Auth0 setup

Tartalom

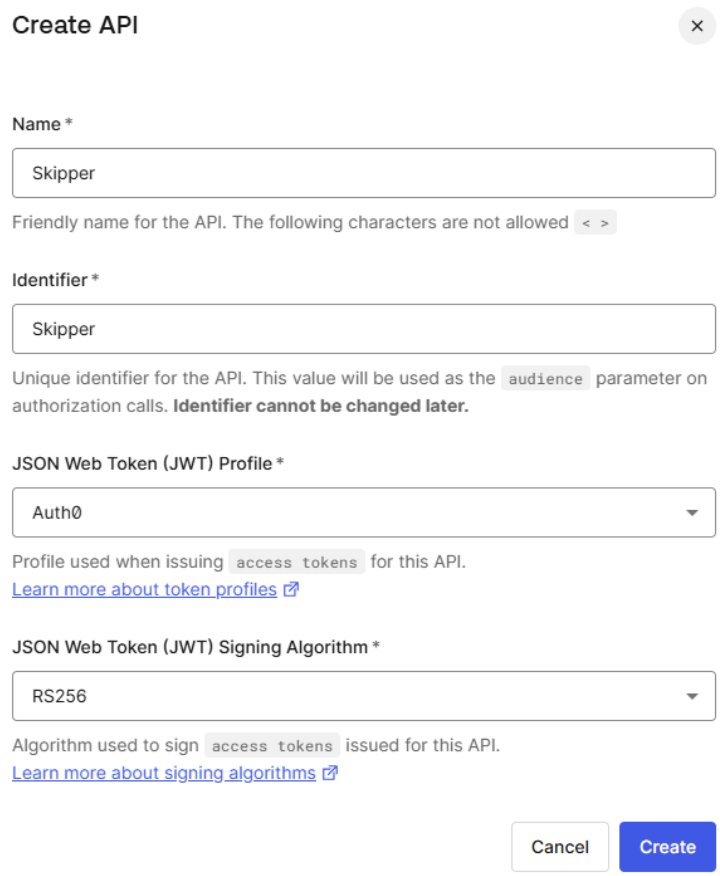
SKIPPER - Skills Portfolio of Personal Development (n° 2022-1-HU01-KA220-HED-000086240) project has been funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Registration

Auth0 registration can be done via various sign-up method. It does not matter what you choose.

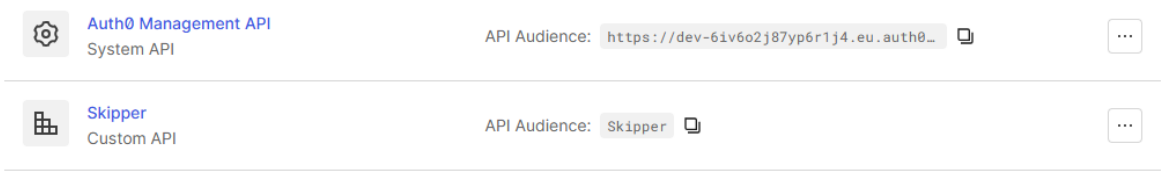
Create a custom API

In the left menu click on **Applications --> APIs** and click on **Create API** button. Then fill the inputs as follows:



1. Figure Create a new API

Let the **Name** and **Identifier** identical. After you created your custom API you should see two APIs:



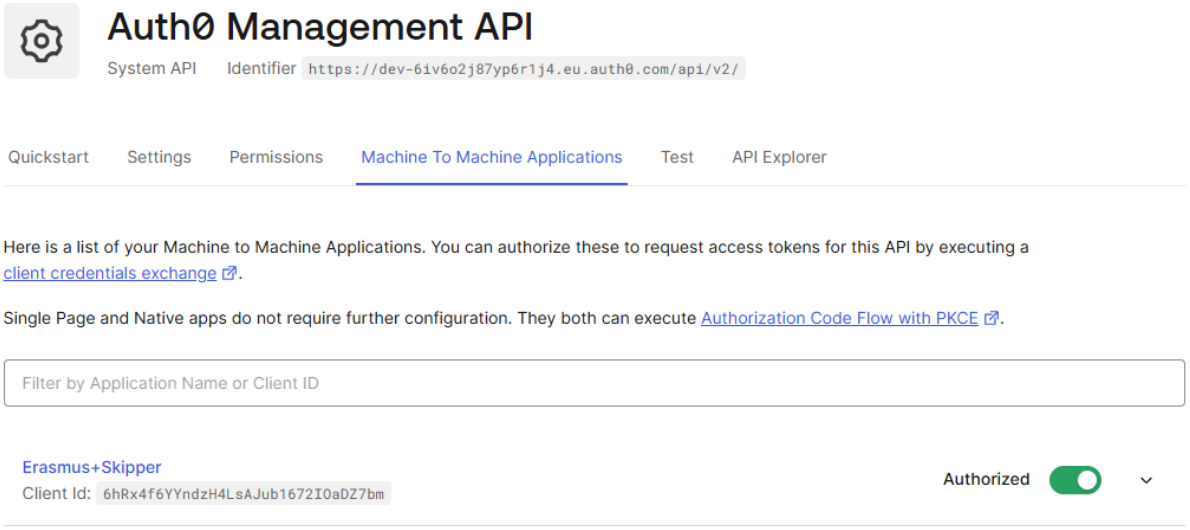
2. Figure Two needed APIs

Skipper API is your own custom API. Auth0 Management API a system generated API which was generated automatically.

Auth0 Management API

Machine To Machine Tab:

Enable the Machine to machine application as authorized.

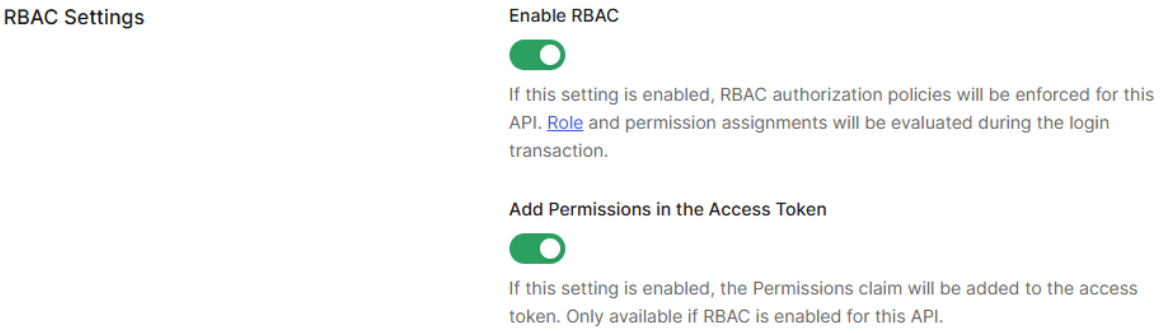


3. Figure Auth0 Management API

Your custom API (Skipper)

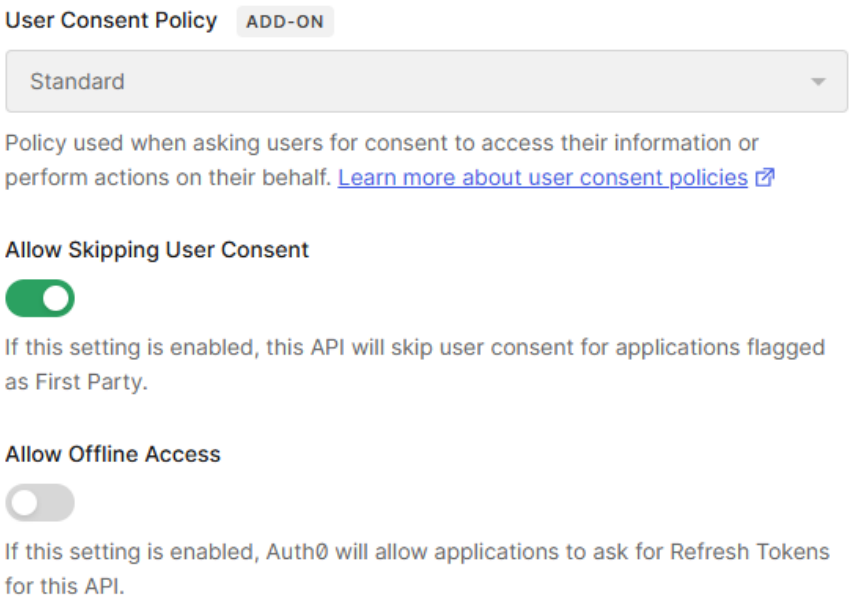
Settings tab:

Enable RBAC settings



4. Figure RBAC settings

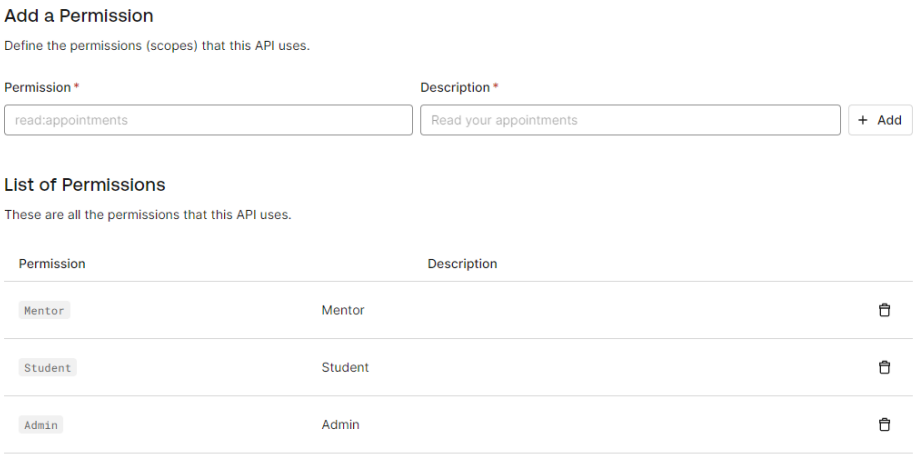
Access settings:



5. Figure Access settings

Permissions tab

In **Permissions** tab create Student, Mentor and Admin permissions as follows:



1. ábra Add permissions

Machine to Machine Applications tab

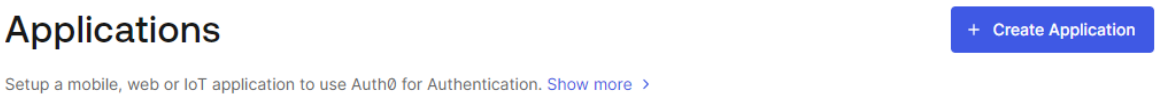
Enable the newly created application (switch to Authorized). If not yet have then after application is created return to this page to enable.

2. ábra Enable new API

Create Application on Applications page Backend

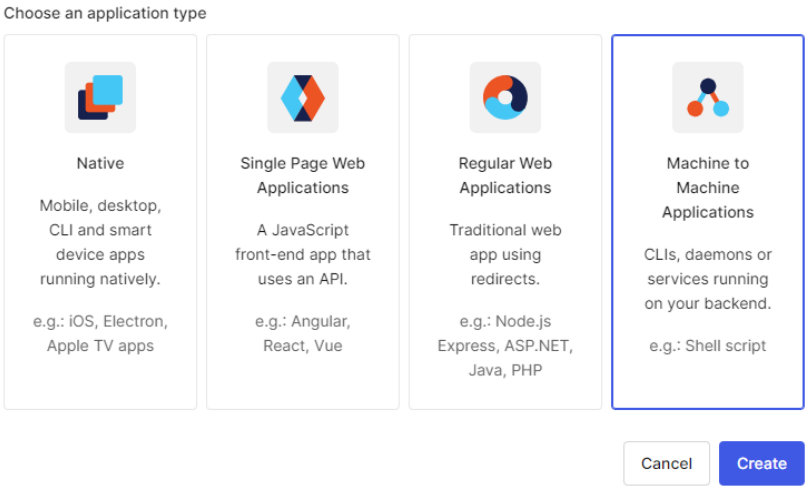
To use back-end server with AUTH0 a new application must be created. After you log-in into Auth0 dashboard the following steps are required to make:

1. In the left menu click on **Applications --> Applications** and click on **Create Application** button.



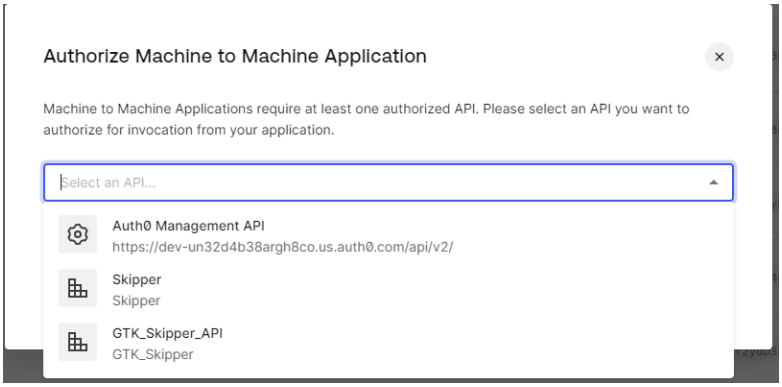
3. ábra Create new application

Add a name, select **Machinet to Machine Applications** and click on **Create** button.



4. ábra Machine to Machine Applications

Your custom API must be selected (in example: Skipper) for the new M2M Application. If does not have yet you have to create one before create applications.



5. ábra Select an API for Application

2. Select the newly created API and make the following settings:
    a. **Settings** tab:

Application URIs section:

- **Allowed Callback URLs**: http://localhost:3000, http://localhost:3000/dashboard, https://**aut0-skipper-stage.eu**.webtask.run/auth0-authentication-api-debugger, http://localhost:3000/events

Highlighted part must be your unique tenant domain.

- Allowed Logout URLs: http://localhost:3000
- Allowed Web Origins: http://localhost:3000

Application URIs

Application Login URI

https://myapp.org/login

In some scenarios, Auth0 will need to redirect to your application's login page. This URI needs to point to a route in your application that should redirect to your tenant's `/authorize` endpoint. Learn more

Allowed Callback URLs

http://localhost:3000, http://localhost:3000/dashboard, https://aut0-skipper-stage.eu.webtask.run/auth0-authentication-api-debugger, http://localhost:3000/events

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (`https://`) otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://` . You can use Organization URL parameters in these URLs.

Allowed Logout URLs

http://localhost:3000

Comma-separated list of allowed logout URLs for redirecting users post-logout. You can use wildcards at the subdomain level (`*.google.com`). Query strings and hash information are not taken into account when validating these URLs. Learn more about logout

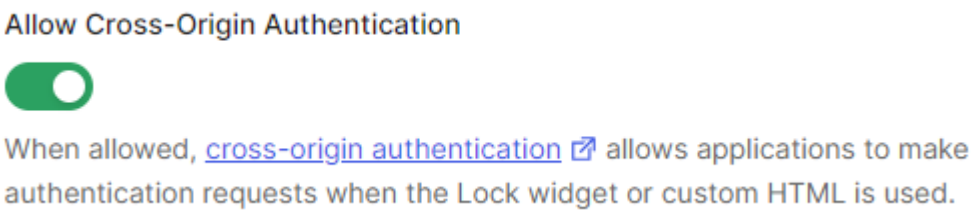
Allowed Web Origins

http://localhost:3000

Comma-separated list of allowed origins for use with Cross-Origin Authentication, Device Flow, and web message response mode, in the form of `<scheme> "://" <host> [ ":" <port> ]` , such as `https://login.mydomain.com` or `http://localhost:3000` . You can use wildcards at the subdomain level (e.g.: `https://*.contoso.com`). Query strings and hash information are not taken into account when validating these URLs.

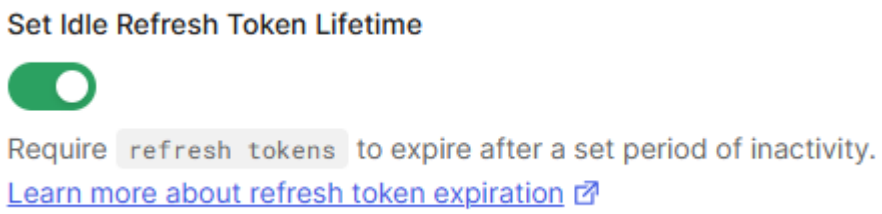
Cross-Origin Authentication section:

Enable Cross-Origin Authentication.



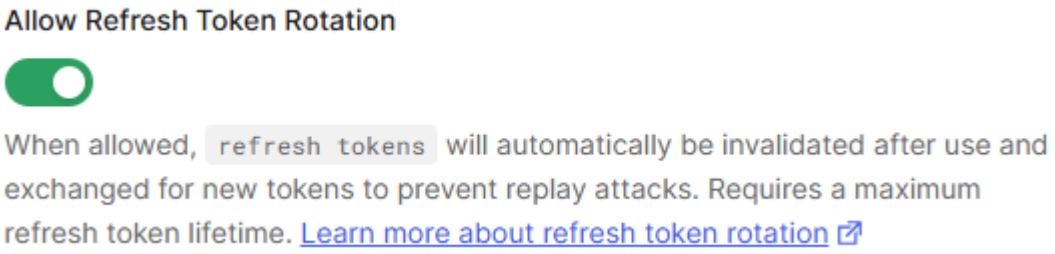
Refresh Token Expiration section:

Enable Inactivity Expiration



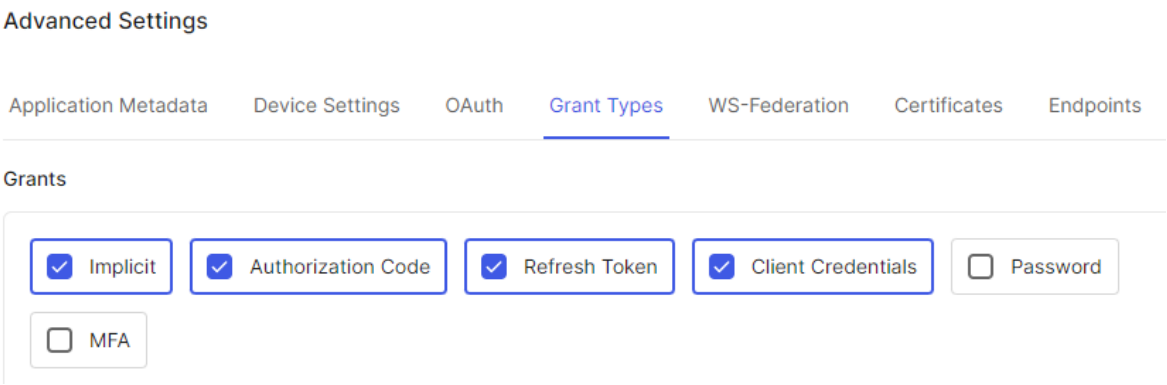
Refresh Token Rotation section:

Enable Rotation.



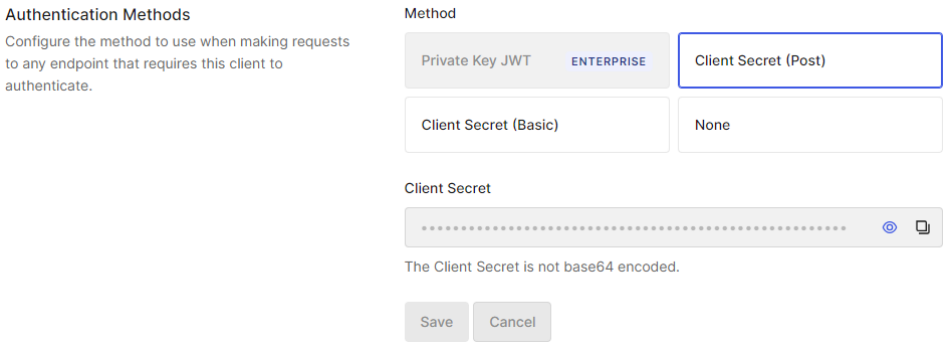
Advanced Settings section:

On Grant Types tab enable: Implicit, Authorization Code, Refresh Token and Client Credentials



Credentials tab

Select **Client Secret (Post)** option and save it.



6. ábra Set authentication method

APIs tab

Enable **Auth0 Management API** and the **created API** (in example: Skipper)**.**

7. ábra Enable APIs

- Backend env configs

  HTTP_PORT=3000
  CLIENT_ORIGIN_URL=FRONT_END_ORIGIN_URL
  AUTH0_AUDIENCE=Created API name (in example: Skipper)
  AUTH0_DOMAIN= Created Application (tenant) Domain from setting tab
  MANAGEMENT_CLIENT_DOMAIN= Created Application (tenant) Domain from setting tab
  CLIENT_SECRET= Created Application Client Secret from setting tab
  CLIENT_ID= Created Application Client ID from setting tab
  PDF_WORKER=../pdf/pdfWorker.ts
  QUESTIONNARIE_FILE=../../../resources/data-service/csv/questionnaries-proofread.csv

  MINIO_ENDPOINT=MINIO endpoint
  MINIO_PORT=MINIO port
  MINIO_USER=MINIO username
  MINIO_PASSWORD=MINIO password
  MINIO_BUCKET=BUCKET name
  MINIO_SSL=true or false

NODEMAILER_HOST=HOST
NODEMAILER_PORT=POST
NODEMAILER_USER=Username
NODEMAILER_PASSWORD=password
NODEMAILER_SSL=true or false

Minio

To store passports, university logo and profile pictures you need to configure a Minio server. If the minio server is configured appropriately you only need to specify the minio env variables and it should work.

Email

To make system able to send e-mail notifications you need to configure the Nodemailer env variables and set up an e-mail service.

The e-mail template can be found under **\src\services\email\templates** folder. You are free to modify as you wish.

Passport translations

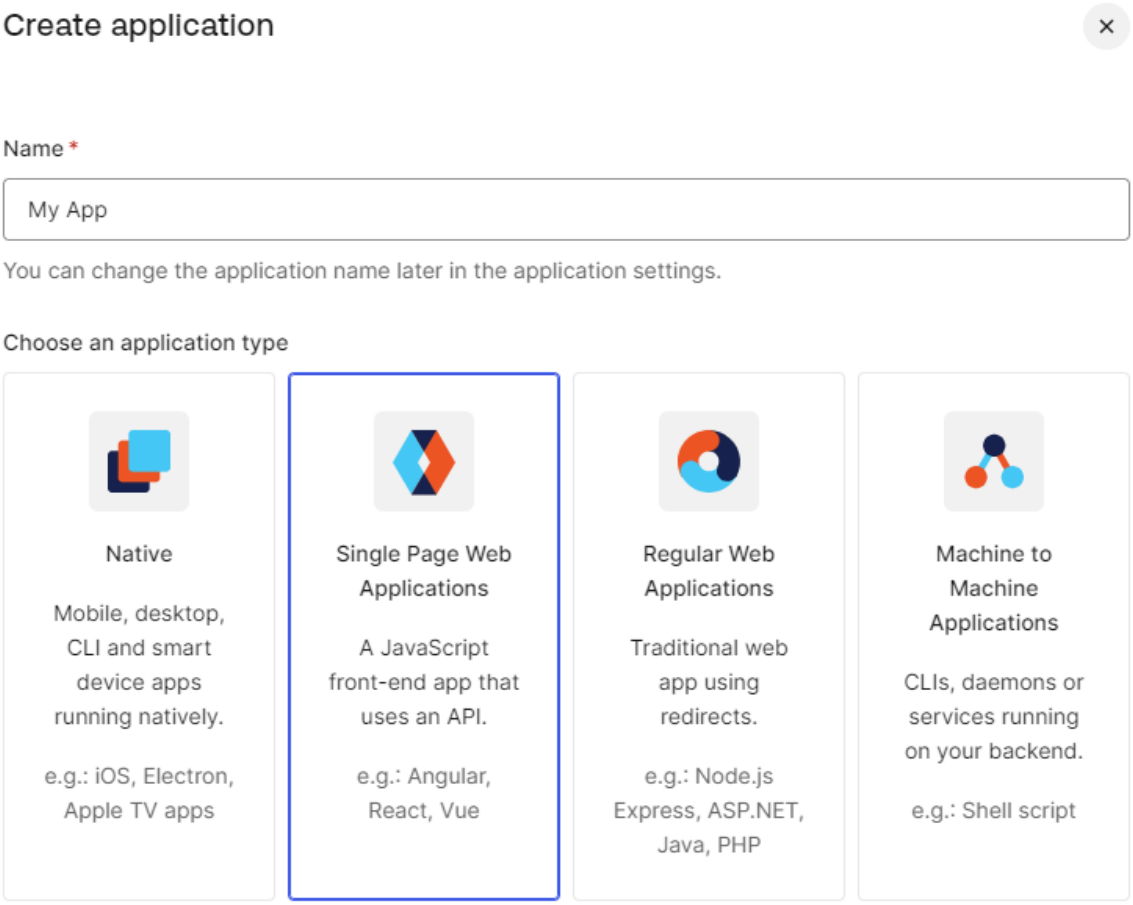
Passports are always translated into application language. The translation files can be found under **resources\locales** folder. The translation files should place in subfolder named with language code (e.g.: en) and must be named as **translations.json**.

Badges, logos

The badges nad logos are placed under **assets** folder. These images are used during passport generation.

Create Application on Applications page Frontend

- Applications -> Applications menu
- Click on **Create Application** button
- Create new **Single Page Web Application** with react technology



8. ábra Frontend Application létrehozása

Settings tab:

- **Allowed URLs**: http://localhost:3006, http://localhost:3006/dashboard, **https://dev-un32d4b38argh8co.us**.webtask.run/auth0-authentication-api-debugger, http://localhost:3006/events, http://skipper.example:3006/institution, http://localhost:3006/profile-setup, **your-skipper-domain**/dashboard
- **Allowed Logout URLs**: **<your skipper domain>,** http://localhost:3006
- **Allowed Web Origins**: http://localhost:3006
- Refresh Token Rotation: On

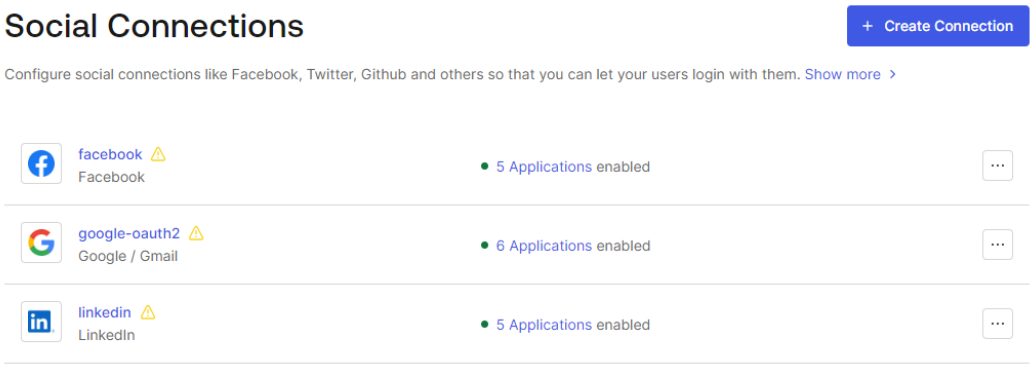
Frontend .env confings

Create .env file in the frontend application root folder with the following content:

- VITE_AUTH0_DOMAIN: Created Application Domain from setting tab
- VITE_AUTH0_CLIENT_ID: Created Application Client ID from setting tab
- VITE_AUDIANCE: Created API name
- VITE_AUTH0_CALLBACK_URL: <url>/dashboard
- VITE_API_SERVER_URL: <url>/api
- VITE_LOGOUT_URL: <url>
- PORT: 3006 or any other port except the port that the backend uses

Authentication Settings

- **Email-Password** authentication is on by default
- Authentication -> Social
- Click on **Create connection** button
- Add the following options:
    - Facebook
    - Goggle oauth2
    - Linkedin

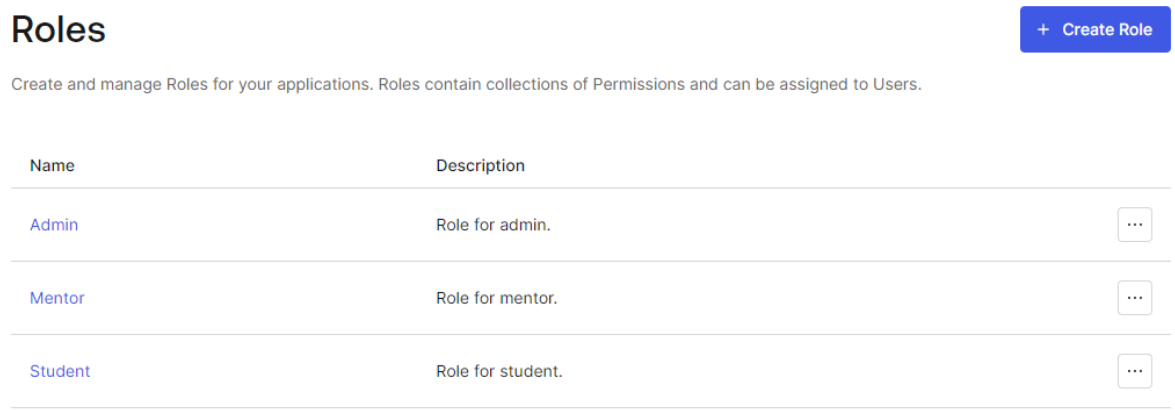


9. ábra Social login options

In every social login disable the possibility of Sync user profile:

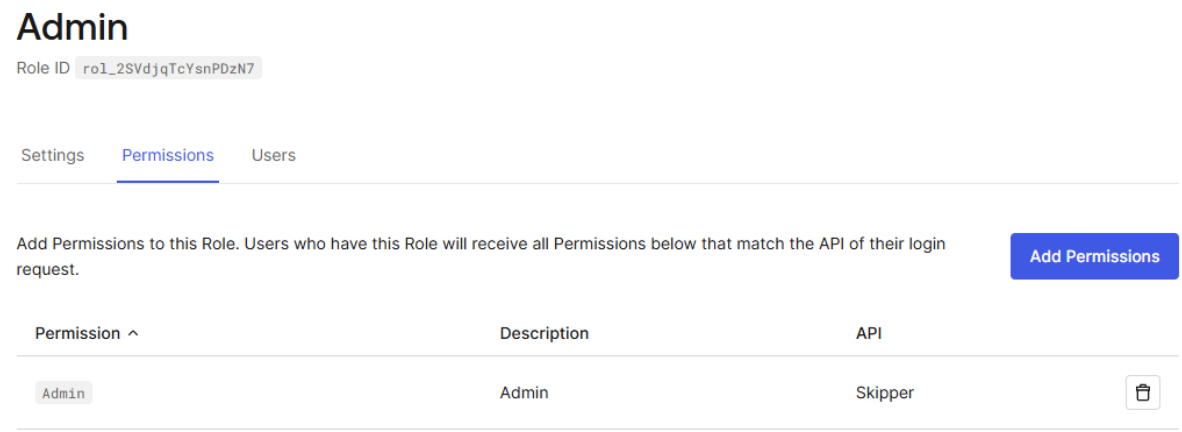
Role handling

- User Management -> Roles
- Create the following roles by clicking on **Create role** button:
    - Name: Admin, Description: <Some role description>
    - Name: Mentor, Description: <Some role description>
    - Name: Student, Description: <Some role description>



10. ábra Required roles

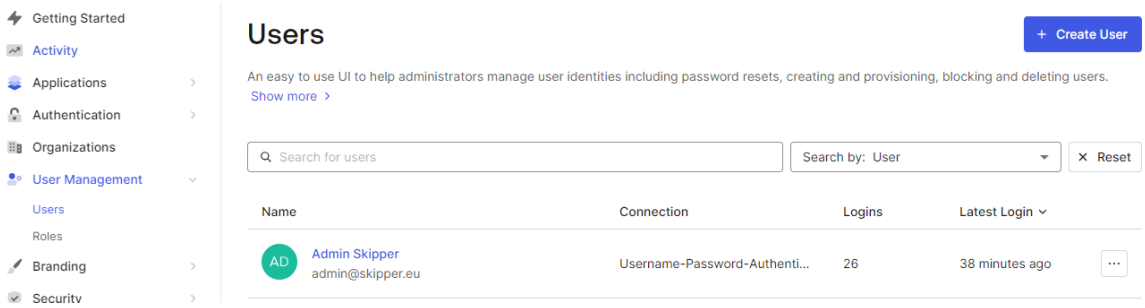
For each role please set the appropriate permission on Permission tab of selected role with Add Permission button. The example show this for Admin role:



6. Figure For example: Admin role

Add administrator user

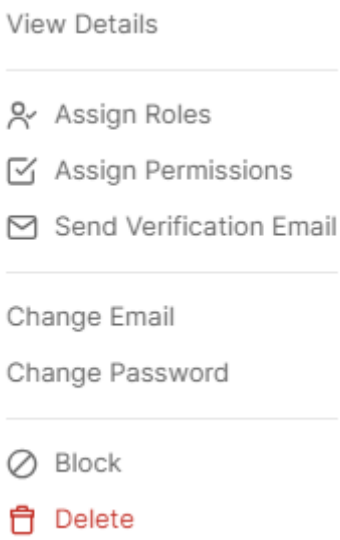
- User Management -> Users
- Click on **Create User** button and add the user information
- After the user creation the new user appear on Users list like on the following example



11. ábra Example user

- Click on **...** button in the users table and select **Assign Roles** option
- For admin on **Details** tab and in **User Metadata** section the registered university id (edu_id) must be configured as follows:

12. ábra Link admin to university



13. ábra ... menu on user row

- In a modal you can add Admin role to the user by selecting the created Admin role

Create Actions

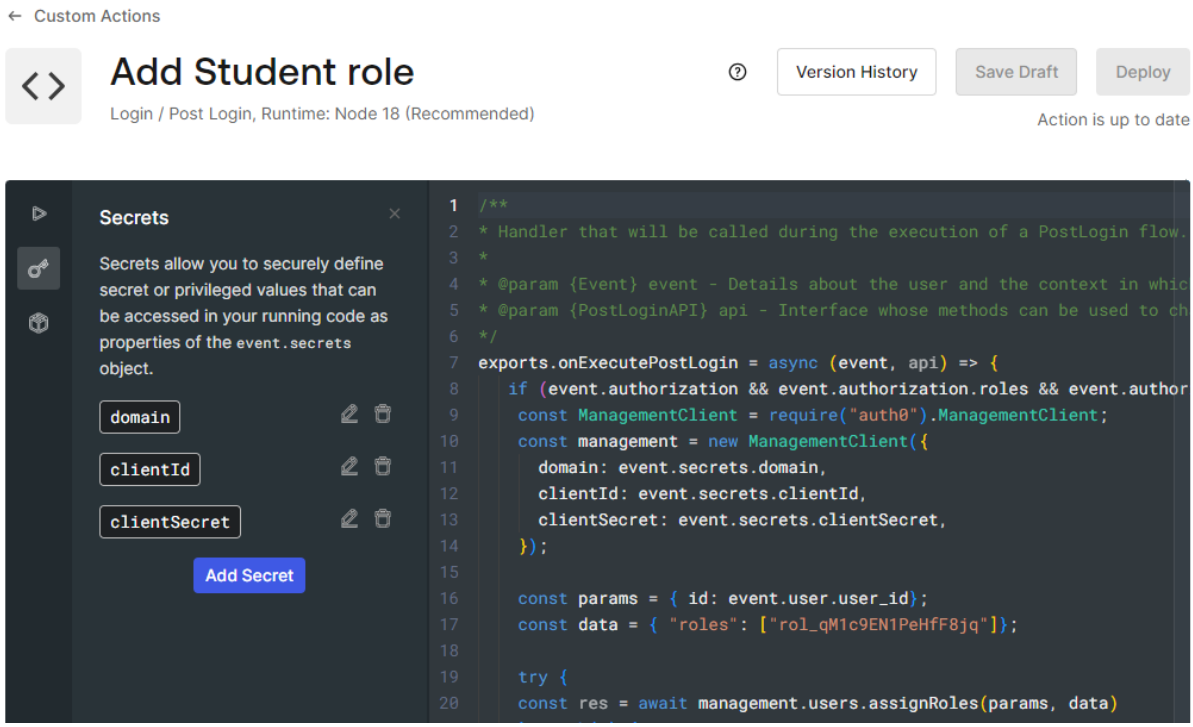
You need to create two custom actions to add default roles to the students automatically and to return user roles in the token on login action.

- Actions -> Library menu
- Click on **Create Action** button and select **Build from scratch** option:

First Action

- Name: Name of your action like: “Add Student role”
- Trigger: Login / Post Login
- Runtime: Node 18

When you have an action, you can define what it should do in a code editor like in the following picture.



14. ábra Add Student role action

Add Secrets

In the left hand side you can add secrets by clicking on the second button with a key in it called “Secrets”.

Add the following keys by clicking **Add Secret** button:

- Key: domain, Value: <your Application domain from setting tab>
- Key: clientId, Value: <your Application Client ID from setting tab>
- Key: clientSecret, Value: <your Application Client Secret from setting tab>

Add dependencies by clicking **Dependencies** button under the Secrets button:

- **Click on Add Dependency**
    a. Name: auth0
    b. Version: 4.2.0

Now you can add the following code:

```
exports.onExecutePostLogin = async (event, api) => {

   if (event.authorization && event.authorization.roles &&
event.authorization.roles.length === 0) {

    const ManagementClient = require("auth0").ManagementClient;

    const management = new ManagementClient({

      domain: event.secrets.domain,

      clientId: event.secrets.clientId,

      clientSecret: event.secrets.clientSecret,

    });


    const params = { id: event.user.user_id};

    const data = { "roles": ["<ID of the created Student role>"]};


    try {

    const res = await management.users.assignRoles(params, data)

    } catch(e) {

      console.log(e)

    }

  }

};
```

Now you can click on **Deploy** button to deploy your action.

Second action

Create new action with the name of “Add roles to token” and add the following settings:

Secrets:

- domain

- clientId
- clientSecret

Dependencies:

- Auth0@4.2.0

Code:

```
exports.onExecutePostLogin = async (event, api) => {
  const ManagementClient = require('auth0').ManagementClient;
  const management = new ManagementClient({
    domain: event.secrets.domain,
    clientId: event.secrets.clientId,
    clientSecret: event.secrets.clientSecret,
    scope: 'read:users update:users read:roles'
  });

  const namespace = 'roles';
  if (event.authorization) {
    api.idToken.setCustomClaim(`${namespace}/roles`,
event.authorization.roles);
    api.accessToken.setCustomClaim(`${namespace}/roles`,
event.authorization.roles);
    api.idToken.setCustomClaim(`${namespace}/registry_code`,
event.user.user_metadata.registry_code);
    api.accessToken.setCustomClaim(`${namespace}/registry_code`,
event.user.user_metadata.registry_code);
  }
};
```

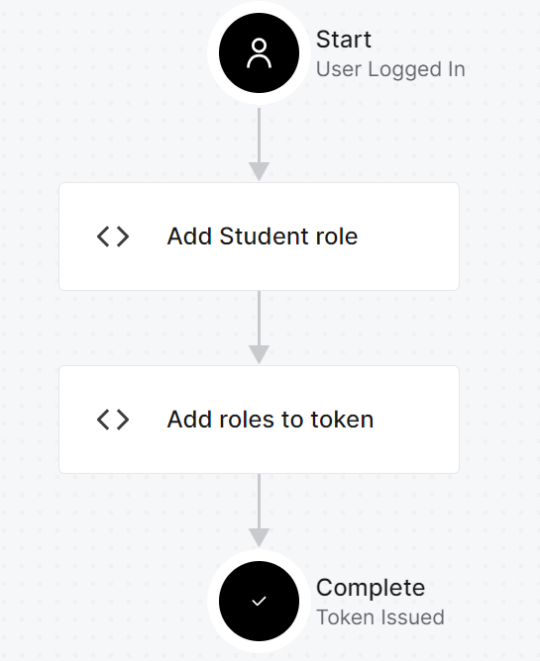
Deploy your action by clicking on **Deploy** button.

Use Actions

- Actions -> Flows menu
- Select **Login** option

- Now you can drag and drop your custom actions from the right hand side to the login flow in the following order:
    1. **Add student role** action
    2. **Add roles to token** action

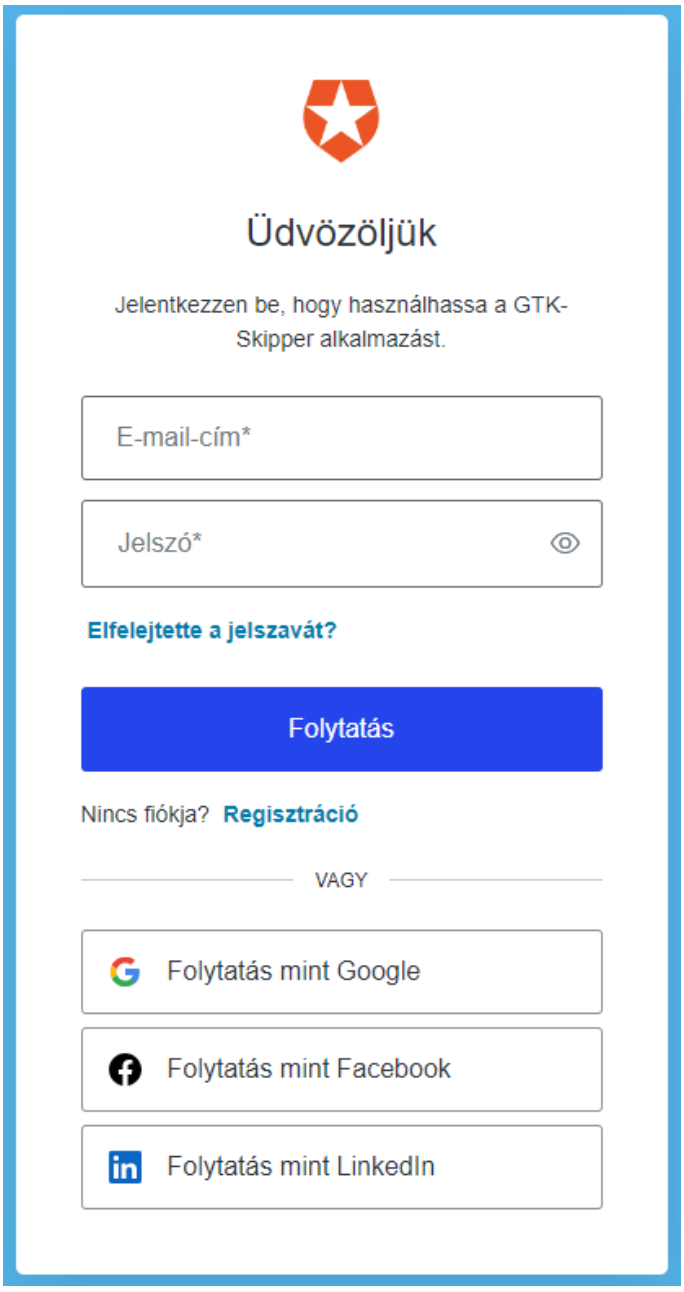
After the setting you need see the following flow in the editor:



15. ábra Login actions flow

Login page

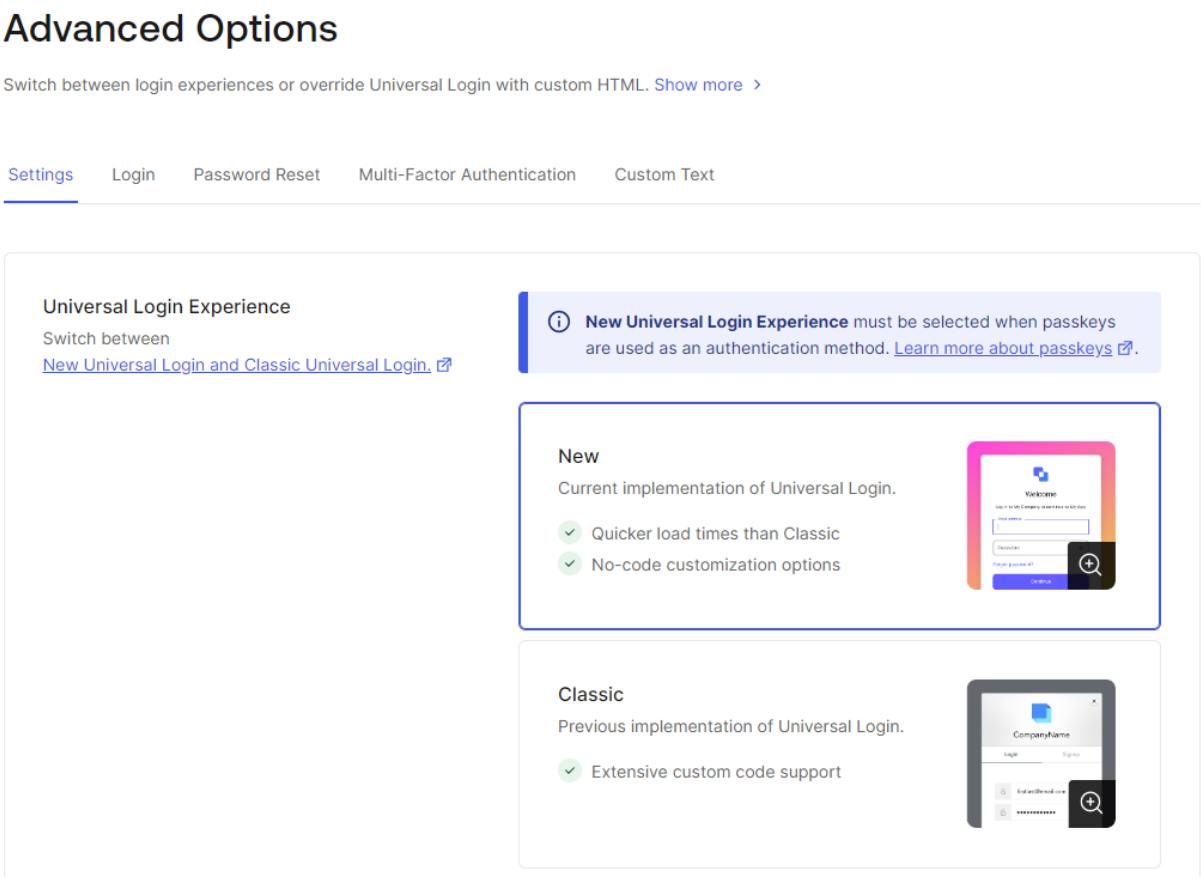
We have the **New Universal Login** page in the Skipper application, we can see that in the following picture.



16. ábra New Universal Login page

Select New Login Page

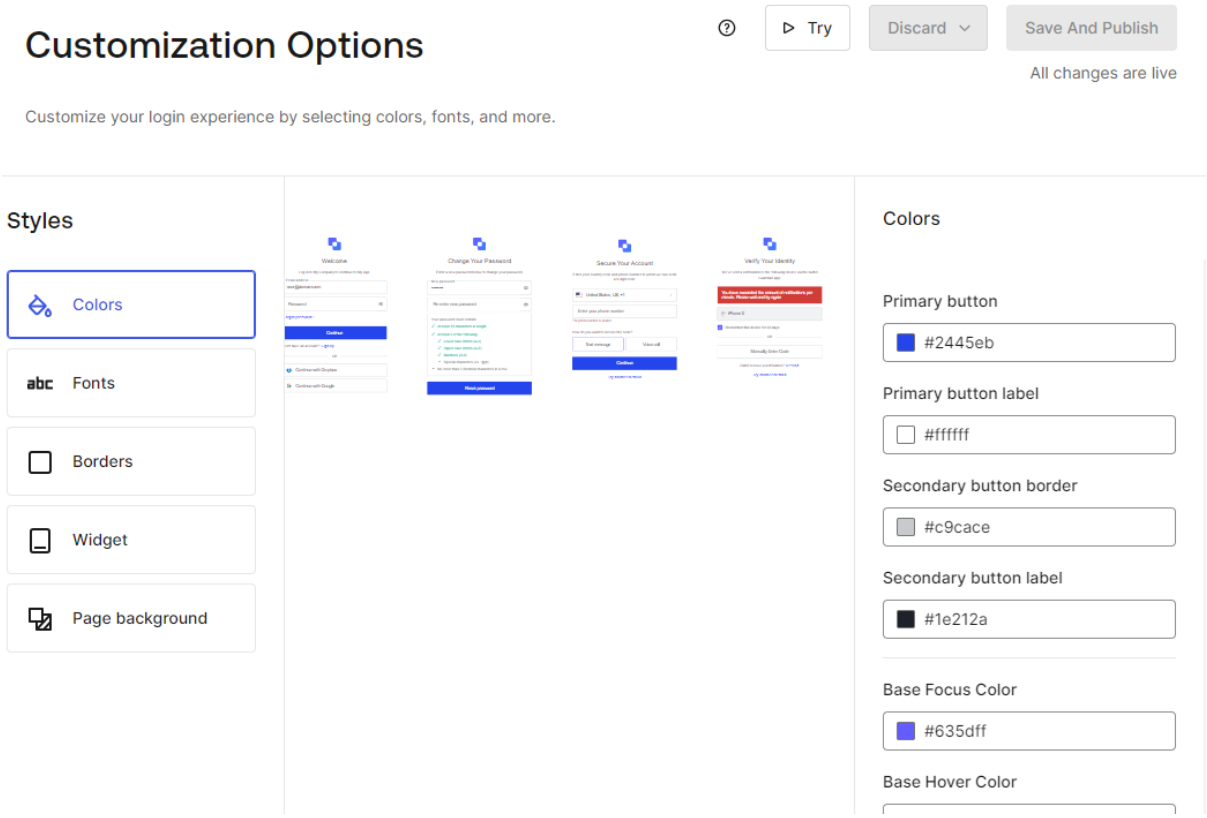
- Branding -> Universal Login menu
- Click on **Advanced Options** card in the bott of the page
- In the **Setting tab** you should select the New Universal Login experience



17. ábra How to select Login Page

Create Custom Login Page

- Branding -> Universal Login menu
- In this page you can customize the following properties as you wish:
    - Company Logo
    - Primary Color
    - Page Background Color
- In the **Customization Options** you can customize almost every element of the login page, like labels, buttons, fonts, etc..

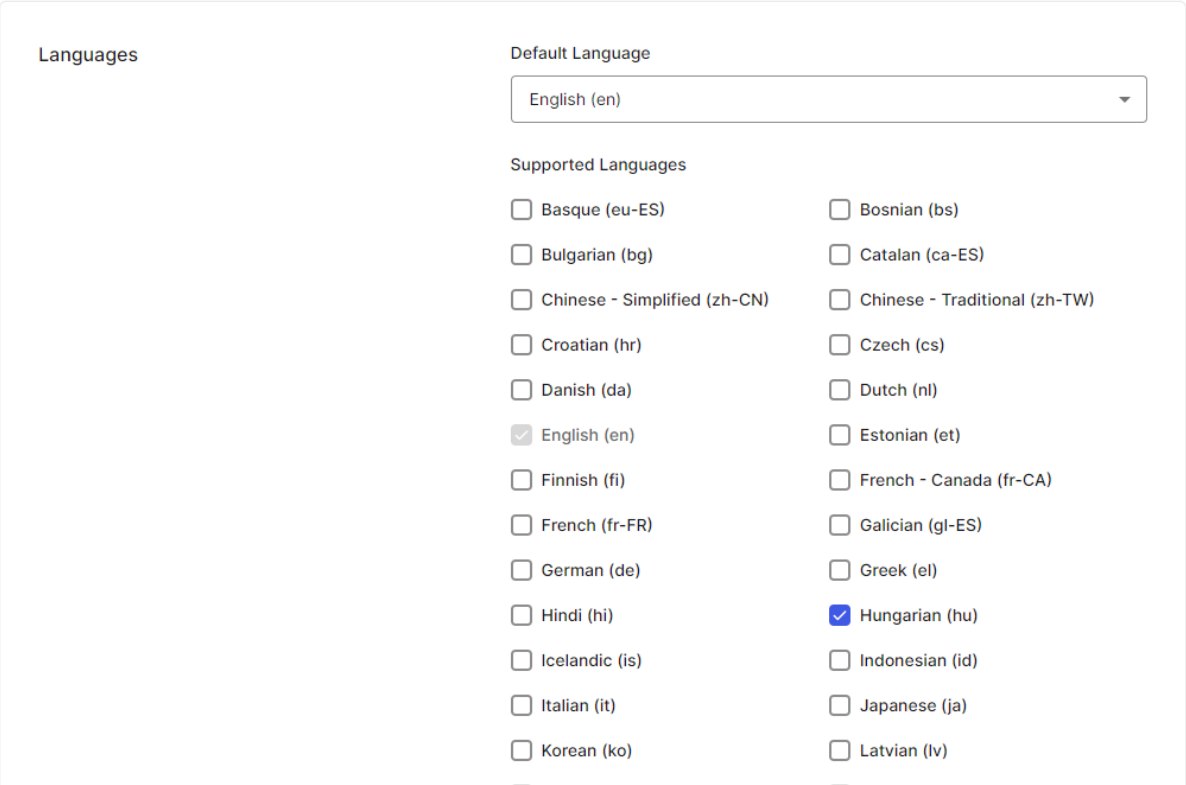


18. ábra Login Page Customization

Translations

The Universal Login page will be translated to the selected languages automatically, but you have to select the allowed languages in the Auth0 dashboard before that.

In the **Settings** menu there is a card with the title of **Languages**, this is where you can select the languages.

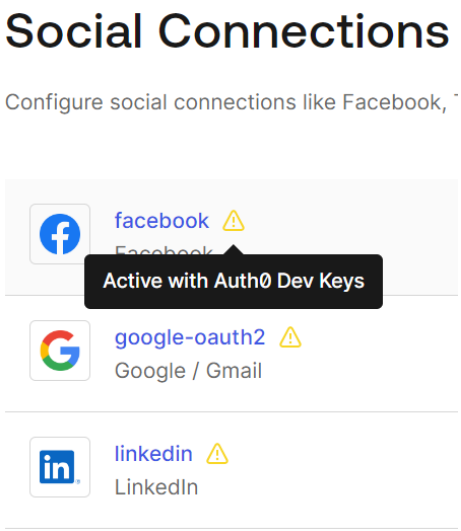


19. ábra Select allowed languages

Social login settings

- Google
- Facebook
- Linkedin

Social logins now works with developer keys in development mode. We need to set this to production with the following settings. If every setting is done, the warning icons will disappear from Auth0 "Authentication" > "Social" menu.



Google

Create a Google Cloud Project in the Google Cloud Console.

Create a new project if you don't have one already:

- Click on the project dropdown at the top, then "New Project."
- Name your project and choose a billing account if necessary.

Enable Google People API

- In your project, go to the "APIs & Services" > "Library."
- Search for "Google People API"
- Enable the "Google People API" API.

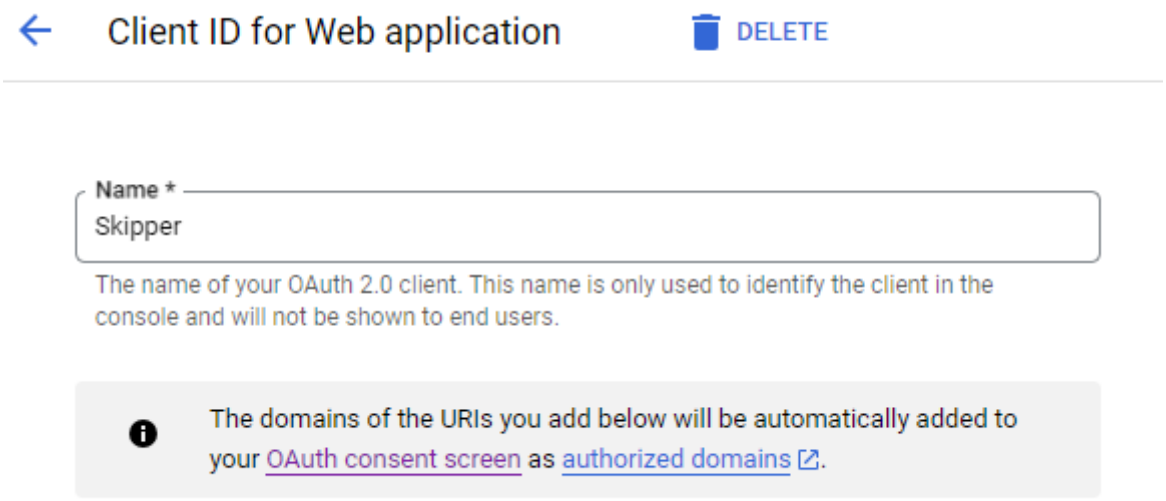
Create OAuth Credentials

- Go to "APIs & Services" > "Credentials."
- Click on "Create Credentials" and select "OAuth Client ID"
- Choose "Web application" as the application type.

Set up the consent screen if not already done:

- You'll need to fill in basic information, such as the application name, support email, and developer contact information.

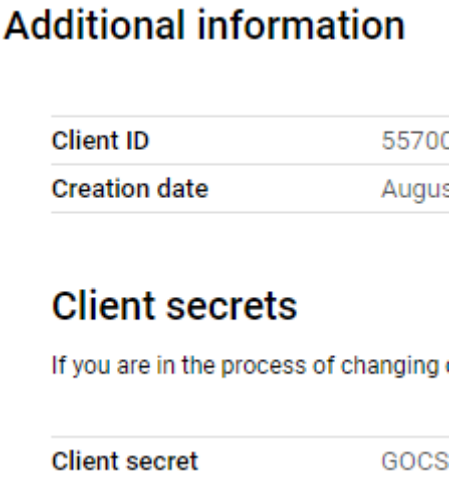
- You might need to verify the app depending on its scope and usage.



Configure the OAuth client:

- Authorized redirect URIs: Add the callback URLs that Auth0 will use, which you can find in the Auth0 dashboard on the top left corner.
    - https:// YOUR_DOMAIN /login/callback
    - DOMAIN: <whole domain from your auth0 account (like in the picture)>.<Region>.auth0.com

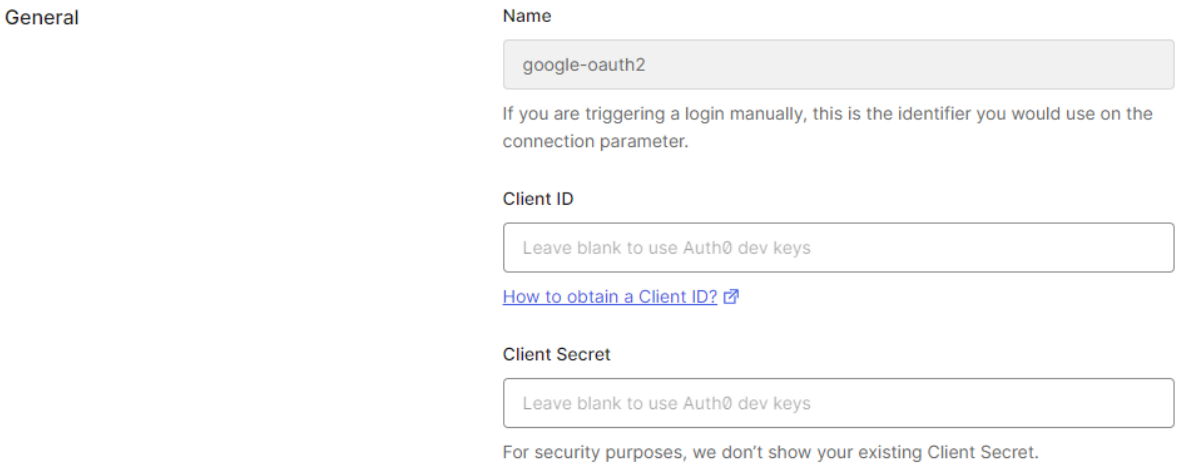
Save and get the Client ID and Client Secret.



Set up Auth0 with Google OAuth Credentials

- Go to the Auth0 Dashboard and navigate to "Authentication" > "Social."
- Select Google from the available connections.

In the General section enter your Google OAuth Client ID and Client Secret that you obtained from the Google Cloud Console.



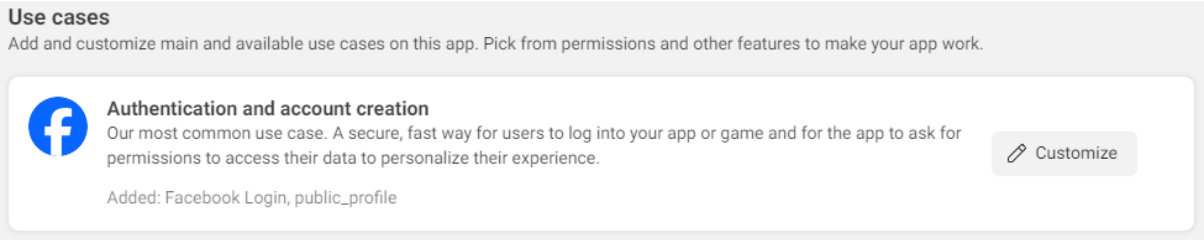
Save your settings.

Facebook

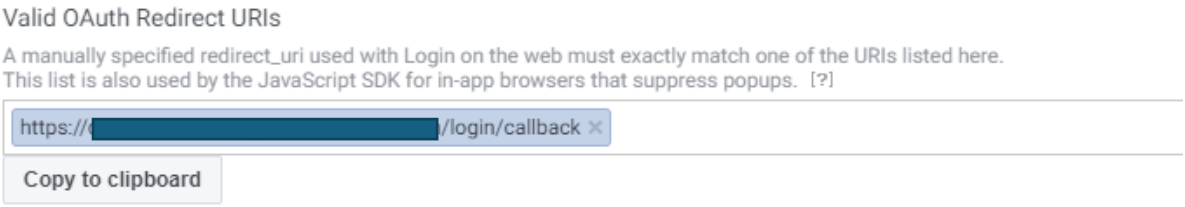
Create a Facebook App in the Facebook for Developers website.

- Log in with your Facebook account.
- Go to "My Apps" in the top right and click "Create App."
- In the " Use cases " tab you need to select " Authenticate and request data from users with Facebook Login "
- Fill in the basic details (e.g., App Name, Contact Email) and click "Create App ID."

After the app is created, you need to set redirect url for login. This can be done in the "Use cases " menupoint with clicking on "Customize" in the " Authentication and account creation " card.



In the left menu bar select "Settings" and add the " Valid OAuth Redirect URIs ". This needs to be the same as that you set in the GCP console.



Save your settings with clicking on "Save changes" button in the end of the page.

In the "App settings" > "Basic" menu, you can find the App ID and the App secret, that you need to set up in the Auth0 dasboard.

Set Up Auth0 with Facebook OAuth Credentials as you did for Google.

Linkedin

Create a LinkedIn Application via the LinkedIn Developer Portal.

- Log in with your LinkedIn account.
- Click on "My Apps" and then "Create App."
- Fill in the required details (e.g., App Name, LinkedIn Page, Company, Email).
- Agree to the LinkedIn API Terms of Use and click "Create app."

Set Up LinkedIn OAuth

- After creating the app, you'll be directed to the application dashboard.
- In the "Auth" tab of your app dashboard, you'll find your Client ID and Client Secret.
- You'll need these for Auth0 configuration.

Configure OAuth Redirect URLs

- In the LinkedIn application dashboard, under "Auth", go to "Redirect URLs".
- Add the Authorized Redirect URLs provided by Auth0.
    - You can find these URLs in the Auth0 dashboard under the LinkedIn connection settings.
    - https:// YOUR_DOMAIN /login/callback

Set Up Auth0 with LinkedIn OAuth Credentials