Sequence Diagram - Alur Menghasilkan Laporan Penjualan

Diagram ini memvisualisasikan urutan interaksi sistem saat User Pemilik atau User Pegawai meminta untuk melihat laporan penjualan, misalnya dalam rentang tanggal tertentu.

Aktor dan Komponen yang Terlibat

1. **User (Pemilik/Pegawai):** Pengguna yang ingin melihat laporan di dasbor.
2. **main-app (PHP):** Aplikasi backend yang memproses permintaan dan mengagregasi data.
3. **postgres-db (Database):** Tempat mengambil dan memproses data penjualan.

Visualisasi Diagram (PlantUML)

Gunakan kode berikut di PlantUML Web Server untuk menghasilkan diagram visual dari alur ini.

Prompt untuk PlantUML:

```
@startuml
' Diagram Urutan untuk Alur Menghasilkan Laporan Penjualan

title Alur Proses Menghasilkan Laporan Penjualan

actor "User (Pemilik/Pegawai)" as User
participant "main-app (PHP)" as App
database "postgres-db" as DB

User -> App : 1. GET /reports/sales?start_date=...&end_date=...
activate App

App -> App : 2. Validasi input (format tanggal, hak akses user)
App -> App : 3. Ambil store_id dari sesi user

App -> DB : 4. Kirim Query Agregasi Penjualan\nSELECT DATE(order_date), SUM(total_amount) ...\nFROM orders WHERE store_id = ? AND order_date BETWEEN ? AND ? ...\nGROUP BY DATE(order_date)
activate DB

DB --> App : 5. Hasil data laporan (agregat)
deactivate DB

App -> App : 6. Proses dan format data untuk chart & tabel
```

App --> User : 7. Response (Tampilkan halaman laporan dengan visualisasi data)
deactivate App

@enduml

Penjelasan Langkah-demi-Langkah

1. **GET /reports/sales?start_date=...&end_date=...**
    - **Aksi:** Pengguna membuka halaman "Laporan", memilih rentang tanggal (misalnya, 1 Juli - 31 Juli), dan mengklik "Tampilkan Laporan".
    - **Aliran:** Peramban mengirimkan permintaan GET ke aplikasi PHP dengan rentang tanggal sebagai parameter URL.
2. **Validasi input**
    - **Aksi:** Aplikasi memastikan format tanggal yang diterima valid dan pengguna yang meminta laporan memiliki hak akses yang sesuai.
3. **Ambil store_id dari sesi user**
    - **Aksi:** Aplikasi mengambil store_id dari sesi pengguna yang sedang login untuk memastikan laporan yang ditampilkan hanya berasal dari toko mereka sendiri.
4. **Kirim Query Agregasi Penjualan**
    - **Aksi:** Ini adalah inti dari proses. Aplikasi membuat sebuah *query* SQL yang kompleks untuk meminta database melakukan perhitungan.
    - **Contoh Query:** "Berikan saya total penjualan (SUM(total_amount)) untuk setiap hari (GROUP BY DATE(order_date)) dari toko ini (WHERE store_id = ?) dalam rentang tanggal yang diberikan, dan hanya untuk pesanan yang statusnya 'completed' atau 'shipped'."
    - **Aliran:** PHP mengirim *query* agregasi ini ke PostgreSQL.
5. **Hasil data laporan (agregat)**
    - **Aksi:** Database melakukan semua perhitungan berat (menjumlahkan, mengelompokkan) dan mengembalikan hasilnya.
    - **Aliran:** Database mengirimkan hasil yang sudah "matang" dan ringkas kembali ke aplikasi PHP, bukan data mentah per pesanan. Misalnya: [{"tanggal": "2025-07-01", "total": 500000}, {"tanggal": "2025-07-02", "total": 750000}, ...]
6. **Proses dan format data untuk chart & tabel**
    - **Aksi:** Aplikasi PHP mengambil data agregat tersebut dan mengubah formatnya agar mudah digunakan oleh pustaka grafik (charting library) di sisi frontend (misalnya, untuk membuat grafik garis penjualan harian) dan untuk ditampilkan dalam bentuk tabel.
7. **Response (Tampilkan halaman laporan ...)**
    - **Aksi:** Aplikasi merender halaman laporan lengkap dengan grafik dan tabel.
    - **Aliran:** PHP mengirimkan halaman HTML final kembali ke peramban pengguna.

Ini adalah alur kerja terakhir yang penting untuk kita petakan. Kita telah mencakup registrasi, operasional inti, dan analisis. Dengan ini, tahap perancangan diagram urutan kita bisa

dianggap selesai.
Apakah Anda sudah merasa cukup dengan semua diagram yang telah kita buat dan siap untuk melanjutkan ke tahap implementasi?