How to access user related data through PeaKass Web API

With PeaKass Web API, external applications retrieve PeaKass content such as data and events.

To access user-related data through the Web API, you must authorize an application to access that particular information. In this tutorial, you'll create a simple server-side application using Node.js and JavaScript, and learn how to:

1. Prepare your environment
2. Set up your PeaKass account
3. Register an application with PeaKass
4. Authorize users
5. Get authorization to access user data
6. Retrieve data from a Web API endpoint

Note: The complete source code of the application you'll create in this tutorial is available in PeaKass GitHub repository.

In this example, you'll retrieve data from the **Web API /me endpoint**. This endpoint includes information both about and for the user.

1. Prepare your environment

The server-side application in this tutorial uses Node.js as its software platform. Check that you have Node.js installed and set up with default settings for your environment.

Note: To test that Node.js is installed and set up correctly, you can create a server.js file in a text editor of your choice.

2. Set up your PeaKass account

To use the Web API, start by creating a PeaKass user account on the PeaKass developer website. With your account set up, go to the Dashboard page and log in. Accept the latest Developer Terms of Service to complete your account set up.

3. Register an application with PeaKass

Any application can request data from PeaKass API endpoints without requiring registration. However, if your application seeks access to user’s personal data (profile, etc.), it **must** be registered. Registering brings additional perks: registered applications get benefits such as higher rate limits at some endpoints.

Note: You can register an application even before it's launched.

4. Authorize users

The authorization code flow gets a code from the PeaKass account service and exchanges that code for an access token. Afterwards, the flow uses code-to-token exchange requires a secret key. For security purposes, this is done through direct server-to-server communication.

Note: Authorization code flow used in this tutorial is available on PeaKass Github page.