Angular Auth

Authentication & Authorization

Nikos Anifantis
Full Stack Software Engineer
@nikosanif

*ngThess

Nikos Anifantis

Fullstack Software Engineer

- Fullstack Software Engineer @TurintechAI

  TURINTECH

- I write stuff at:

  nikosanif.medium.com

  dev.to/nikosanif

- How to reach me:

  @nikosanif

  /nikosanifantis

  /nikosanif

Agenda

- What is user Authentication & Authorization?
- JWT-based Auth in a Nutshell
- Auth in Angular
- POC application: nikosanif/angular-authentication

Authentication & Authorization

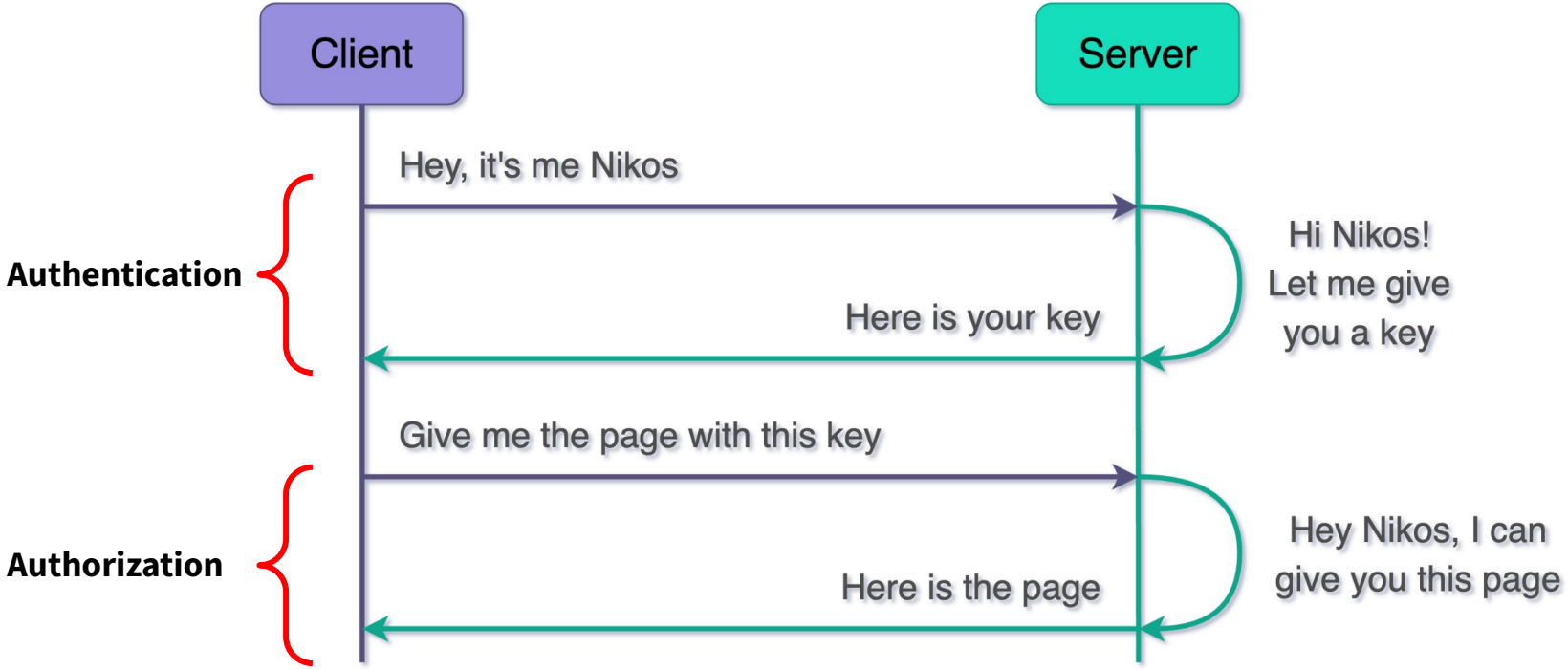
Authentication

- Verifies who the user is
- Defined by user (e.g. passwords, biometrics)
- Visible to and partially changeable by the user

Authorization

- Grants or denies permissions
- Defined by security teams (e.g. admins)
- It isn't visible to or changeable by the user

Authentication & Authorization - Example



Authorization Strategies

Session-based - Session ID

Token-based - JSON Web Token

What is JWT?

- **JSON Web Tokens** are an open, industry standard RFC 7519 method for representing claims securely between parties as a JSON object.

- The information can be **verified** and **trusted** because it is digitally signed.

- JWTs can be signed using a **secret** (with HMAC algorithm) or **public/private key** pair using RSA or ECDSA.

Why should we use JWT?

- **Security** - Securely transmitting information between parties using public/private key pairs.
- **Ease** - Ease of client side processing of the JWT on multiple platforms.
- **Compact** - Because of its size, it can be sent through URLs, POST parameter, or inside an HTTP header.
- **Self-Contained** - The payload contains all the required information about the user to avoid querying the server more than once.

How do JWT work?

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6
IkpXVCJ9.eyJzdWIiOiIxMjM0NTY
3ODkwIiwibmFtZSI6IkpvaG4gRG9
lIiwiaWF0IjoxNTE2MjM5MDIyfQ.
SflKxwRJSMeKKF2QT4fwpMeJf36P
Ok6yJV_adQssw5c
```

Jwt.io

Decoded

HEADER:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

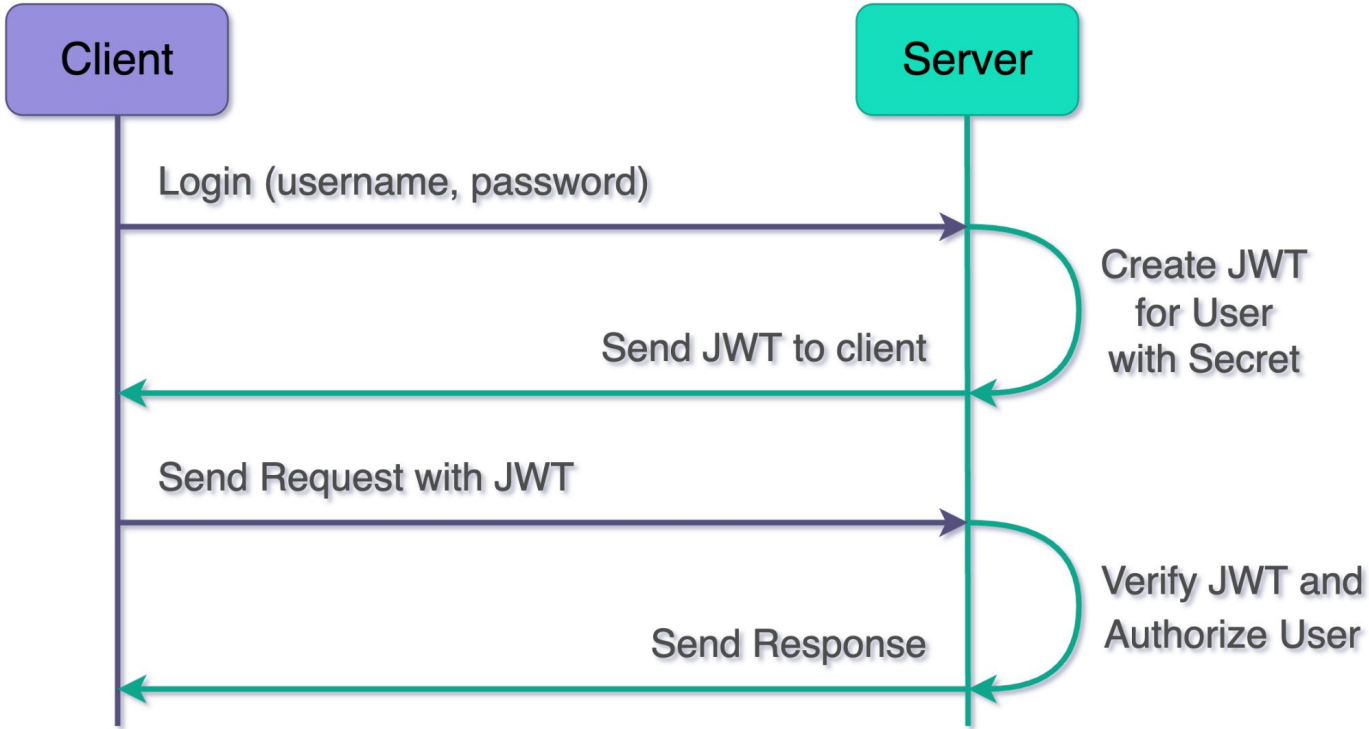
PAYLOAD:

```
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

JWT to verify the authenticity of a user



Blacklisting JWT

- What if a JWT is **stolen** or we want to **invalidate** it?

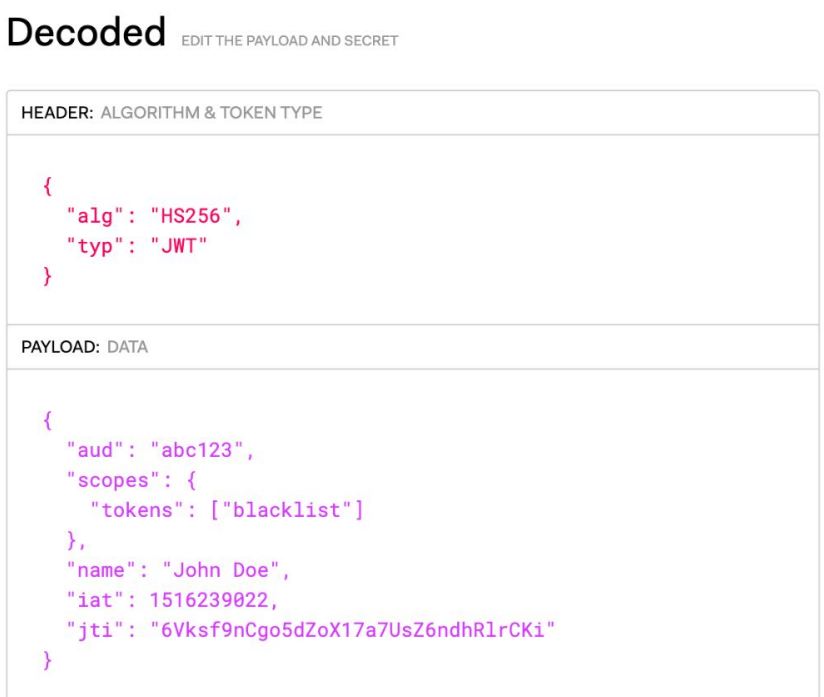
  We should add a JWT token to a “Blacklist”.

Blacklisting JWT

- Who should be able to **revoke** JWTs?

  Other JWTs. But…

  - It has the permission to do it - `scopes`
  - Each JWT must individually be identified - `jti` & `aud`



Auth in Angular

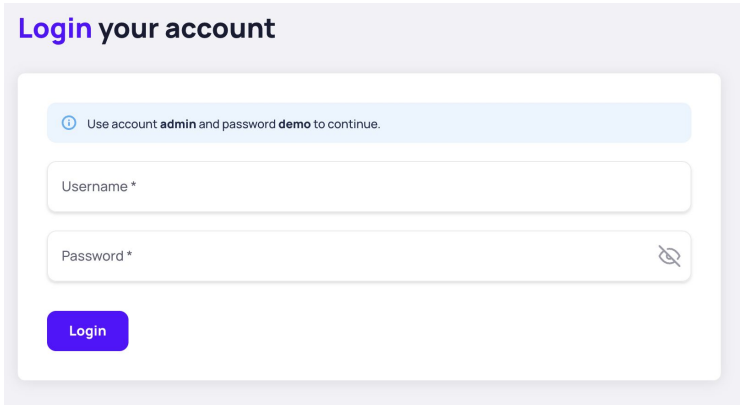
Examples & Source code

Prerequisites - Server API

- **API Endpoints** (HTTP protocol)
  - Authentication
    - Login
    - Logout (revoke tokens)
  - Users
    - Get info about myself (me)
    - Get info about a user by ID

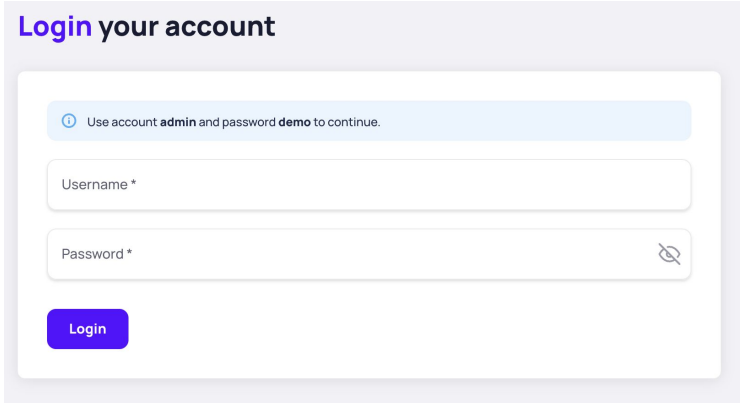
Login & Store access token

```
@Component({ /*...*/ })
export class LoginComponent {

  readonly loginForm = new FormGroup({
    username: new FormControl('', {
      validators: [Validators.required],
    }),

    password: new FormControl('', {
      validators: [Validators.required],
    }),
  });

  submit() {
    const { username, password } = this.loginForm.value;
    this.authService.login(username, password);
  }
}
```



Login & Store access token

```
@Injectable()
export class AuthService {
  login(username: string, password: string): Observable<AccessData> {

    return this.http
      .post<AccessData>(`.../api/auth/login`, { username, password })
      .pipe(
        tap(data => {
          // save tokens in local storage
          this.tokenStorageService.saveTokens(data.access_token, data.refresh_token);
        })
      );
  }
}
```



HTTP Interceptors

```ts
@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  // ...

  intercept(
    req: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    const accessToken = this.tokenStorageService.getAccessToken();

    if (accessToken) {
      req = req.clone({
        setHeaders: { Authorization: `Bearer ${accessToken}` },
      });
    }

    return next.handle(req).pipe(s => this.handleErrors(s));
  }

  // ...
}
```

Auth Route Guards

```ts
@Injectable({ providedIn: 'root' })
export class AuthGuardService implements CanActivate {

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): Observable<boolean> {
    return this.store.select(selectIsLoggedIn).pipe(
      take(1),
      tap(isLoggedIn => {
        if (!isLoggedIn) {
          this.router.navigate(['/login'], { queryParams: { returnUrl: state.url } });
        }
      })
    );
  }

}
```

```ts
const routes: Routes = [
  // ...
  {
    path: 'secured-feat',
    canActivate: [AuthGuardService],
    loadChildren: () => import('./secured-feat.module').then(
  }
  // ...
];
```

Auth Route Guards

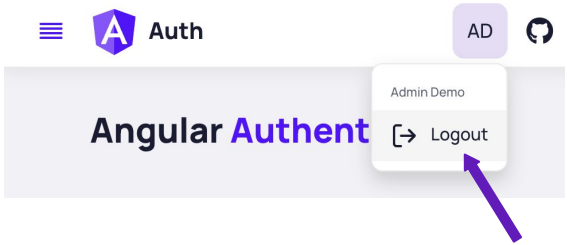
```
const routes: Routes = [
  // ...
  {
    path: 'login',
    component: LoginComponent,
    canActivate: [NoAuthGuardService],
  }
  // ...
];
```

```
@Injectable({ providedIn: 'root' })
export class NoAuthGuardService implements CanActivate {

  canActivate(): Observable<boolean> {
    return this.store.select(selectIsLoggedIn).pipe(
      take(1),
      map(isLoggedIn => {
        if (isLoggedIn) {
          this.router.navigateByUrl('/');
        }

        return !isLoggedIn;
      })
    );
  }
}
```

Logout & Blacklisting token

```
@Injectable()
export class AuthService {
  // ...

  logout(): Observable<void> {
    return this.http
      .get<void>(`../api/auth/logout`)
      .pipe(finalize(() => this.tokenStorageService.removeTokens()));
  }
}
```

Revoke current token



POC Application

nikosanif / angular-authentication

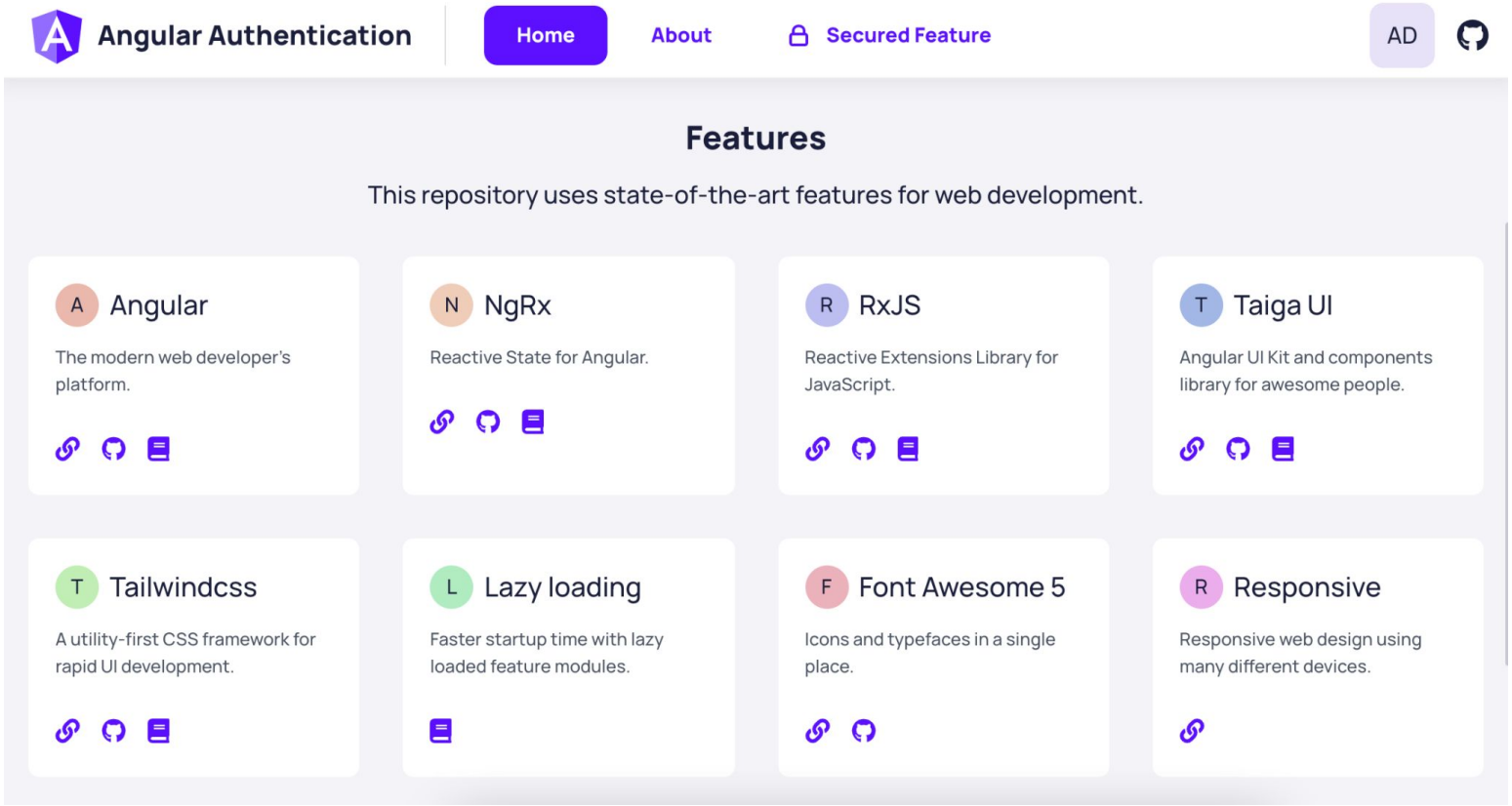
An Angular application that demonstrates best practices for user authentication.

Angular Authentication

An Angular application that demonstrates best practices for user authentication flow.

By @nikosanif

nikosanif / angular-authentication



What's next?

- Manage user roles / permissions
- Handle expiration dates
- "Remember me" functionality
- Migrate to Nrwl/Nx
- Implement backend server (?)
- … **your** features! Any contribution or feature request is welcome 😀

References

- Angular Authentication, https://github.com/nikosanif/angular-authentication
- JWT.IO, https://jwt.io/
- JSON Web Tokens - Prashant Walke, https://www.slideshare.net/PrashantWalke3/json-web-token-jwt-137077119
- Authentication vs. Authorization - Auth0, https://auth0.com/docs/get-started/authentication-and-authorization
- Adding JWT API Keys to a DenyList - Auth0, https://auth0.com/blog/denylist-json-web-token-api-keys/

Thank you!

Nikos Anifantis

@nikosanif

/nikosanifantis

/nikosanif

***ngThess**