

# Inloggning mot REST-API:et

- Vi lägger till funktionalitet i `LoginPage.tsx` för inloggning
- Samt skapa en `ProtectedRoute` -komponent för att skydda vissa routes

# Att skydda vissa routes

- Skapa komponenten ProtectedRoute
- Med hjälp av följande komponent kan vi skydda de routes som kräver inloggning

```
import { Navigate, Outlet } from "react-router-dom";

const ProtectedRoute = () => {
  //vi kommer att hämta jwt-token från API:et och spara i localStorage
  const token = localStorage.getItem("token");
  //om token finns i localStorage; navigera till tänkt route, annars till /
  return token ? <Outlet /> : <Navigate to="/" />;
};

export default ProtectedRoute;
```

# Skydda routes i App-komponenten

- Omgärda Routes som kräver inloggning med vår ProtectedRoute-komponent

```
import ProtectedRoute from "../components/ProtectedRoute";

//omslut /dashboard-routen med ProtectedRoute
<Route element={<ProtectedRoute />}>
  <Route path="/dashboard" element={<Dashboard />} />
</Route>;
```

- Nu går det inte längre att nå `/dashboard` utan en jwt-token

# LoginPage-komponenten uppdateringar

- Vi definierar två useState för username och password

```
//högst upp i filen bland andra imports
import { useState } from "react";
//i komponenten
const [username, setUsername] = useState("");
const [password, setPassword] = useState("");
```

# LoginPage-komponenten uppdateringar

- Vi skaffar även ett handtag till useNavigate-hooken - den ska användas för navigering

```
//högst upp i filen bland andra imports
import { useNavigate } from "react-router-dom";
//i komponenten
const navigate = useNavigate();
```

# LoginPage-komponenten uppdateringar

- Vi lägger till `onChange` -listeners på text-fälten för username och password
- Lyssnarna sätter username och password i respektive state

```
<input type="text" placeholder="Användarnamn" onChange={(e) => setUsername(e.target.value)}  
/>  
  
<input type="password" placeholder="Användarnamn" onChange={(e) => setPassword(e.target.value)}  
/>
```

# LoginPage-komponenten uppdateringar

- Vi uppdaterar onClick-funktionen `handleLogin()` i LoginPage
- Nästa slide

```
//omvandla till asynkron funktion
const handleLogin = async () => {
  try {
    //hämta jwt-token från vårt API
    const response = await fetch(
      "http://localhost:8080/token-service/v1/request-token",
      {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ username, password }),
      }
    );
    //vid felaktig inloggning
    if (!response.ok) {
      throw new Error("Felaktiga inloggningsuppgifter!");
    }
    //hämta token från responsen
    const token = await response.text();
    //Spara JWT-token
    localStorage.setItem("token", token);
    console.log(token);
    //Skicka användaren till dashboarden
    navigate("/dashboard");
  } catch (error) {
    if (error instanceof Error) {
      alert(error.message);
    }
  }
};
```



# Uppdatering i Dashboard-komponenten

- Vi ska fixa logout-knappen så att den fungerar
- Vi behöver importera hooken `useNavigate()`

```
import { useNavigate } from "react-router-dom";
```

- Och använda först i vår Dashboard-komponent:

```
const navigate = useNavigate();
```

## Till sist: uppdatering i Dashboard-komponenten

- Vi lägger till en `onClick` -handler på logout-knappen
- Denna tar bort token från `localStorage` och skickar användaren till login-sidan

```
const handleLogout = () => {  
  localStorage.removeItem("token"); // Ta bort token  
  navigate("/"); // Skicka användaren tillbaka till inloggningssidan  
};  
//lägg till onClick på utloggnings-knappen  
<button onClick={handleLogout}>Logga ut</button>;
```

# Starta databasen och backend

- Starta databasen och backend och försök att logga in från react-appen med:

```
username: admin@gmail.com
```

```
password: admin2025
```