УДК 004.021

ЗАЩИТА ИНФОРМАЦИИ ПОЛЬЗОВАТЕЛЯ НА КЛИЕНТСКОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ ПОСРЕДСТВОМ JWT ТОКЕНА И ВСТРОЕННЫХ МЕХАНИЗМОВ ANGULAR 5

П.Д. ТАЛАЙКО
(Представлено: канд. тех. наук, доц. И. Б. БУРАЧЁНОК)

Рассмотрены механизмы фреймворка Angular 5, которые позволяют проверить права доступа пользователя к тем или иным данным и разрешить или запретить доступ к определенным страницам веб-приложения.

В современных веб-приложениях на клиентских частях зачастую необходимо разграничить доступ пользователей к данным и запретить просмотр некоторых страниц для одних и разрешить для вторых.

В представленной статье расскажем о механизмах Angular 5 решающих задачи разграничения доступа.

Angular представляет фреймворк от компании Google для создания клиентских приложений. Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений. В этом плане Angular является наследником другого фреймворка AngularJS. В то же время Angular это не новая версия AngularJS, а принципиально новый фреймворк.

Angular 5 предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и так далее.

Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript. Поэтому перед началом работы рекомендуется ознакомиться с основами данного языка.

Приложения на Angular можно описать с помощью таких языков как Dart или JavaScript. Однако основным языком для Angular все-таки является TypeScript.

Server написан на фреймворках Spring Boot, Spring Data и Spring Security, реализует авторизацию и аутентификацию пользователей благодаря технологии JWT.

JWT (JSON Web Token) – это открытый стандарт RFC 7519 для создания токенов доступа, основанный на JSON формате [4]. Он, как правило, используется для передачи данных авторизации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для обращения к ресурсам веб-приложения.

При вводе логина и пароля на клиентской части, данные в формате JSON отправляются на сервер POST запросом на определённый URL для получения токена JWT токена, который хранится у пользователя локально в браузере в localStorage, при совершении какого-либо действия клиентское приложение передает JWT токен в заголовке Authorization, в котором хранится информация о его ролях, если пользователь локально изменил содержание токена, то при прохождении обработки на сервере и сравнения с секретным ключом он будет недействителен, данное гарантируется спецификацией JWT.

На клиентской части необходимо написать сервисы для отправления HTTP запросов на сервер. В Angular 5 таким сервисом является HttpClient, который предоставляет основные методы для общения по HTTP протоколу [2].

Сервер реализует REST API и разграничивает доступ пользователей к определенным функциям, а при возникновении исключительных ситуаций возвращает соответственный код ошибки. Задача клиента корректно обработать эти ошибки и сообщить о их наличии.

Для установления заголовка Authorization в HTTP запрос нужно воспользоваться классом HttpHeaders. Неправильно устанавливать для всех запросов HTTP свой заголовок. В свою очередь можем воспользоваться HttpInterceptor.

HttpInterceptor – это interface Angular 5 позволяющий обработать HTTP запрос перед его отправкой на сервер [3]. Благодаря этому интерфейсу можем манипулировать глобально запросами с клиента.

После того как пользователь получил свой личный токен идентификации в системе, он устанавливается в localStorage (локальное хранилище браузера). При обращении к серверу в заголовок с помощью класса, реализующего интерфейс HttpInterceptor, можно установить токен, и при после обработать ответ.

Пример класса AuthInterceptor, реализующего интерфейс HttpInterceptor приведён в листинге 1.

```
@Injectable()
export class AuthInterceptor implements HttpInterceptor {

  private apiTokenName = 'token';

  constructor(private tokenStorage: TokenStorageService, private router: Rout-
er, private userStore: AuthService) {
  }

  intercept(req: HttpRequest<any>, next: HttpHandler): Observa-
ble<HttpEvent<any>> {
    const token = TokenStorageService.token();

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

    return next.handle(req).pipe(
      tap(
        (response: HttpResponse<any>) => {
          if (response.body && response.body.token) {
            this.userStore.setAuthentication(true);
            TokenStorageService.setToken(response.body.token);
            this.router.navigateByUrl('/');
          }
        }, (error: HttpErrorResponse) => {
          if (error instanceof HttpErrorResponse) {
            switch (error.status) {
              case 401: {
                this.router.navigateByUrl('/auth');
                this.userStore.setAuthentication(false);
              }
                break;
              case 403: {
                this.router.navigateByUrl('/auth');
              }
                break;
            }
          } else {
            return Observable.throw(error);
          }
        },
      ),
    );
  }
}
```

Листинг 1. – Реализация класса AuthInterceptor

Из кода можно видеть, что при возвращении ошибки с кодом 401 (Unauthorization) и 403 (Forbiden) происходит перенаправление на страницу авторизации. При успешном доступе к ресурсам происходит установка статуса this.userStore.setAuthentication(true).

В Angular 5 существует ещё один полезный механизм, позволяющий ограничивать доступ к сайтам по тем или иным причинам и называется Guard [5]. Guard'ы позволяют по определенному условию давать доступ к сайту или нет. Реализация Guard изображена в листинге 2.

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

  constructor(private auth: AuthService, private route: Router) {

  }
```

```
      canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot):
Observable<boolean> | Promise<boolean> | boolean {
        if (!this.auth.isAuthentication()) {
          this.route.navigateByUrl('/auth');
          return false;
        }
        return true;
      }
    }
```

Листинг 2. – Реализация класса MainGuard

Guard'ы устанавливаются на ссылку страницы (route) приложения и при переходе между компонентов вызываются и разрешают или запрещают доступ. В листинге можем видеть, что класс MainGuard реализует интерфейс CanActivate, имеющий один метод canActivate. В данном примере показано что если пользователь авторизован, то запрещать переход на страницу авторизации, а при выходе и изменении статуса пользователя на неавторизованный, доступ на страницу авторизации будет разрешён.

В данной статье представлены механизмы для обеспечения удобства работы с авторизацией и аутентификацией пользователя и доступа его к определённым компонентам веб-приложения. А также приведены основы клиентской работы с токенами JWT.

ЛИТЕРАТУРА

1. Token-Based Authentication [Электронный ресурс] // Git hub gist. – Режим доступа: https://gist.github.com/zmts/802dc9c3510d79fd40f9dc38a12bccfc/. – Дата доступа: 09.09.18.
2. HTTP [Электронный ресурс] // Angular. – Режим доступа: https://angular.io/tutorial/toh-pt6/. – Дата доступа: 10.09.18.
3. Angular Authentication: Using the Http Client and Http Interceptors [Электронный ресурс] / Medium. – Режим доступа: https://medium.com/@ryanchenkie_40935/angular-authentication-using-the-http-client-and-http-interceptors-2f9d1540eb8/. – Дата доступа: 10.09.18.
4. JWT [Электронный ресурс] // JWT. – Режим доступа: https://jwt.io/. – Дата доступа: 10.09.18.
5. Routing & Navigation [Электронный ресурс] // Angular. – Режим доступа: https://angular.io/guide/router/. – Дата доступа: 10.09.18.