**Как происходит процесс аутентификации:**

Пользователь вводит логин и пароль и отправляет запрос с этими данными на сторону нашего приложения, а приложение проверяет пользователя в базе данных, если пользователь есть, то аутентификация проходит успешно.

**OAuth 2:**

OAuth 2 – это протокол авторизации наших пользователей. Его суть заключается в том, что у нас будет сторонний authorization сервер, через который будет проходит авторизация пользователя. При этом, пользователь должен вводить логин и пароль именно от authorization сервер, а значит, нашему приложению не нужно будет хранить данные пользователя.

OAuth 2 **– не дает нам возможность получить данные пользователя**. Данные пользователя находятся на стороне сервера авторизации. Если нам нужно получить данные пользователя (напр. Для сохранения этих данных в базе данных) **мы должны использовать другой протокол OpenID connect.**

Этот протокол отличается от OAuth 2 только тем, что, когда пользователь успешно пройдет процедуру авторизации на стороне сервера авторизации, то помимо access token, refresh token, **он передает еще и id token** **в формате JWT**. И этот it token хранит данные пользователя.

JWT – это набор пользовательских данных в зашифрованном виде.

OAuth 2 – это про авторизацию

OpenID – это про аутентификацию.

**Authorization сервер** – Google, Yandex, Facebook и тд.

**Как работает OAuth 2 внутри:**

Внутри себя протокол авторизации OAuth 2 работает при помощи **OAuth 2 flow,** который имеет три вида:

1. Authorization flow – чаще всего используется
2. Implicit flow
3. Hybrid flow

**Сам же цикл работает следующим образом:**

Пользователь, кликает на один из Auth серверов (Google, VK), наше приложение это видит и перенаправляет (302 статус) пользователя на URL сервера авторизации, где будет проходить авторизация пользователя. Вместе с URL так же передается следующие параметры:

1. ClientID – Username клиента
2. redirect URI – URI нашего клиента.
3. scope – дает понять серверу авторизации, какой протокол используется. **OAuth или OpenID**
4. response type – говорит, какой именно flow использовать. Так как чаще всего используется Authorization flow, то response type будет code (response type = code).

После того, как пользователь был перенаправлен, сервер авторизации предоставляет свою форму логина. Пользователь вводит логин и пароль от сервера авторизации и, если пользователь ввел верные данные, то сервер авторизации **возвращает пользователю** 302 статус, также code, scope, state и **redirect URI нашего auth клиента**. Пользователь переходит поэтому URI и передает набор параметров (code, scope, state) auth клиенту.

Далее, auth клиент делает запрос к серверу авторизации, чтобы получить необходимые токены, этот запрос не видит наш пользователь. Если авторизация прошла успешно, то сервер авторизации возвращает 200 статус и набор токенов. Ну а потом, auth-клиент возвращает стартовую страницу нашего приложения пользователю.

**Набор токенов:**

Access token (некая строка, которая хранит роли пользователя)

Refresh token (Опционально)

**Implicit flow и его отличие от Authorization flow:**

Отличие заключается в том, что после того, как пользователь успешно прошел процедуру авторизации на стороне сервера авторизации, то сервер авторизации сразу возвращает набор токенов и 200 статус (не возвращает authorization code, как это было **с Authorization flow**). При этом, эти токены попадают нашему пользователю. **Implicit flow** используется вместе с JavaScript.