

47. Авторизация

Цель:

Познакомиться со следующими особенностями авторизации:

- виды авторизаций

- OAuth 2

- JWT tokens

Авторизация/аутентификация:

**Авторизация:** предоставление пользователю прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

**Аутентификация:** процедура проверки легальности пользователя или данных, например, проверки соответствия введенного пользователем пароля к учётной записи паролю в базе данных.

Виды авторизаций:

1. Сессии

2. Токены

3. Беспарольный вход

4. Единая точка входа (Single Sign On, SSO) 5. Аутентификация в соцсетях

Сессии:

1. Пользователь вводит в браузере своё имя и пароль, после чего клиентское приложение отправляет запрос на сервер.

2. Сервер проверяет пользователя, аутентифицирует его, затем в ответе клиентскому приложению шлет уникальный пользовательский токен. При этом этот токен сохраняется в память или базу данных.

3. Клиентское приложение сохраняет токены в куках и отправляет их при каждом последующем запросе.

4. Сервер получает каждый запрос, требующий аутентификации, с помощью токена проверяет пользователя и возвращает запрошенные данные.

5. Когда пользователь выходит, клиентское приложение удаляет его токен, поэтому все последующие запросы от этого клиента становятся неавторизованными.

Токены:

1. Пользователь вводит имя и пароль, после чего клиентское приложение отправляет запрос на сервер.

2. Сервер проверяет их и возвращает токены (JWT), который может содержать метаданные вроде user\_id, роль, разрешения и т. д.

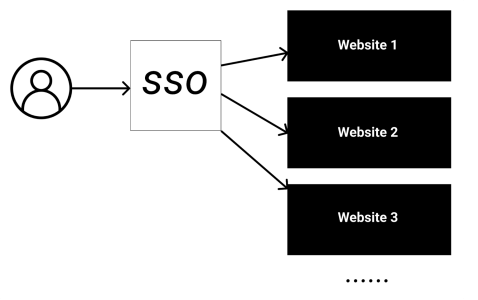
3. Токен хранится на клиентской стороне, чаще всего в локальном хранилище, но может лежать и в хранилище сессий или кук.

4. Последующие запросы к серверу обычно содержат этот токен в качестве дополнительного заголовка авторизации в виде *Bearer {JWT}*.

5. Сервер расшифровывает JWT, если токен верный, сервер обрабатывает запрос. 6. Когда пользователь выходит из системы, токен на клиентской стороне удаляется.

Беспарольная аутентификация:

1. Беспарольная аутентификация — это способ входа и аутентификации пользователей без ввода паролей. 2. Вместо ввода почты/имени и пароля пользователи вводят только свою почту. Приложение отправляет на этот адрес одноразовую ссылку, пользователь по ней кликает и автоматически входит в клиентское приложение авторизованным пользователем.

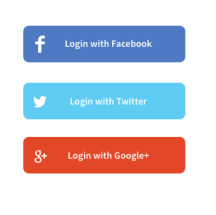
Единая точка входа (Single Sign On, SSO):

SSO:

1. Пользователь входит в один из сервисов Google. 2. Пользователь получает сгенерированную в Google Accounts куку . 3. Пользователь идёт в другой продукт Google.

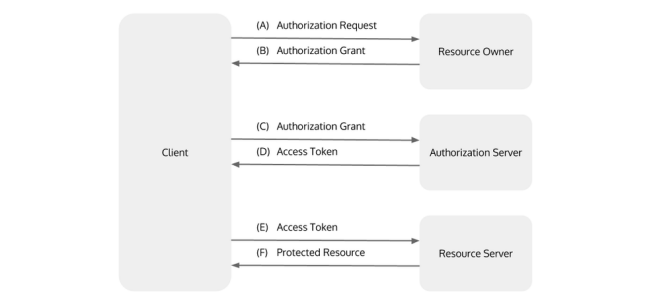
4. Пользователь снова перенаправляется в Google Accounts. 5. Google Accounts видит, что пользователю уже присвоена кука, и перенаправляет пользователя в запрошенный продукт.

Аутентификация в соцсетях:

Аутентификацией в соцсетях — это тоже SSO 

Есть возможность аутентифицировать пользователей по их аккаунтам в соцсетях. Тогда пользователям не придётся проходить аутентификацию отдельно в другом приложении.

OAuth 2:



OAuth 2. Преимущества:

● Обращение к ресурсам происходит по HTTP/HTTPS с указанием токена в заголовках. Это позволяет использовать OAuth практически на любых платформах: мобильных и десктоп приложениях, сайтах, и даже в плагинах для браузеров.

● Возможность авторизации пользователя.

● Популярность - большинство компаний используют его в своих API. ● Простота реализации

● Большое количество документации, статей и ресурсов ● Наличие готовых решений, которые можно изменять под свои нужды

OAuth 2. Минусы:

● Нет единого установленного формата, вследствие чего на каждый сервис нужно иметь отдельную реализацию. ● При аутентификации иногда приходится делать дополнительные запросы для получения даже минимальной информации о пользователе. Решается использованием jwt токена, но далеко не все сервисы его поддерживают. ● При краже токена у злоумышленника на какое-то время появляется доступ к защищенным данным.

Tokens JWT:

**Токены доступа / access token** — это токены, с помощью которых можно получить доступ к защищенным ресурсам. Это короткоживущие, но многоразовые токены. В них может содержаться дополнительная информация, например, время жизни токена. Все зависит от желания разработчика.

**Рефреш токен / refresh token** — эти токены служат только для получение нового токена доступа. Они долгоживущие, но одноразовые.