Dokumentation: State Management i Flyhigh Blazor App

Formålet med denne dokumentation er at beskrive den valgte strategi for håndtering af bruger-login i vores Blazor-app. Da Flyhigh Hotel henvender sig til et klientel, der forventer den **højeste grad af sikkerhed og diskretion**, er vores valg baseret på et princip om maksimal datasikkerhed.

Den valgte løsning er en specialbygget AuthenticationStateProvider, der bruger browserens sessionStorage.

Vores Design-overvejelser:

For at levere en løsning, der matcher et luksusbrand, stillede vi os selv følgende spørgsmål:

- **Spørgsmål: Hvorfor ikke bare bruge en simpel "AppState"-klasse?**
    - **Svar:** Fordi en simpel løsning glemmer alt, så snart man genindlæser siden. For at give en professionel oplevelse skal systemet kunne huske et login under en aktiv session. Vigtigst af alt integrerer vores valgte system direkte med Blazors indbyggede sikkerhedsfunktioner som [Authorize], hvilket er afgørende for at kunne beskytte vores gæsters data korrekt.
- **Spørgsmål: For et luksushotel er brugeroplevelsen altafgørende. Hvorfor så vælge sessionStorage og ikke det mere bekvemme localStorage?**
    - **Svar:** Dette er det vigtigste sikkerhedsvalg. For et standard-hostel kunne localStorage måske være en acceptabel bekvemmelighed. For Flyhigh Hotel, hvor gæster kan være offentlige personer eller forretningsledere, er det ikke en mulighed. Vi **skal** vælge den mest sikre standard. sessionStorage fungerer som et **digitalt nøglekort, der automatisk deaktiveres**, når gæsten forlader hotellet (lukker browser-fanen). Dette sikrer, at en gæsts session ikke efterlades åben på f.eks. en offentlig eller delt computer – en risiko, vores klientel ikke kan acceptere.
- **Spørgsmål: Er et JWT-tokens udløbsdato så ikke sikkerhed nok i sig selv?**
    - **Svar:** Nej, de udgør to forskellige sikkerhedslag, som er standard for systemer med høje sikkerhedskrav. Tokenets udløbsdato er **serverens** forsvar (gør et stjålet nøglekort ubrugeligt efter en time). sessionStorage er **browserens** forsvar (minimerer risikoen for, at nøglekortet bliver stjålet i første omgang).

Teknisk Flow

1. **Login:** Brugeren logger ind. API'et returnerer et JWT. Vores APIService gemmer dette token i sessionStorage og notificerer vores CustomAuthenticationStateProvider.
2. **State Opdatering:** Provideren udløser en global event (NotifyAuthenticationStateChanged), der øjeblikkeligt opdaterer hele brugerfladen (f.eks. skifter "Login"-knappen ud med "Log ud").
3. **Logout:** Tokenet fjernes fra sessionStorage, og en lignende event udløses for at nulstille brugerfladen.
4. **Sideindlæsning:** Ved hver sideindlæsning tjekker CustomAuthenticationStateProvider automatisk sessionStorage. Hvis et gyldigt token findes, bliver brugeren anset for at være logget ind.

Konklusion

Vores løsning er en robust og sikker implementering, der bruger Blazors "Best Practice" til at levere en state management-løsning, der lever op til de høje standarder for diskretion og sikkerhed, som gæster på et luksushotel forventer.

Sidenote:

Vores idé er at give gæsten kontrol over sin egen balance mellem sikkerhed og bekvemmelighed. Systemet er som standard sat til maksimal sikkerhed (login udløber, når browseren lukkes), men med "Husk Mig"-funktionen anerkender vi, at en gæst på sin private computer kan ønske at forblive logget ind i længere tid for en mere gnidningsfri oplevelse.

Funktion for Brugeren

- Hvis "Husk Mig" IKKE er vinget af: Når brugeren lukker browser-fanen, bliver de logget ud.
- Hvis "Husk Mig" ER vinget af: Systemet vil huske brugerens login, selv efter browseren er lukket og genstartet. Brugeren vil forblive logget ind, indtil de aktivt trykker "Log ud", eller deres token udløber efter længere tid.

Hvordan det virker teknisk med localStorage

1. Ved Login: I vores APIService tjekker vi, om "Husk Mig" er markeret.
    - Hvis ja, gemmer vi JWT-tokenet i localStorage, som er persistent.
    - Hvis nej, gemmer vi det i sessionStorage, som slettes, når sessionen slutter.
2. Ved Opstart: Vores CustomAuthenticationStateProvider bliver opdateret til først at lede efter et token i sessionStorage. Hvis den ikke finder noget, leder den i localStorage.
3. Ved Logout: Vores LogoutAsync-metode opdateres til at slette tokenet fra både sessionStorage og localStorage for at sikre et rent logout.