

Protokoll: "SingleSignOn" mit OAuth

Angabe:

Ziel ist es eine SingleSignOn Möglichkeit (google login oder gerne auch eine alternative) mittels OAuth für eine Beispiel-Web-Applikationen zu implementieren bestehend aus Frontend und Backend. Welches (SPA-)Framework ihr dabei verwendet ist euch überlassen.

Euer Backend soll nur authentifizierten User zur Verfügung stehen.

Der Einfachheit halber, darf Firebase, Superbase oder ähnliches dazu verwendet werden.

Wichtig ist jedoch im Protokoll darzulegen, welche Schritte euch Firebase/etc. tatsächlich abnimmt und welche Schritte ihr implementieren musstet und vor allem auch wie ihr diese mit eurem verwendeten Frontend-Framework umgesetzt habt.

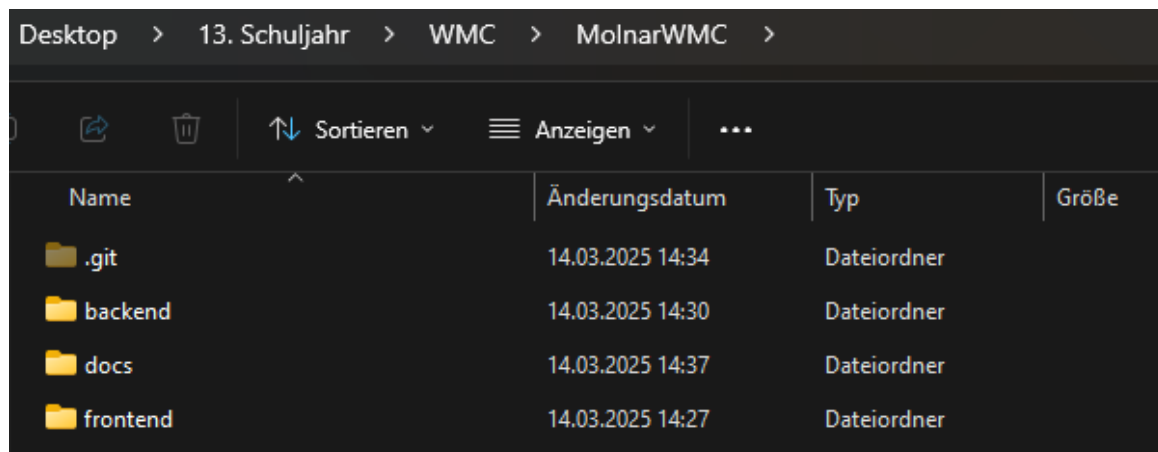
Legt auch dar, welches Problem OAuth eigentlich löst.

Bonus 1: Authorisierung - Implementiere auch die Möglichkeit teile der Applikation nur Usern mit bestimmten Rollen zur Verfügung zu stellen (Stichwort: Admin-User)

Bonus 2: Recherchiere potentielle Angriffsvektoren für Web-Applikationen

Abgabe: Protokoll + src-code (per github repo auch möglich: wenn private-repo, dann invite an: <https://github.com/Alexander-Lenz-HTL>)

GitHub Repo Initialisieren

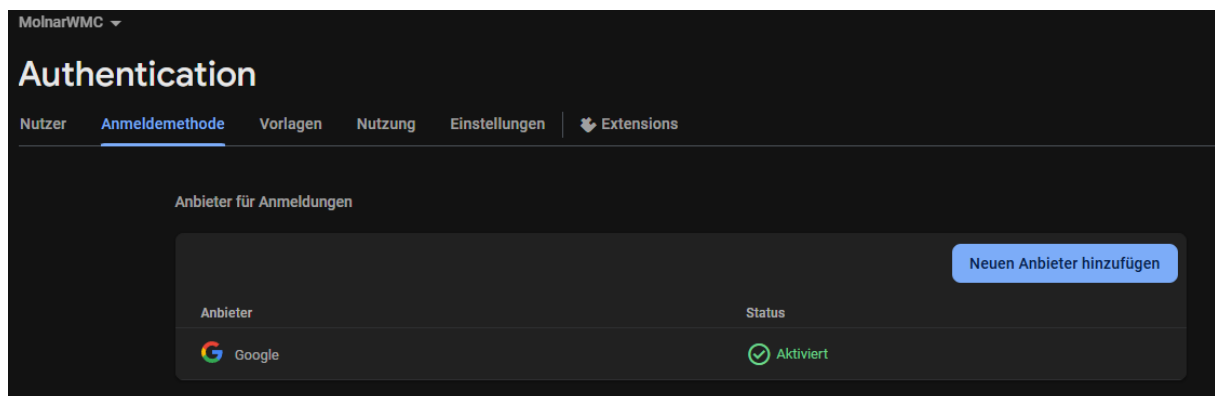


Name	Änderungsdatum	Typ	Größe
.git	14.03.2025 14:34	Dateiordner	
backend	14.03.2025 14:30	Dateiordner	
docs	14.03.2025 14:37	Dateiordner	
frontend	14.03.2025 14:27	Dateiordner	

In der Firebase Console Anmelden und Projekt anlegen



Authentifizierung mit Google aktivieren



Firebase zu meiner Web-App hinzufügen

1. npm install firebase
2. Initialisieren:

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AlzaSyBK9oDCepHIK2dIzFLNHwBZQGcowQrTkbl",
  authDomain: "molnarwmc.firebaseio.com",
  projectId: "molnarwmc",
  storageBucket: "molnarwmc.firebaseio.com",
  messagingSenderId: "1083387675634",
  appId: "1:1083387675634:web:9accb1038caec2d3fea16b",
```

```
    measurementId: "G-G8PYB79M0S"
  };

  // Initialize Firebase
  const app = initializeApp(firebaseConfig);
  const analytics = getAnalytics(app);
```

Firebase Config im Projekt:

```
TS environment.ts U X
frontend > MolnarWMC-Web > src > environments > TS environment.ts > ...
1  export const environment = {
2    production: false,
3    firebaseConfig: {
4      apiKey: "AIzaSyBK9oDCepHIK2dIzFLNHwBZQGcowQrTkBI",
5      authDomain: "molnarwmc.firebaseio.com",
6      projectId: "molnarwmc",
7      storageBucket: "molnarwmc.firebaseio.com",
8      messagingSenderId: "1083387675634",
9      appId: "1:1083387675634:web:9accb1038caec2d3fea16b"
10   }
11 };
12
```

@angular/fire installieren im powershell

Frontend Konfigurieren:

...MolnarWMC-Web/src/app/services/auth.service.ts

```
import { Injectable, inject } from '@angular/core';
import { Auth, signInWithPopup, GoogleAuthProvider, signOut, User } from
 '@angular/fire/auth';
import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  private auth = inject(Auth);
  private userSubject = new BehaviorSubject<User | null>(null);
```

```
user$ = this.userSubject.asObservable();

constructor() {
  this.auth.onAuthStateChanged((user) => {
    this.userSubject.next(user);
  });
}

async loginWithGoogle() {
  try {
    const provider = new GoogleAuthProvider();
    const result = await signInWithPopup(this.auth, provider);
    this.userSubject.next(result.user);
    console.log('User:', result.user);
    return result.user;
  } catch (error) {
    console.error('Login-Fehler:', error);
    return null;
  }
}

async logout() {
  await signOut(this.auth);
  this.userSubject.next(null);
}
}
```

Diese Datei kümmert sich um das Anmelden und Abmelden mit Google.

Sie verwendet Firebase Authentication, um den Benutzer einzuloggen und den Status zu speichern.

... MolnarWMC-Web/src/app/app.component.ts

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AuthService } from '../services/auth.service';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
```

```
export class AppComponent {
  user$;

  constructor(private authService: AuthService) {
    this.user$ = this.authService.user$;
  }

  login() {
    this.authService.loginWithGoogle();
  }

  logout() {
    this.authService.logout();
  }
}
```

Diese Datei steuert das Verhalten der Hauptseite der Anwendung.

Sie ruft den AuthService auf, um den Benutzer einzuloggen oder auszuloggen.

Außerdem speichert sie den aktuellen Benutzer, damit er in der Oberfläche angezeigt werden kann.

...MolnarWMC-Web/src/app/app.component.html

```
<div class="container">
  <h1>Login mit Google</h1>

  <ng-container *ngIf="user$ | async as user; else loggedOut">
    <div class="user-info">
      <img [src]="user.photoURL" alt="Profilbild">
      <p>Willkommen, {{ user.displayName }}</p>
    </div>
    <button class="logout-btn" (click)="logout()">Logout</button>
  </ng-container>

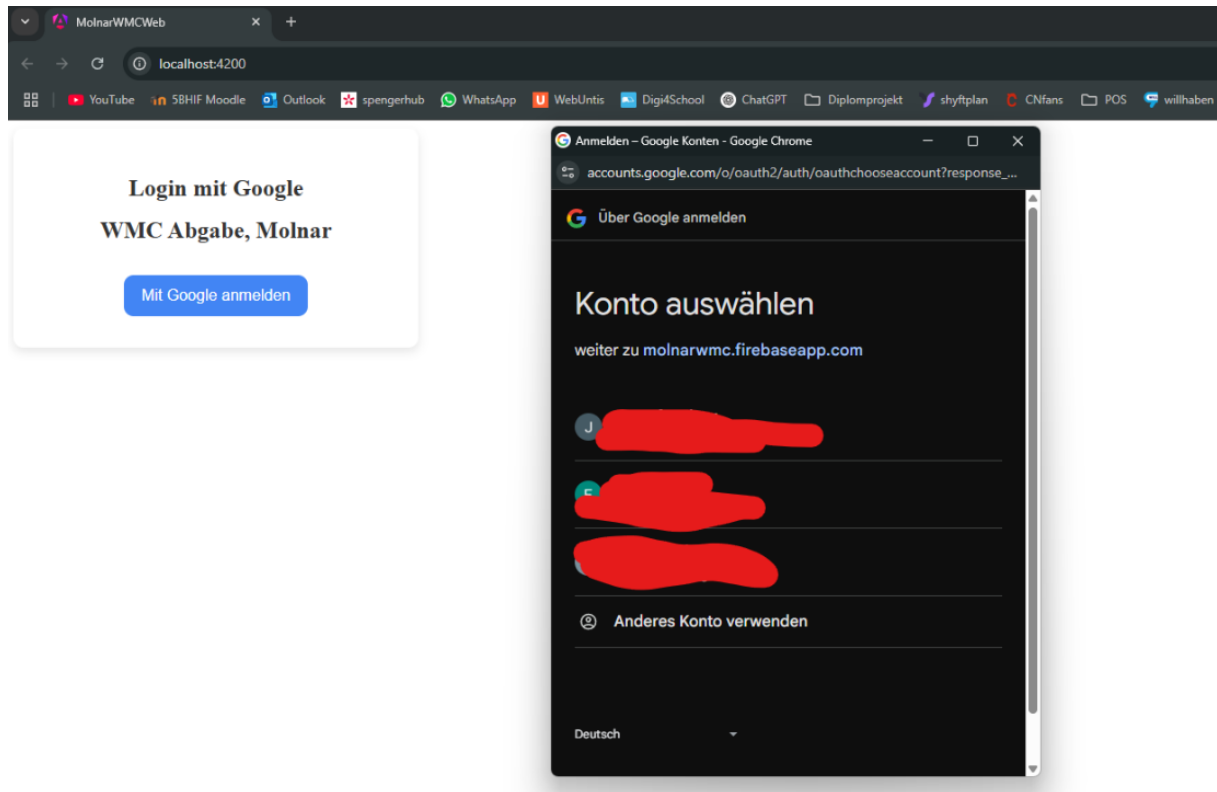
  <ng-template #loggedOut>
    <button (click)="login()">Mit Google anmelden</button>
  </ng-template>
</div>
```

Diese Datei zeigt die Login- und Logout-Buttons sowie die Benutzerinformationen an.

Wenn der Benutzer eingeloggt ist, sieht man seinen Namen und sein Profilbild.

Wenn er ausgeloggt ist, wird nur der "Mit Google anmelden"-Button angezeigt.

Frontend mit scss erweitern und dann fertig:



Angemeldet:

