kebutuhan utama seperti otomatisasi pembuatan SPPD, notifikasi persetujuan, serta integrasi dengan sistem keuangan.

c. **Analisis Kebutuhan:**

- Menyusun daftar kebutuhan fungsional, seperti pembuatan dokumen SPPD secara otomatis, pelacakan status, dan pengelolaan database perjalanan dinas.
- Identifikasi kebutuhan non-fungsional seperti keamanan data, kecepatan sistem, dan kemudahan penggunaan.

3.3.2. Proses Analisis Kebutuhan

a. **Studi Alur Kerja:**

- Menganalisis alur kerja yang ada untuk SPPD, mulai dari pengajuan oleh pegawai hingga persetujuan oleh pejabat terkait dan pelaporan setelah perjalanan dinas selesai.

b. **Identifikasi Pemangku Kepentingan:**

- Mengidentifikasi semua pihak yang terlibat dalam proses SPPD, termasuk pegawai yang mengajukan, atasan yang menyetujui, bagian keuangan, dan admin sistem.

c. **Penentuan Prioritas Kebutuhan:**

- Mengkategorikan kebutuhan menjadi prioritas tinggi, menengah, dan rendah berdasarkan urgensi dan dampaknya terhadap efisiensi proses SPPD.

3.3.3. Desain dan Pembangunan Prototipe (User Design)

a. **Pengembangan Prototipe Awal:**

- Membuat prototipe awal dari sistem SPPD, mencakup fitur utama seperti form pengajuan SPPD, dashboard persetujuan, dan sistem notifikasi.

b. **Iterasi Desain dengan Umpan Balik Pengguna:**

- Melibatkan pengguna dari berbagai divisi untuk menguji prototipe. Mengumpulkan umpan balik dan melakukan perbaikan sesuai kebutuhan.

c. **Pengujian Alur Kerja:**

- Menguji alur kerja pada prototipe untuk memastikan kemudahan penggunaan dan kecepatan proses pengajuan dan persetujuan SPPD.

3.3.4. Proses Perancangan Sistem

a. **Desain Arsitektur Sistem:**

- Merancang arsitektur sistem yang akan digunakan, termasuk desain database untuk menyimpan data SPPD, desain antarmuka pengguna (UI), dan integrasi dengan sistem lain seperti keuangan dan kepegawaian.

b. **Desain Database:**

- Merancang database yang efisien untuk menyimpan data SPPD, data pegawai, data perjalanan dinas, dan histori persetujuan.

c. **Desain Antarmuka Pengguna:**

- Merancang UI/UX yang intuitif dan mudah digunakan oleh pegawai, atasan, dan bagian keuangan. Antarmuka harus mendukung pembuatan dan persetujuan SPPD dengan langkah-langkah yang jelas.

3.3.5. Pembangunan (Construction)

a. **Pengembangan Fitur Lengkap:**

- Menyelesaikan pengembangan semua fitur yang dibutuhkan, termasuk pencetakan SPPD, pelacakan status, dan laporan perjalanan dinas.

b. **Pengujian dan Penyempurnaan:**

- Melakukan pengujian menyeluruh pada sistem untuk memastikan tidak ada bug dan semua fungsi berjalan sesuai kebutuhan.

c. **Penyusunan Dokumentasi:**

- Menyusun dokumentasi teknis dan manual pengguna untuk mendukung implementasi dan pemeliharaan sistem.

3.3.6. Penerapan (Cutover)

a. **Pelatihan Pengguna:**

- Memberikan pelatihan kepada pegawai dan pejabat terkait tentang cara menggunakan sistem SPPD yang baru.

b. **Implementasi Sistem:**

- Meluncurkan sistem di lingkungan localhost, memastikan semua data dan sistem terintegrasi dengan baik.

c. **Pemeliharaan dan Dukungan:**

- Memberikan dukungan teknis pasca-implementasi dan melakukan pemeliharaan berkala untuk memastikan sistem berjalan optimal.

3.4. Perancangan Sistem dengan pendekatan Objek Oriented (OOAD)

3.4.1. Identifikasi Pengguna/user

a. Administrator/Keuangan " Memiliki akses penuh untuk mengelola data dan laporan. "
b. User Biasa "Hanya dapat melihat riwayat perjalanan dinasnya sendiri"

Administrator

1. Mengelola laporan perjalanan dinas:

a. Laporan per orang (individu).

b. Laporan per bagian (departemen).

c. Laporan per golongan.

d. Laporan per jenis jabatan.

e. Laporan per perjalanan.

f. Laporan siklus harian, bulanan, hingga tahunan.

2. Mendaftarkan pengguna baru (user dan administrator).

3. Mengelola data master:

   a. Data pegawai.

   b. Data pejabat.

   c. Tarif standar minimum biaya perjalanan dinas.

4. Mengelola data perjalanan dinas.

User Biasa

1. Melihat riwayat perjalanan dinasnya sendiri.

3.4.2. Sekenario Sistem

Tabel 3. 1 *Use Case Login ke Sistem*

Aktor: Administrator, User Biasa
Deskripsi: Pengguna melakukan login ke sistem untuk mengakses fitur sesuai hak akses.
Pre-kondisi: Pengguna sudah terdaftar di sistem.
Langkah-langkah Utama:
1. Pengguna mengakses halaman login.
2. Pengguna memasukkan email dan kata sandi.

3. Sistem memverifikasi username dan password pengguna.
4. Jika valid, sistem mengarahkan pengguna ke dashboard sesuai hak akses.
Post-kondisi: Pengguna berhasil login dan diarahkan ke dashboard.
Alternatif:
Jika username dan password tidak valid, sistem menampilkan pesan kesalahan.

Tabel 3. 2. *Use Case Mengelola Data Pegawai*

Aktor: Administrator
Deskripsi: Administrator dapat melakukan CRUD (Create, Read, Update, Delete) data pegawai.
Pre-kondisi: Administrator sudah login ke sistem.
Langkah-langkah Utama:
1. Administrator memilih menu "Data Pegawai".
2. Sistem menampilkan daftar pegawai.
3. Administrator dapat:
i. Menambahkan pegawai baru dengan mengisi form dan menyimpannya.
ii. Mengedit data pegawai yang ada.
iii. Menghapus data pegawai.
4. Sistem menyimpan perubahan ke database.
Post-kondisi: Data pegawai berhasil dikelola.
Alternatif:
Jika data tidak valid, sistem menampilkan pesan kesalahan.

Tabel 3. 3. Use Case Mengelola Data Pejabat

Aktor: Administrator
Deskripsi: Administrator dapat melakukan CRUD data pejabat.
Pre-kondisi: Administrator sudah login ke sistem.
Langkah-langkah Utama:
1. Administrator memilih menu "Data Pejabat".
2. Sistem menampilkan daftar pejabat.
3. Administrator dapat:
i. Menambahkan pejabat baru.
ii. Mengedit data pejabat yang ada.
iii. Menghapus data pejabat.
4. Sistem menyimpan perubahan ke database.
Post-kondisi: Data pejabat berhasil dikelola.
Alternatif:
Jika data tidak valid, sistem menampilkan pesan kesalahan.

Tabel 3. 4. Use Case Mengelola Tarif Standar Minimum Biaya

Aktor: Administrator
Deskripsi: Administrator dapat mengelola tarif standar minimum biaya.
Pre-kondisi: Administrator sudah login ke sistem.
Langkah-langkah Utama:
1. Administrator memilih menu "Tarif Standar Minimum".
2. Sistem menampilkan daftar tarif standar minimum biaya.
3. Administrator dapat:
i. Menambahkan tarif standar baru.
ii. Mengedit tarif standar yang ada.
iii. Menghapus tarif standar.
4. Sistem menyimpan perubahan ke database.
Post-kondisi: Tarif standar minimum biaya berhasil dikelola.
Alternatif:

Jika data tidak valid, sistem menampilkan pesan kesalahan.

Tabel 3. 5. Use Case Membuat Surat Perintah Perjalanan Dinas (SPPD)

Aktor: Administrator
Deskripsi: Administrator dapat membuat surat perintah perjalanan dinas baru.
Pre-kondisi: Administrator sudah login ke sistem.
Langkah-langkah Utama:
1. Administrator memilih menu "Buat SPPD".
2. Sistem menampilkan form input untuk SPPD.
3. Administrator mengisi data SPPD, seperti tujuan perjalanan, tanggal, pejabat yang menandatangani, dll.
4. Administrator menyimpan data SPPD.
5. Sistem menyimpan data SPPD ke database.
Post-kondisi: SPPD baru berhasil dibuat.
Alternatif:
Jika data tidak valid, sistem menampilkan pesan kesalahan.

Tabel 3. 6. Use Case Mengelola SPPD

Aktor: Administrator
Deskripsi: Administrator dapat mengelola SPPD yang sudah ada.
Pre-kondisi: Administrator sudah login ke sistem.
Langkah-langkah Utama:
1. Administrator memilih menu "Kelola SPPD".
2. Sistem menampilkan daftar SPPD yang sudah ada.
3. Administrator dapat:
i. Mengedit SPPD yang ada.
ii. Menghapus SPPD.
4. Sistem menyimpan perubahan ke database.
Post-kondisi: SPPD berhasil dikelola.
Alternatif:

Jika data tidak valid, sistem menampilkan pesan kesalahan.

Tabel 3. 7. Use Case Melihat Riwayat Perjalanan Dinas

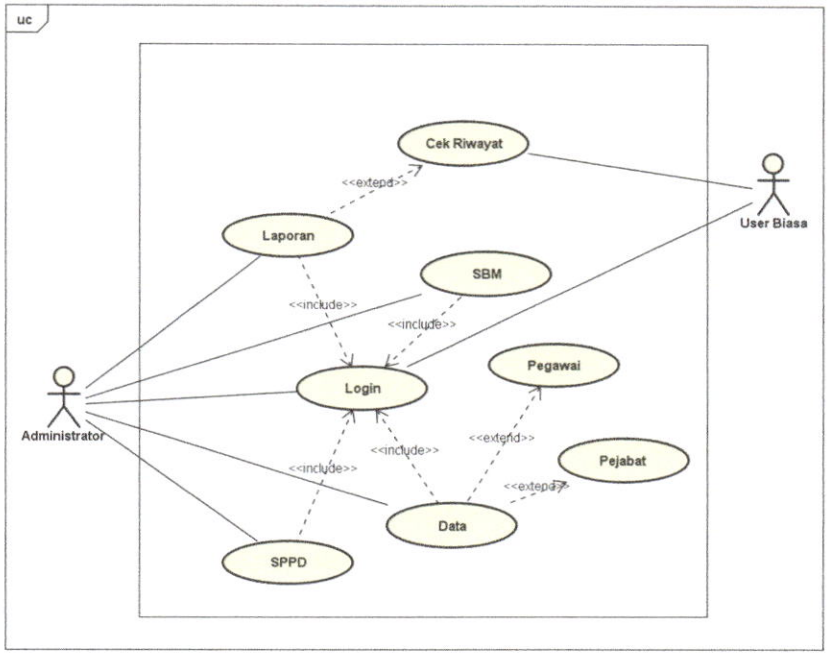
Aktor: User Biasa
Deskripsi: User biasa dapat melihat riwayat perjalanan dinasnya sendiri.
Pre-kondisi: User sudah login ke sistem.
Langkah-langkah Utama:
1. User memilih menu "Riwayat Perjalanan Dinas".
2. Sistem menampilkan daftar riwayat perjalanan dinas user tersebut.
3. User dapat melakukan pencarian atau filter berdasarkan kriteria tertentu.
Post-kondisi: Riwayat perjalanan dinas user berhasil ditampilkan.
Alternatif:
Jika data tidak ditemukan, sistem menampilkan pesan bahwa tidak ada data yang sesuai.

Tabel 3. 8. Use Case Mengelola Laporan Perjalanan Dinas

Aktor: Administrator
Deskripsi: Administrator dapat mengelola laporan perjalanan dinas berdasarkan berbagai kriteria.
Pre-kondisi: Administrator sudah login ke sistem.
Langkah-langkah Utama:
1. Administrator memilih menu "Laporan Perjalanan Dinas".
2. Sistem menampilkan opsi kriteria laporan (per orang, per bagian, per golongan, per jenis jabatan, per perjalanan, harian, bulanan, tahunan).
3. Administrator memilih kriteria dan periode laporan.
4. Sistem menampilkan laporan sesuai kriteria yang dipilih.
5. Administrator dapat mengunduh laporan dalam format PDF

atau Excel.
Post-kondisi: Laporan perjalanan dinas berhasil ditampilkan dan/atau diunduh.
Alternatif:
Jika tidak ada data yang sesuai dengan kriteria, sistem menampilkan pesan bahwa data tidak ditemukan.

3.4.3. Usecase Diagram



Gambar 3. 1. *Usecase diagram sistem surat perintah perjalanan dinas*

Penjelasan Diagram:

- **Aktor**:
    a. **Administrator**: Memiliki akses untuk mengelola data, membuat SPPD, mengelola SPPD, dan laporan.
    b. **User Biasa**: Hanya dapat melihat riwayat perjalanan dinas.
- **Use Cases**:

a. **Login**: Proses autentikasi untuk semua pengguna.

b. **Data:**

- Mengelola Data Pegawai (CRUD data pegawai.)
- Mengelola Data Pejabat (CRUD data pejabat)

c. **SBM:** Mengelola Tarif Standar Minimum Biaya, CRUD tarif standar.

d. **SPPD:**

- Membuat SPPD: Proses pembuatan SPPD baru.
- Mengelola SPPD: Edit dan hapus SPPD.

e. **Cek Riwayat:** Melihat Riwayat Perjalanan Dinas, User biasa melihat riwayatnya.

f. **Laporan:** Mengelola Laporan Perjalanan Dinas, Administrator membuat laporan perjalanan dinas.

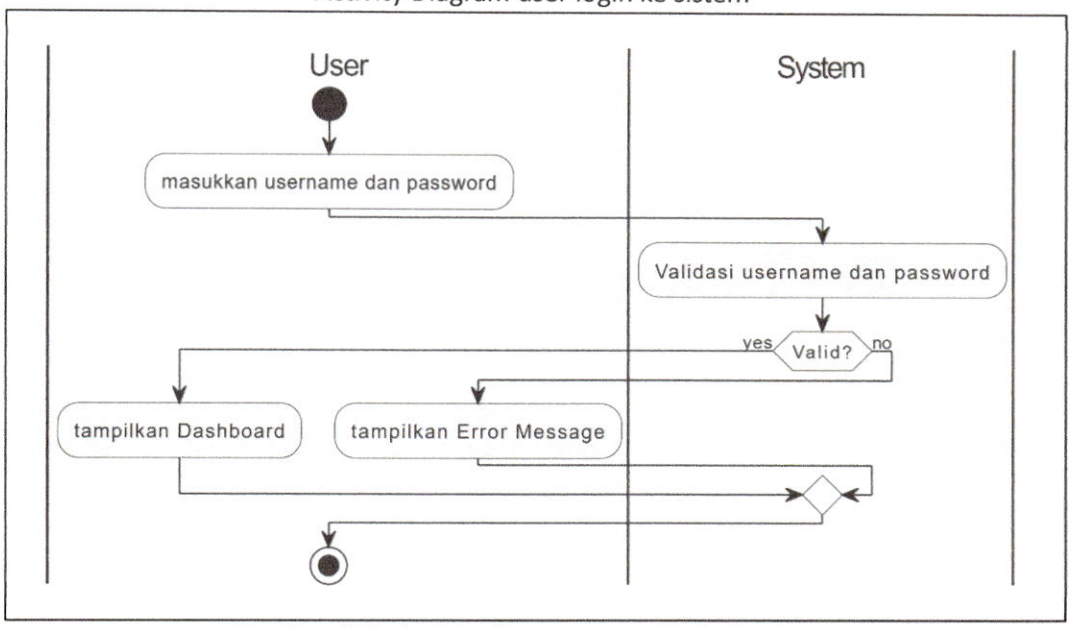
- **Hubungan**:

a. **<<include>>**: Use case lain yang diperlukan untuk menjalankan use case utama (misalnya, Login diperlukan untuk semua operasi).

b. **<<extend>>**: Use case tambahan yang bisa memperluas use case utama (misalnya, Mengelola SPPD diperluas dari Membuat SPPD jika ada perubahan).

3.4.4. Activity Diagram

Activity Diagram user login ke sistem



Gambar 3. 2. *Activity Diagram user login ke sistem*

Penjelasan :

Pengguna:

- Mengakses halaman login.
- Memasukkan email dan kata sandi.

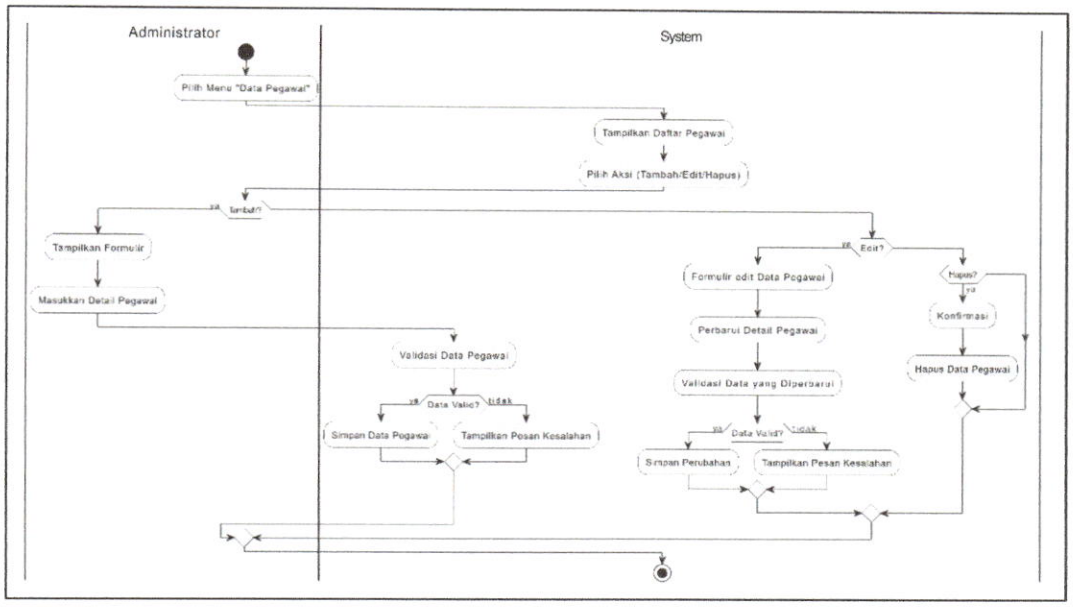
System:

- Memverifikasi username dan password.
- Jika valid, mengarahkan pengguna ke dashboard yang sesuai dengan hak akses.
- Jika tidak valid, menampilkan pesan kesalahan.

Post-kondisi:

- Pengguna berhasil login dan diarahkan ke dashboard setelah proses verifikasi selesai.

Activity Diagram Mengelola Data Pegawai



Gambar 3. 3. *Activity Diagram Mengelola Data Pegawai*

Penjelasan :

Administrator:

- Memilih menu "Data Pegawai".
- Memilih aksi yang diinginkan: tambah, edit, atau hapus data pegawai.
- Untuk aksi tambah, mengisi formulir dengan detail pegawai baru.
- Untuk aksi edit, memperbarui data pegawai yang ada.
- Untuk aksi hapus, melakukan konfirmasi penghapusan pegawai.

System:

- Menampilkan daftar pegawai.
- Memvalidasi data pegawai jika Administrator melakukan penambahan atau pengeditan.
- Menyimpan data pegawai baru, perubahan data pegawai, atau menghapus data pegawai sesuai dengan aksi Administrator.

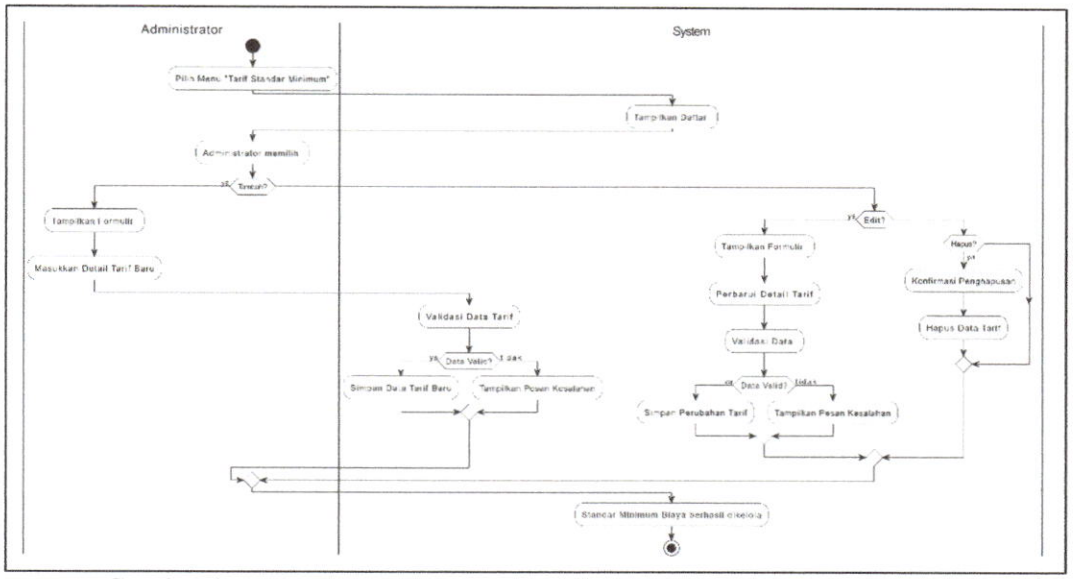
Validasi dan Penanganan Kesalahan:

- Sistem memvalidasi data yang dimasukkan oleh Administrator dan menampilkan pesan kesalahan jika data tidak valid.

Post-kondisi:

- Data pegawai berhasil dikelola setelah tindakan yang sesuai dilakukan.

Activity Diagram Mengelola **Tarif Standar Minimum Biaya**



Gambar 3. 4. *Activity Diagram Mengelola Tarif Standar Minimum Biaya*

Penjelasan :

Administrator:

- Memilih menu "Tarif Standar Minimum".
- Memilih aksi yang diinginkan: menambah, mengedit, atau menghapus tarif standar.
- Untuk aksi tambah, mengisi formulir dan memasukkan detail tarif baru.
- Untuk aksi edit, mengisi formulir edit dan memperbarui detail tarif.
- Untuk aksi hapus, melakukan konfirmasi penghapusan.

System:

- Menampilkan daftar tarif standar minimum biaya.
- Memvalidasi data tarif jika Administrator menambah atau mengedit tarif.
- Menyimpan perubahan tarif atau menghapus data tarif sesuai dengan aksi Administrator.

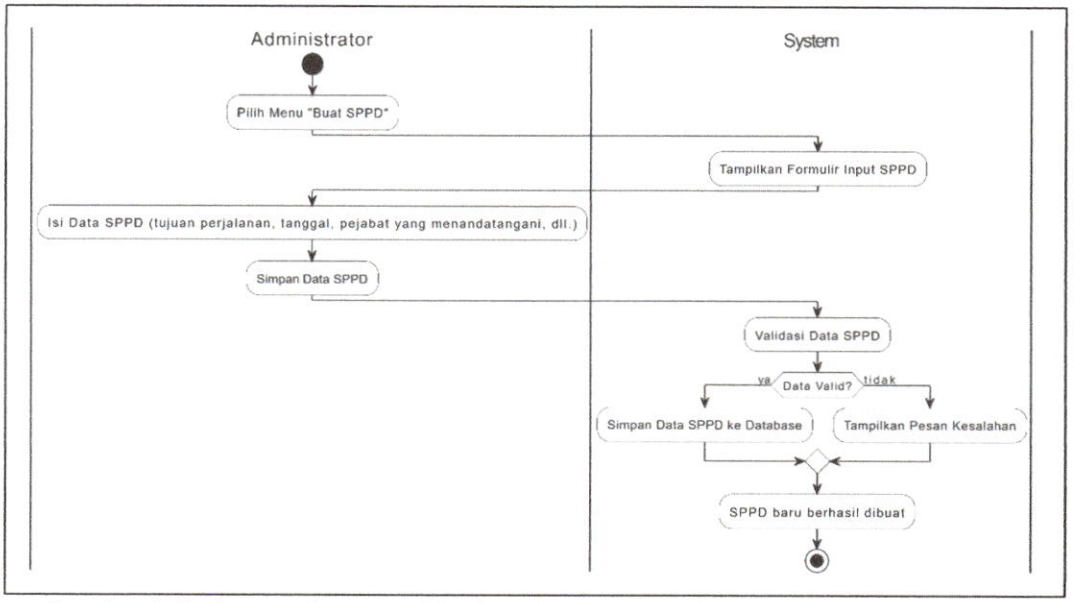
Validasi dan Penanganan Kesalahan:

- Sistem memvalidasi data yang dimasukkan Administrator dan menampilkan pesan kesalahan jika data tidak valid.

Post-kondisi:

- Tarif standar minimum biaya berhasil dikelola setelah tindakan yang sesuai dilakukan.

Activity Diagram **Membuat Surat Perintah Perjalanan Dinas (SPPD)**



Gambar 3. 5. *Activity Diagram Membuat Surat Perintah Perjalanan Dinas (SPPD)*

Penjelasan :

Administrator:

- Memilih menu "Buat SPPD".
- Mengisi data SPPD seperti tujuan perjalanan, tanggal, pejabat yang menandatangani, dan informasi relevan lainnya.
- Menyimpan data SPPD setelah pengisian.

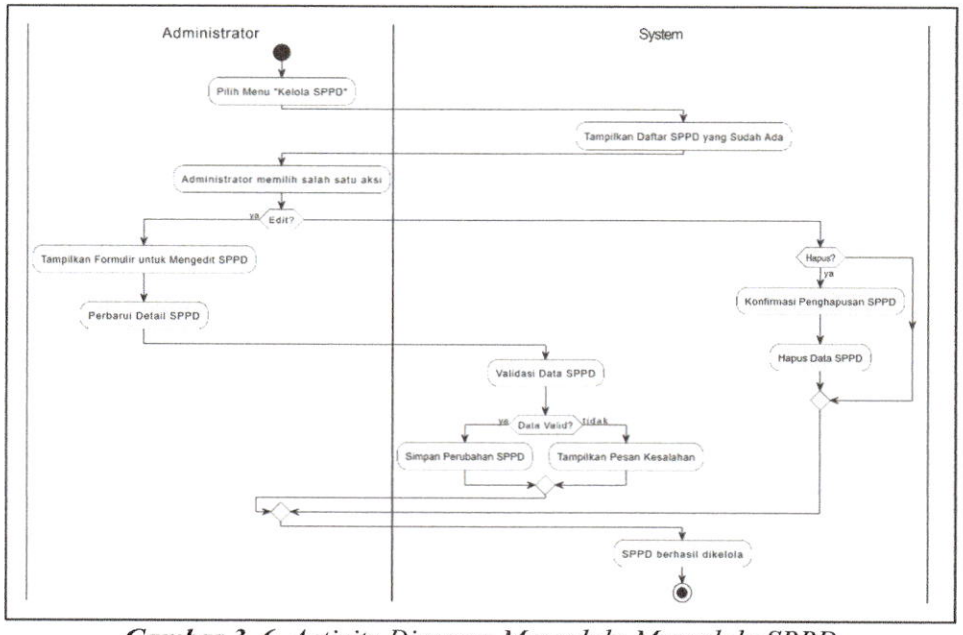
System:

- Menampilkan formulir input untuk membuat SPPD.
- Memvalidasi data SPPD yang dimasukkan oleh Administrator.
- Menyimpan data SPPD ke database jika data valid.
- Menampilkan pesan kesalahan jika data tidak valid.

Post-kondisi:

- SPPD baru berhasil dibuat dan disimpan setelah proses validasi.

Activity Diagram Mengelola Mengelola SPPD



***Gambar 3. 6**. Activity Diagram Mengelola Mengelola SPPD*

Penjelasan :

Administrator:

- Memilih menu "Kelola SPPD".
- Memilih aksi (Edit/Hapus) dan melakukan tindakan yang sesuai.
- Untuk aksi Edit, Administrator mengisi formulir edit dan memperbarui detail SPPD.
- Untuk aksi Hapus, Administrator melakukan konfirmasi penghapusan.

System:

- Menampilkan daftar SPPD yang sudah ada.
- Memvalidasi data jika Administrator mengedit SPPD.
- Menghapus data SPPD sesuai dengan konfirmasi dari Administrator.

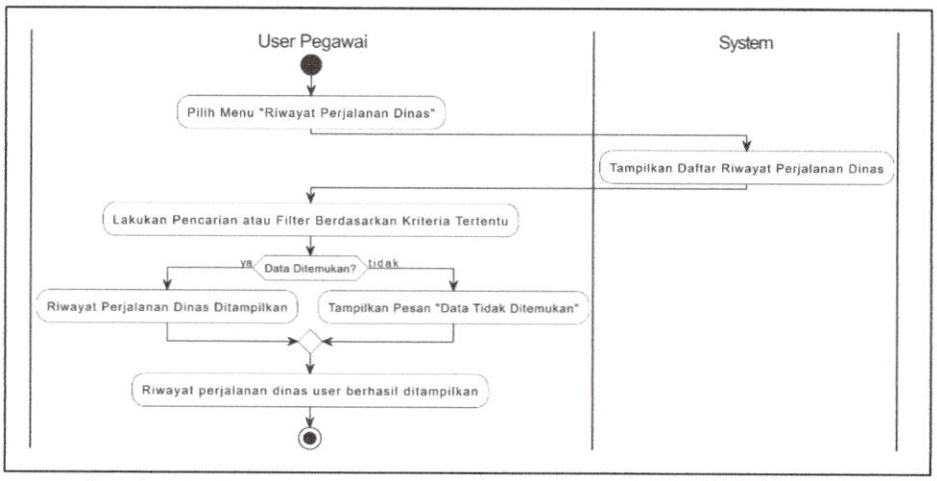
Validasi dan Penanganan Kesalahan:

- Jika data tidak valid, sistem menampilkan pesan kesalahan.

Post-kondisi:

- SPPD berhasil dikelola setelah tindakan yang sesuai dilakukan.

Activity Diagram **Melihat Riwayat Perjalanan Dinas**



***Gambar 3.** 7. Activity Diagram Melihat Riwayat Perjalanan Dinas*

Penjelasan :

User Pegawai:

- Memilih menu "Riwayat Perjalanan Dinas".
- Melakukan pencarian atau filter untuk menemukan riwayat perjalanan dinas yang relevan.

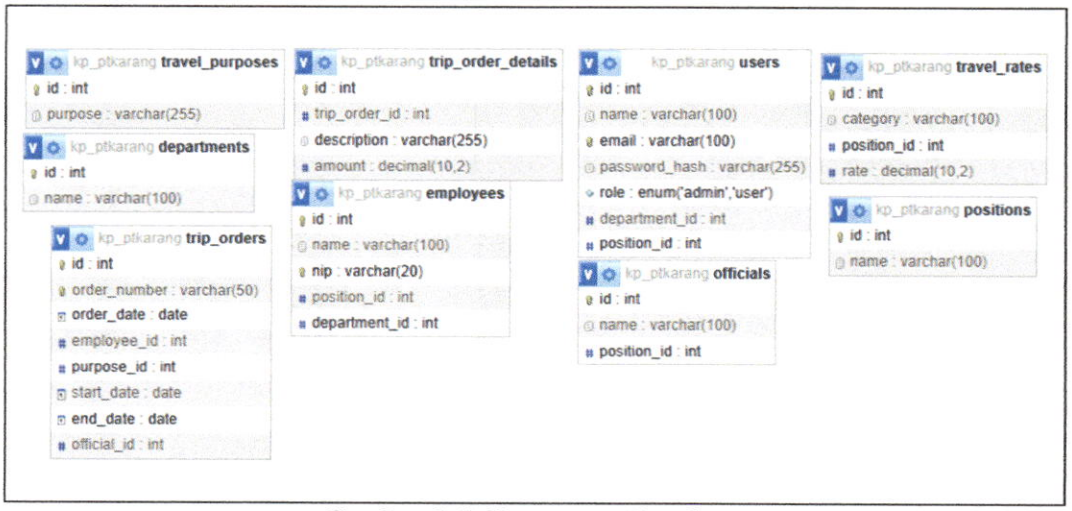
System:

- Menampilkan daftar riwayat perjalanan dinas untuk user tersebut.
- Menampilkan pesan "Data Tidak Ditemukan" jika tidak ada data yang sesuai dengan kriteria pencarian atau filter.

Post-kondisi:

- Riwayat perjalanan dinas user berhasil ditampilkan atau pesan yang sesuai ditampilkan jika data tidak ditemukan.

3.4.5. Rancangan Database



***Gambar 3. 8**. Rancangan Database*

SQL Query :

```
CREATE TABLE departments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE positions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role ENUM('admin', 'user') NOT NULL,
    department_id INT,
    position_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(id),
    FOREIGN KEY (position_id) REFERENCES positions(id)
);
CREATE TABLE officials (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    position_id INT,
    FOREIGN KEY (position_id) REFERENCES positions(id)
);
CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    nip VARCHAR(20) UNIQUE NOT NULL,
    position_id INT,
    department_id INT,
    FOREIGN KEY (position_id) REFERENCES positions(id),
    FOREIGN KEY (department_id) REFERENCES departments(id)
);
CREATE TABLE travel_purposes (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    purpose VARCHAR(255) NOT NULL
);
CREATE TABLE travel_rates (
    id INT AUTO_INCREMENT PRIMARY KEY,
    category VARCHAR(100) NOT NULL,
    position_id INT,
    rate DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (position_id) REFERENCES positions(id)
);
CREATE TABLE trip_orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_number VARCHAR(50) UNIQUE NOT NULL,
    order_date DATE NOT NULL,
    employee_id INT,
    purpose_id INT,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    official_id INT,
    FOREIGN KEY (employee_id) REFERENCES employees(id),
    FOREIGN KEY (purpose_id) REFERENCES travel_purposes(id),
    FOREIGN KEY (official_id) REFERENCES officials(id)
);
CREATE TABLE trip_order_details (
    id INT AUTO_INCREMENT PRIMARY KEY,
    trip_order_id INT,
    description VARCHAR(255),
    amount DECIMAL(10, 2),
    FOREIGN KEY (trip_order_id) REFERENCES trip_orders(id)
);
```

Penjelasan Skema

1. **users**: Tabel ini menyimpan informasi pengguna dengan kolom untuk peran (administrator atau user biasa), serta referensi ke departemen dan posisi.
2. **departments**: Tabel ini menyimpan informasi departemen.
3. **positions**: Tabel ini menyimpan informasi posisi/jabatan.
4. **officials**: Tabel ini menyimpan informasi pejabat yang dapat menandatangani surat perintah.
5. **employees**: Tabel ini menyimpan informasi pegawai dan terhubung dengan posisi dan departemen.
6. **travel_purposes**: Tabel ini menyimpan tujuan perjalanan dinas.
7. **travel_rates**: Tabel ini menyimpan tarif standar biaya perjalanan dinas berdasarkan kategori dan posisi.
8. **trip_orders**: Tabel ini menyimpan data utama surat perintah perjalanan dinas.
9. **trip_order_details**: Tabel ini menyimpan rincian biaya terkait perjalanan dinas.