Front End Development 1
Sesi 26

Angular Authentication & HTTP Request

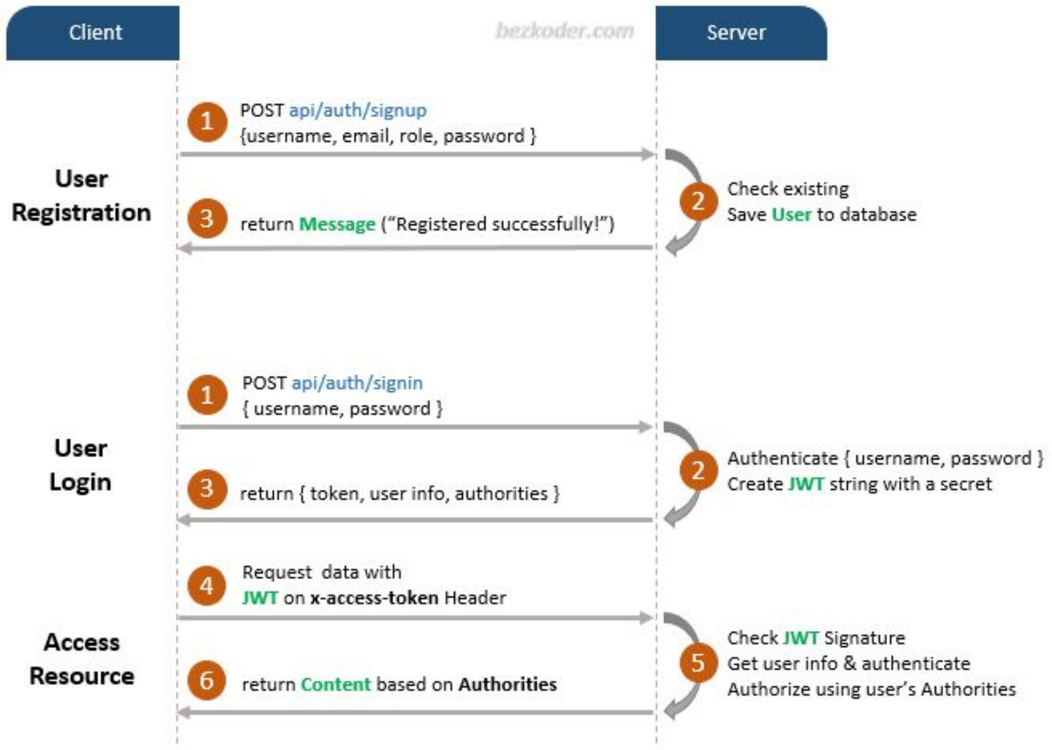
Angular Authentication

Untuk dapat membuat aplikasi yang secured, dibutuhkan sistem autentikasi user dengan menggunakan JSON Web Token (JWT) dan Web API.

Apa itu JWT?

JWT atau JSON Web Token merupakan token dalam bentuk string yang telah divalidasi dan digenerate oleh server. Token string ini yang membantu komunikasi antara client dan server. Apabila token invalid atau tidak disediakan, maka komunikasi client-server tidak dapat dilakukan.

Angular Authentication: Flow

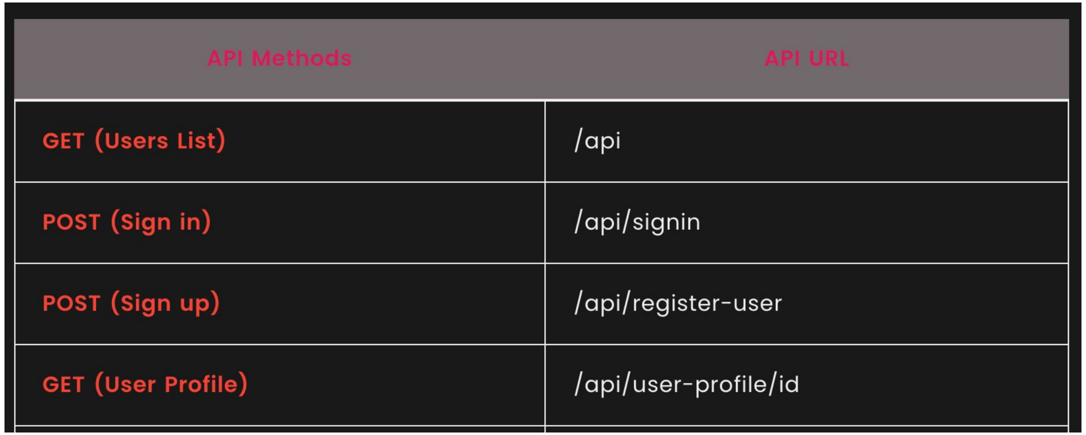


Client
bezkoder.com
Server
User Registration
1 POST api/auth/signup {username, email, role, password }
2 Check existing Save User to database
3 return Message ("Registered successfully!")
User Login
1 POST api/auth/signin { username, password }
2 Authenticate { username, password } Create JWT string with a secret
3 return { token, user info, authorities }
Access Resource
4 Request data with JWT on x-access-token Header
5 Check JWT Signature Get user info & authenticate Authorize using user's Authorities
6 return Content based on Authorities

Angular Authentication

Untuk Sesi ini, akan membuat fitur register, login, menampilkan user profile.

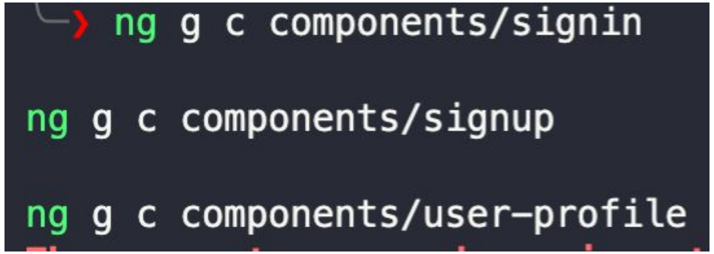
Berikut contoh endpoint server API yang digunakan untuk contoh sesi ini:



API Methods	API URL
GET (Users List)	/api
POST (Sign in)	/api/signin
POST (Sign up)	/api/register-user
GET (User Profile)	/api/user-profile/id

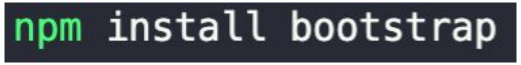
Angular Authentication: Generate Komponen & konfigurasi

Buatlah aplikasi baru dengan angular routing dan generate komponen-komponen berikut:

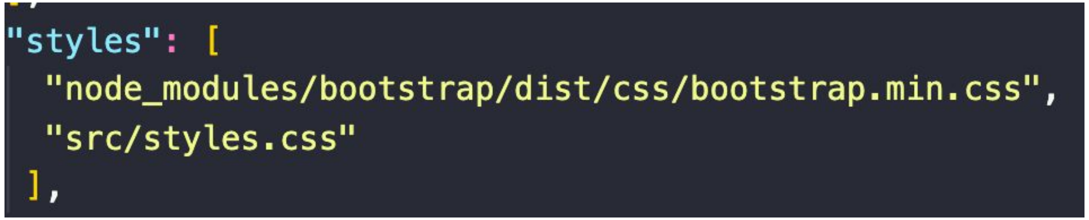


```
╰─❯ ng g c components/signin

ng g c components/signup

ng g c components/user-profile
```

Install Bootstrap dan masukkan ke angular.json



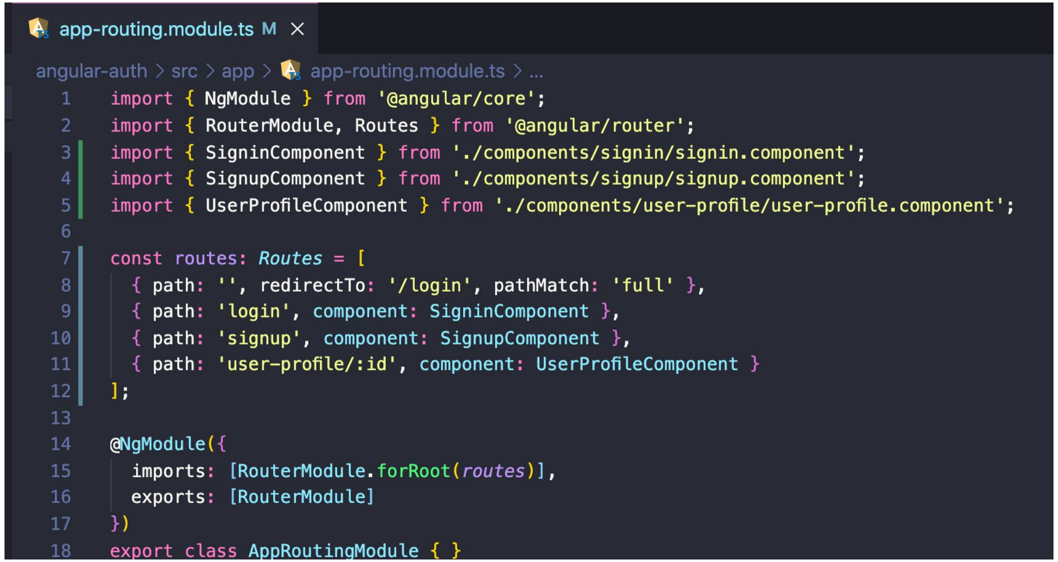
```
npm install bootstrap
```



```
"styles": [
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "src/styles.css"
 ],
```

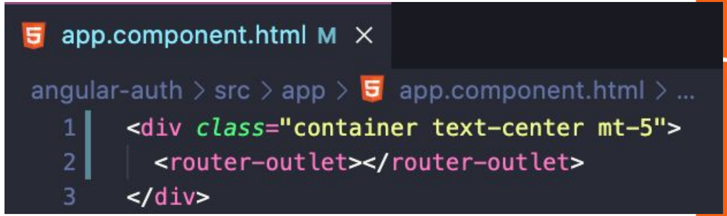
Angular Authentication: Generate Komponen & konfigurasi

Daftarkan routing beserta komponen di file app-routing.module.ts


app-routing.module.ts M ×

angular-auth > src > app > app-routing.module.ts > ...

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { SigninComponent } from './components/signin/signin.component';
import { SignupComponent } from './components/signup/signup.component';
import { UserProfileComponent } from './components/user-profile/user-profile.component';

const routes: Routes = [
  { path: '', redirectTo: '/login', pathMatch: 'full' },
  { path: 'login', component: SigninComponent },
  { path: 'signup', component: SignupComponent },
  { path: 'user-profile/:id', component: UserProfileComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```


app.component.html M ×

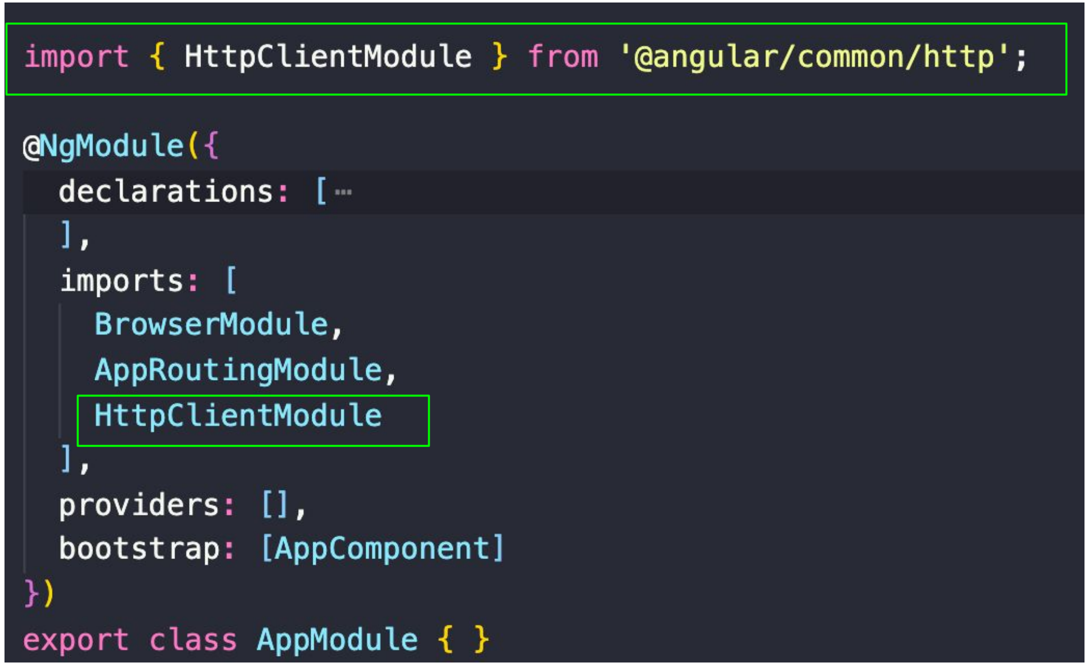
angular-auth > src > app > app.component.html > ...

```
<div class="container text-center mt-5">
  <router-outlet></router-outlet>
</div>
```

HACKTIV8

Angular HTTPClient

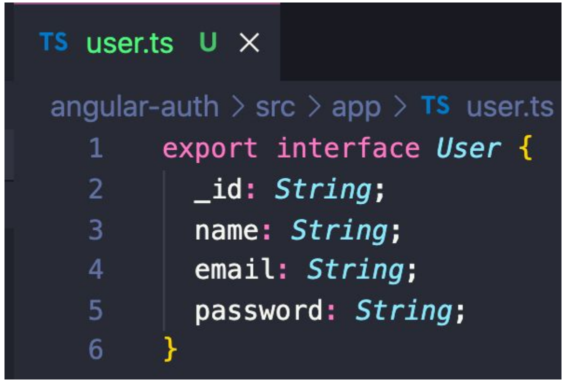
Untuk dapat menghandle REST APIs, perlu mengimport **HttpClientModule** didalam **app.module.ts**



```
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [⋯
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Angular Authentication: Generate Auth Service

Buatlah 1 file baru bernama user. Dimana User berisi properti _id, name, email dan password.

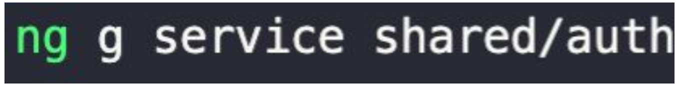


TS user.ts U ×

angular-auth > src > app > TS user.ts

```typescript
export interface User {
  _id: String;
  name: String;
  email: String;
  password: String;
}
```

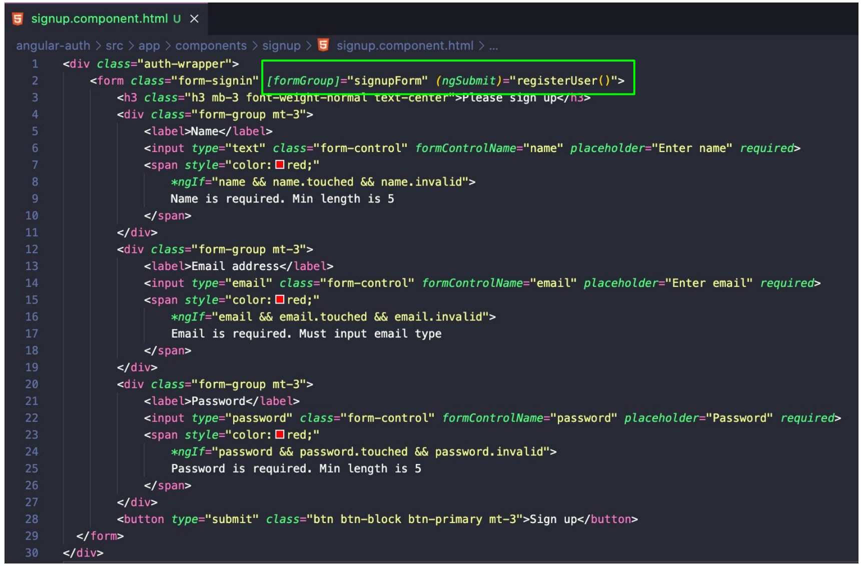
Generate 1 service baru bernama auth yang akan diletakkan di dalam folder shared



```
ng g service shared/auth
```

HACKTIV8

Angular Authentication: SignUp template

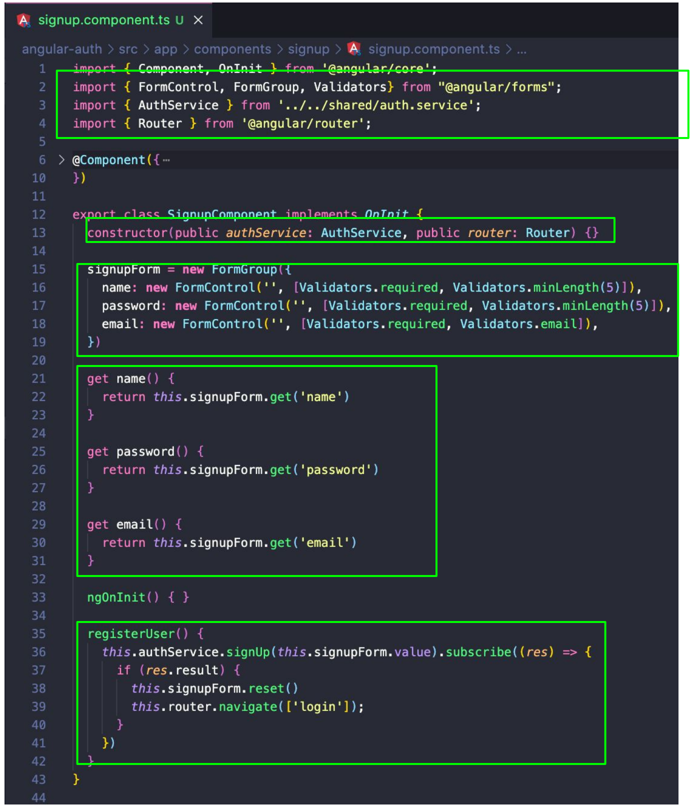

signup.component.html U
angular-auth > src > app > components > signup > signup.component.html > ...

```html
<div class="auth-wrapper">
    <form class="form-signin" [formGroup]="signupForm" (ngSubmit)="registerUser()">
        <h3 class="h3 mb-3 font-weight-normal text-center">Please sign up</h3>
        <div class="form-group mt-3">
            <label>Name</label>
            <input type="text" class="form-control" formControlName="name" placeholder="Enter name" required>
            <span style="color:red;"
                *ngIf="name && name.touched && name.invalid">
                Name is required. Min length is 5
            </span>
        </div>
        <div class="form-group mt-3">
            <label>Email address</label>
            <input type="email" class="form-control" formControlName="email" placeholder="Enter email" required>
            <span style="color:red;"
                *ngIf="email && email.touched && email.invalid">
                Email is required. Must input email type
            </span>
        </div>
        <div class="form-group mt-3">
            <label>Password</label>
            <input type="password" class="form-control" formControlName="password" placeholder="Password" required>
            <span style="color:red;"
                *ngIf="password && password.touched && password.invalid">
                Password is required. Min length is 5
            </span>
        </div>
        <button type="submit" class="btn btn-block btn-primary mt-3">Sign up</button>
  </form>
</div>
```

Buat template signup form di file signup.component.

Form disini menggunakan reactive-form.

Angular Authentication: SignUp class component

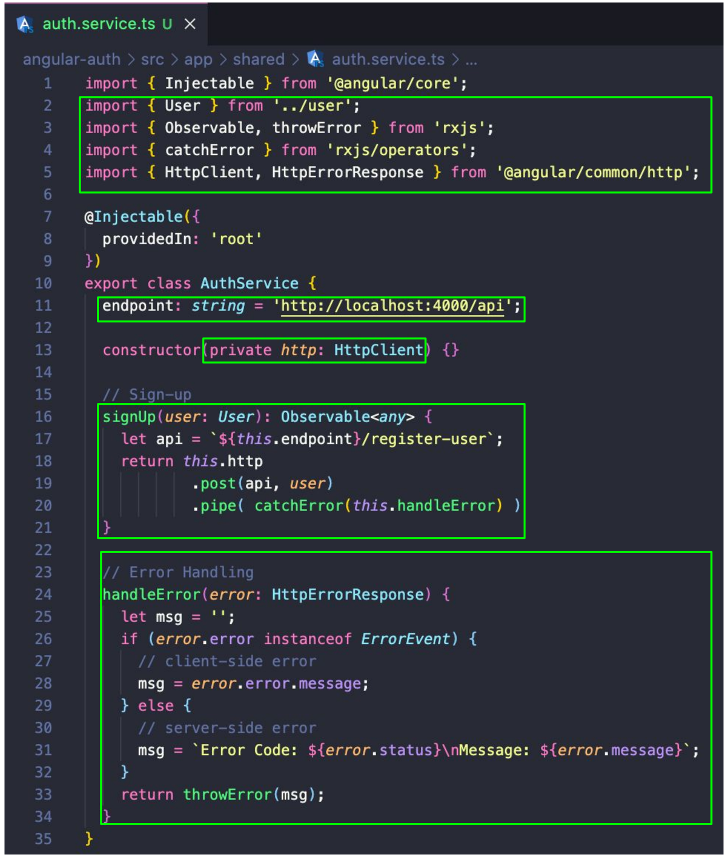

signup.component.ts U
angular-auth > src > app > components > signup > signup.component.ts > ...

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators} from "@angular/forms";
import { AuthService } from '../../shared/auth.service';
import { Router } from '@angular/router';

@Component({…
})

export class SignupComponent implements OnInit {
  constructor(public authService: AuthService, public router: Router) {}

  signupForm = new FormGroup({
    name: new FormControl('', [Validators.required, Validators.minLength(5)]),
    password: new FormControl('', [Validators.required, Validators.minLength(5)]),
    email: new FormControl('', [Validators.required, Validators.email]),
  })

  get name() {
    return this.signupForm.get('name')
  }

  get password() {
    return this.signupForm.get('password')
  }

  get email() {
    return this.signupForm.get('email')
  }

  ngOnInit() { }

  registerUser() {
    this.authService.signUp(this.signupForm.value).subscribe((res) => {
      if (res.result) {
        this.signupForm.reset()
        this.router.navigate(['login']);
      }
    })
  }
}
```

Dalam file **signup.component.ts**,

1. Import authservice dan class-class yang dibutuhkan
2. Masukkan authService dan router di params contructor
3. Buatlah signUpForm yang mempunyai name, password dan email
4. Getter name, password dan email untuk menampilkan validasi di templaye
5. Method **registerUser()** akan memanggil method **signUp authservice** yang menerima parameter value signUpform. Dan menavigasi ke /login jika signup berhasil.

Angular Authentication: SignUp Service



auth.service.ts U ×

angular-auth > src > app > shared > auth.service.ts > ...

```typescript
import { Injectable } from '@angular/core';
import { User } from '../user';
import { Observable, throwError } from 'rxjs';
import { catchError } from 'rxjs/operators';
import { HttpClient, HttpErrorResponse } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  endpoint: string = 'http://localhost:4000/api';

  constructor(private http: HttpClient) {}

  // Sign-up
  signUp(user: User): Observable<any> {
    let api = `${this.endpoint}/register-user`;
    return this.http
        .post(api, user)
        .pipe( catchError(this.handleError) )
  }

  // Error Handling
  handleError(error: HttpErrorResponse) {
    let msg = '';
    if (error.error instanceof ErrorEvent) {
      // client-side error
      msg = error.error.message;
    } else {
      // server-side error
      msg = `Error Code: ${error.status}\nMessage: ${error.message}`;
    }
    return throwError(msg);
  }
}
```

Dalam file **auth.service.t**s,

1. Import interface User, Observable dan throwerror dari rxjs, catchError dan class HttpClient dan HttpErrorResponse.
2. Buat properti **endpoint** yang mengarah ke localhost:4000/api
3. Masukan parameter constructor http yang bertipe HttpClient
4. **SignUp** method menerima parameter user, yang akan dikirim ke server method POST dengan path /register-user. Untuk catch error, observable menyediakan **.pipe()**
5. **handleError** method berfungsi untuk meng-catch error dari request dan melakukan **throwError**.