Sequence Diagram - Alur Login Pengguna

Diagram ini memvisualisasikan urutan interaksi antar komponen sistem saat seorang pengguna mencoba untuk login. Ini membantu kita memahami alur kerja, dependensi, dan memastikan tidak ada langkah logika yang terlewat.

Aktor dan Komponen yang Terlibat

1. **User (Browser):** Pengguna yang berinteraksi melalui peramban web.
2. **reverse-proxy (Nginx):** Web server yang bertindak sebagai pintu gerbang.
3. **main-app (PHP-FPM):** Aplikasi backend yang memproses logika bisnis.
4. **postgres-db (Database):** Tempat menyimpan dan memverifikasi data pengguna.

Visualisasi Diagram (PlantUML)

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

Prompt untuk PlantUML:

```
@startuml
' Diagram Urutan untuk Alur Login Pengguna

title Alur Proses Login Pengguna

' Mendefinisikan Aktor dan Partisipan (Komponen)
actor "User (Browser)" as User
participant "reverse-proxy (Nginx)" as Nginx
participant "main-app (PHP)" as App
database "postgres-db" as DB

' Memulai alur
User -> Nginx : 1. POST /login (email, password)
activate Nginx

Nginx -> App : 2. fastcgi_pass (request)
activate App

App -> DB : 3. SELECT * FROM users WHERE email = [email]
activate DB

DB --> App : 4. User data (termasuk password_hash)
deactivate DB
```

```
App -> App : 5. Verifikasi password (password_verify)

alt Jika password valid

   App -> DB : 6. INSERT/UPDATE session data
   activate DB
   DB --> App : 7. Session ID
   deactivate DB

   App --> Nginx : 8. Response (Set-Cookie: session_id, redirect to dashboard)

else Jika password tidak valid

   App --> Nginx : Response (Error: Login Gagal)

end

Nginx --> User : 9. HTTP Response (Redirect atau Halaman Error)
deactivate App
deactivate Nginx

@enduml
```

Penjelasan Langkah-demi-Langkah

Mari kita lacak alur kerja ini sesuai dengan nomor pada diagram:

1. **POST /login (email, password)**
    - **Aksi:** Pengguna mengisi formulir login di peramban dan mengklik tombol "Login".
    - **Aliran:** Peramban mengirimkan permintaan POST yang berisi email dan kata sandi ke server Nginx.
2. **fastcgi_pass (request)**
    - **Aksi:** Nginx menerima permintaan. Karena tujuannya adalah file PHP, Nginx tidak memprosesnya sendiri.
    - **Aliran:** Nginx meneruskan seluruh permintaan tersebut ke layanan aplikasi PHP (main-app) melalui jaringan internal Docker.
3. **SELECT * FROM users WHERE email = [email]**
    - **Aksi:** Aplikasi PHP menerima data. Langkah pertama adalah memeriksa apakah pengguna dengan email tersebut ada di database.
    - **Aliran:** PHP mengirimkan *query* SELECT ke database PostgreSQL.
4. **User data (termasuk password_hash)**
    - **Aksi:** Database mencari data dan menemukannya.
    - **Aliran:** Database mengembalikan satu baris data pengguna, yang terpenting

adalah password_hash (kata sandi yang sudah dienkripsi).

5. **Verifikasi password (password_verify)**
    - **Aksi:** Aplikasi PHP sekarang memiliki kata sandi yang dikirim oleh pengguna dan *hash* kata sandi dari database. PHP menggunakan fungsi seperti password_verify() untuk membandingkan keduanya dengan aman.
    - **Aliran:** Ini adalah proses internal di dalam aplikasi PHP itu sendiri.
6. **alt Jika password valid ... else ... end**
    - Ini adalah blok kondisional yang menunjukkan dua kemungkinan hasil dari langkah #5.
7. **(Jika Valid) INSERT/UPDATE session data & Session ID**
    - **Aksi:** Jika kata sandi cocok, aplikasi membuat sesi baru untuk pengguna.
    - **Aliran:** PHP menyimpan informasi sesi (misalnya, user_id) ke dalam database (atau bisa juga ke Redis) dan mendapatkan ID unik untuk sesi tersebut.
8. **(Jika Valid) Response (Set-Cookie: session_id, ...)**
    - **Aksi:** Aplikasi PHP menyiapkan respons "sukses".
    - **Aliran:** PHP mengirimkan respons kembali ke Nginx. Respons ini berisi instruksi untuk peramban, seperti Set-Cookie (untuk menyimpan ID sesi) dan redirect (untuk mengarahkan pengguna ke halaman dasbor).
9. **(Jika Tidak Valid) Response (Error: Login Gagal)**
    - **Aksi:** Jika kata sandi tidak cocok, aplikasi menyiapkan respons "gagal".
    - **Aliran:** PHP mengirimkan respons kembali ke Nginx, biasanya dengan pesan error untuk ditampilkan kembali di halaman login.
10. **HTTP Response (Redirect atau Halaman Error)**
    - **Aksi:** Nginx menerima respons dari PHP dan meneruskannya kembali ke pengguna.
    - **Aliran:** Peramban pengguna menerima respons akhir. Jika sukses, peramban akan menyimpan *cookie* dan mengarahkan ke halaman dasbor. Jika gagal, ia akan menampilkan pesan error.

Diagram ini memberikan kejelasan penuh tentang bagaimana komponen-komponen yang berbeda bekerja sama dalam urutan yang tepat untuk menyelesaikan satu fungsi penting. Anda bisa membuat diagram serupa untuk fitur kompleks lainnya seperti "Membuat Pesanan" atau "Menerbitkan Produk".