Supabase authentication | WeWeb Documentation

|  |  |
| --- | --- |
| URL source | <https://docs.weweb.io/plugins/auth-systems/supabase-auth.html> |
| Date de scraping | 2025-04-08T15:55:17.823453+00:00 |

## Supabase authentication ​

## Plugin configuration ​

Supabase comes with an in-built authentication system which you can use in WeWeb.  
To add the Supabase Auth Plugin in WeWeb, go to Plugins > Authentication :  
  
Assuming you have already provided your Supabase project URL and public API key when setting up the Supabase Data source , the only thing left to do will be to add your private API key:  
  
In Supabase, you can find your private API key in Settings > API :  
  
You will then be invited to choose a page to redirect unauthenticated users, i.e. users who are NOT signed-in:

### Images

<https://docs.weweb.io/assets/supabase-auth1.B4K2ndsc.png>

<https://docs.weweb.io/assets/supabase-auth2.DZL0-7dJ.png>

<https://docs.weweb.io/assets/supabase-auth3.ZTXCZbXJ.png>

<https://docs.weweb.io/assets/supabase-auth4.-VgpaYpI.png>

## Sign up and log in flows ​

Once you have added the Supabase Auth plugin to your WeWeb project, you will get access to:  
dedicated Supabase Auth actions in workflows, and dedicated UI elements in the Add menu.  
In the example below, you can see we added the Login UI element related to the Supabase Auth plugin:  
  
That UI element comes with a predefined workflow using the Login with email action but you have access to many more:

### Images

<https://docs.weweb.io/assets/smtp.DjWJ48gh.png>

<https://docs.weweb.io/assets/supabase-auth5.BAMtf9IR.png>

<https://docs.weweb.io/assets/supabase-auth6.DBLzveC3.png>

## Sign up workflow ​

In the example below, we added a workflow on a sign up form, chose the On submit trigger, and added the Supabase Sign up action:  
  
Then, we mapped the email, password, and metadata information in the form to the email, password, and metadata in Supabase before choosing what page the new user should be redirected to:  
  
In the example above, we made sure to add the user's name as an item in that user's metadata.  
In Supabase, you can find the user's metadata in JSON format in a dedicated field of the users table, named raw\_user\_meta\_data :  
  
The same logic will apply to any Supabase action you want to trigger.

### Images

<https://docs.weweb.io/assets/supabase-auth7.CChN863q.gif>

<https://docs.weweb.io/assets/supabase-auth8.DhZlzdub.gif>

<https://docs.weweb.io/assets/supabase-auth9.C59sVhLI.png>

## Reset password flow ​

You'll need three things to allow users to reset their passwords:  
a WeWeb page where users type in their email address and submit a password reset request the URL of your website and an email template properly configured in Supabase a WeWeb page where users are redirected when they click on the link in the email sent by Supabase

## 1. Forgot password page ​

In WeWeb, you'll need to create a page where users can make a request to reset their password.  
That page needs to include:  
an email input field, and a workflow with the Supabase Auth Forgot password action.  
You can design your own custom page or simply use the Reset password UI element that comes with the Supabase Auth plugin:  
  
In the example below, we trigger a Forgot password action on form submit, tell Supabase where it should send the reset password email, and say we want the user to be redirected to the Choose new password page when they click on the link in that email:

### Images

<https://docs.weweb.io/assets/supabase-reset-password1.DleUJJV1.png>

<https://docs.weweb.io/assets/supabase-reset-password2.BuyVG8e_.png>

## 2. Supabase configuration to send password reset email ​

In Supabase, you'll need to add the URL of your live app by going to Authentication > URL Configuration :  
  
In the example above, we added the URL of our project in the WeWeb Editor because we are still building the app but we would change it to the URL of our staging website or live website as we progressed through the building and publishing process.  
You can personalize the email Supabase sends by going to Authentication > Email Templates > Reset password :

### Images

<https://docs.weweb.io/assets/supabase-reset-password3.tjIo3cKg.png>

<https://docs.weweb.io/assets/supabase-reset-password4.B1LX2ROe.png>

## 3. Confirm password page ​

Back in WeWeb, you will need to create a page where users can type in and submit their new password.  
You can design your own custom page or use the Change password UI element that comes with the Supabase Auth plugin.  
In the example below, we triggered a workflow on a form submission and:  
check if the two password input values are identical if they're not, we display an error message if they are, we use the Confirm password action to send the new password to Supabase

### Images

<https://docs.weweb.io/assets/supabase-reset-password5.Bp54Jt1F.png>

## OAuth with providers ​

When using Supabase for authentication, sign up and login with an email address is enabled by default but you can also enable other methods of authentication, including but not limited to Google OAuth, Azure, GitHub, LinkedIn, and Slack:  
  
Each provider will have their own configuration steps which you'll need to work out using their documentation.  
Broadly speaking, it will look something like this:  
In the provider's interface  
create the project add authorized domains create OAuth credentials define redirect URLs  
In Supabase  
save the OAuth credentials of the provider add the base URL of your website define one or more URLs where users can be redirected after logging in  
In WeWeb  
trigger a workflow with the Login with provider action select what provider's consent screen you want to display select which page users should be redirected to after they successfully log in

### Images

<https://docs.weweb.io/assets/supabase-auth17.D7c3pVvo.png>

<https://docs.weweb.io/assets/oauth-flow.Bar2D9pL.png>

## Update user metadata ​

In the default users table of a Supabase project, you can add user metadata in the raw\_user\_meta\_data column which expects a JSON.  
To pass user metadata from WeWeb to Supabase, you have two options.  
1- select a key in the Metadata dropdown when using the Sign Up or Update User Metadata action in a workflow  
  
2- bind the Metadata field to update Supabase with custom metadata.  
This fields expects an array of objects where, for each object, you explicitly say this is my key and this is the corresponding value for this item. For example:  
  
In the example above, we used a no-code formula to create the array of objects but, if you're used to the syntax, it can be easier to write in JavaScript:  
  
Here's a code snippet you might find helpful to avoid typos:

### Code

Langage: unknown

return [  
 {"key":"favorite\_joke", "value":"something about an elephant"},  
 {"key":"favorite\_chocolate", "value":"dark"}  
]

### Images

<https://docs.weweb.io/assets/supabase-auth18.BwxAAGij.png>

<https://docs.weweb.io/assets/supabase-auth20.cUe8_vry.png>

<https://docs.weweb.io/assets/supabase-auth19.CDT3yUkZ.png>

## User roles & permissions ​

Now let's say we want to gate content and set different permissions based on a user's role. We will need to:  
create a roles table in Supabase with the list of available roles create a join table in Supabase that maps the users with a role tell the Supabase Auth plugin in WeWeb where it can find the information

## 1. Add roles in Supabase ​

In Supabase, we'll need to create a roles table with a list of roles and a join table that links the roles table with our users table.  
First, let's create a roles table with three roles and make sure that each role had a UUID and a name :

### Images

<https://docs.weweb.io/assets/supabase-auth10.BR74bRK1.png>

## 2. Join roles and users in Supabase ​

Second, let's create a userRoles join table:  
  
In the join table above, you can see we have an id field that is uniquely identifiable thanks to a UUID.  
This unique id is linked to a userId , which is also a UUID. More specifically, it is the UUID we find in the id field of the users table in the auth schema:  
  
Each row in our userRoles table is also linked to a roleId which is the UUID we find in the id field of the roles table in the public schema:

### Images

<https://docs.weweb.io/assets/supabase-auth11.Vcm0Dn9P.png>

<https://docs.weweb.io/assets/supabase-auth12.BeY6JdLK.png>

<https://docs.weweb.io/assets/supabase-auth13.C-k5e_UJ.gif>

## 3. Link users in WeWeb to roles and users in Supabase ​

Once we've added our list of roles in Supabase and created an empty join table to link our roles with users, it's time to go to WeWeb.  
In Plugins > Supabase Auth > 3. Roles table , we'll click refresh and select the relevant Supabase tables we just created:  
  
Once you've told WeWeb where to find the roles and the join table in Supabase, you'll be able to easily view and maintain user roles in the Users tab in WeWeb:  
  
When you make a change to a user in WeWeb, it will automatically be updated in Supabase.

### Images

<https://docs.weweb.io/assets/supabase-auth14.Bnstqc8i.png>

<https://docs.weweb.io/assets/supabase-auth15.BuFR8yIB.gif>

## Users vs Profiles ​

So far, we've showed you how to work with the default users table that Supabase generates in the auth schema when you create a new project.  
Note that, for security purposes,  the information in that users table is not exposed on the auto-generated API.  
How does this affect your project in WeWeb?

## Let users update their information ​

Let's say you want to let authenticated users update their own information, then you don't need to set up anything else in Supabase.  
You could simply create a user profile page in WeWeb and display their information when they sign in, based on the data you have in the user variable:

### Images

<https://docs.weweb.io/assets/supabase-auth16.DcJpT6Eg.png>

## Display other users' information ​

In some use cases, you might want to display other users' information.  
For example, if you're building an HR portal in WeWeb, you might want HR employees to have access to a list of applicants and their user profiles.  
You wouldn't be able to do that with the users table in the auth schema because each user's information is only available to them.  
For such a use case, Supabase recommends creating a profiles table in the public schema to store user data that you want to access via the API.  
In WeWeb, you would then be able to create a Collection to get data from the profiles table.  
Learn more about managing user data in Supabase :