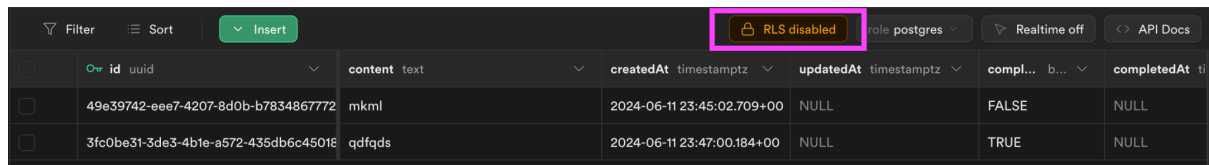


Todo List

Etape #09 : Authentification

🚩 Si lors de l'étape précédente vous avez configuré les règles de sécurité de manière à rendre les tables de données publiques, l'authentification n'est techniquement pas nécessaire pour lire et écrire des valeurs.

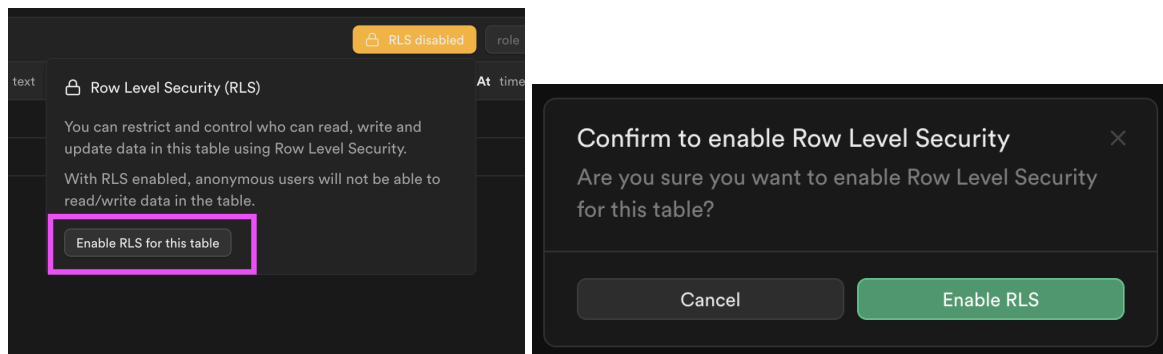


The screenshot shows a database interface for a table named 'tasks'. At the top, there is a toolbar with 'Filter', 'Sort', and 'Insert' buttons. A yellow badge with a lock icon and the text 'RLS disabled' is visible. The table has columns: id (uuid), content (text), createdAt (timestamp), updatedAt (timestamp), compl... (boolean), and completedAt (timestamp). Two rows of data are visible.

	id	content	createdAt	updatedAt	compl...	completedAt
<input type="checkbox"/>	49e39742-eee7-4207-8d0b-b7834867772	mkml	2024-06-11 23:45:02.709+00	NULL	FALSE	NULL
<input type="checkbox"/>	3fc0be31-3de3-4b1e-a572-435db6c4501e	qdfqds	2024-06-11 23:47:00.184+00	NULL	TRUE	NULL

Règles de sécurité désactivées pour la table *tasks*

Cependant, pour tester l'authentification dans des conditions normales, **il est nécessaire de limiter l'accès aux données en lecture et en écriture en adaptant les règles de sécurité sur les tables concernées, pour autoriser uniquement les utilisateurs authentifiés.**



The block contains two screenshots. The left screenshot shows the 'Row Level Security (RLS)' settings for the 'tasks' table. It explains that RLS allows restricting and controlling who can read, write, and update data. A yellow button labeled 'Enable RLS for this table' is highlighted with a red rectangle. The right screenshot is a confirmation dialog titled 'Confirm to enable Row Level Security' with the question 'Are you sure you want to enable Row Level Security for this table?'. It has two buttons: 'Cancel' and 'Enable RLS'.

Activation des règles de sécurité pour la table *tasks*

Filter

Sort

Insert

1 Auth policy

role postgres

Realtime off

API Docs

id	uuid	content	text	createdAt	timestampz	updatedAt	timestampz	compl...	b...	completedAt	ti
49e39742-eee7-4207-8d0b-b7834867772		mkml		2024-06-11 23:45:02.709+00		NULL		FALSE		NULL	
3fc0be31-3de3-4b1e-a572-435db6c45018		qdfqds		2024-06-11 23:47:00.184+00		NULL		TRUE		NULL	

schema: public 29123 Documentation

tasks Row Level Security enabled Disable RLS Create policy

SELECT Enable read access for all users

Applied to: public

Création d'une règle de sécurité pour la table *tasks*

Create a new Row Level Security policy

Policy Name: Enable read access for all users Table: public.tasks

Policy Behavior: Permissive

Policy Command: SELECT INSERT UPDATE DELETE ALL

Target Roles: authenticated

USE OPTIONS ABOVE TO EDIT

```

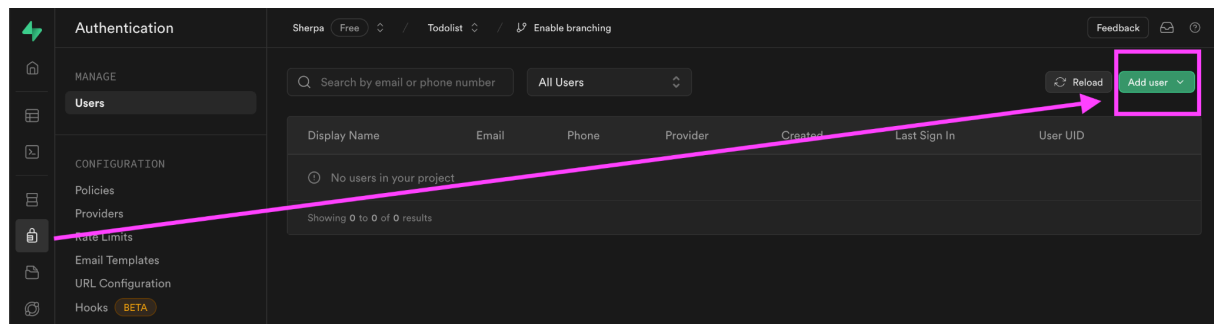
1 create policy "Enable read access for all users"
2 on "public"."tasks"
3 as PERMISSIVE
4 for ALL
5 to authenticated
6 using (
7   true
8 );

```

Use check expression

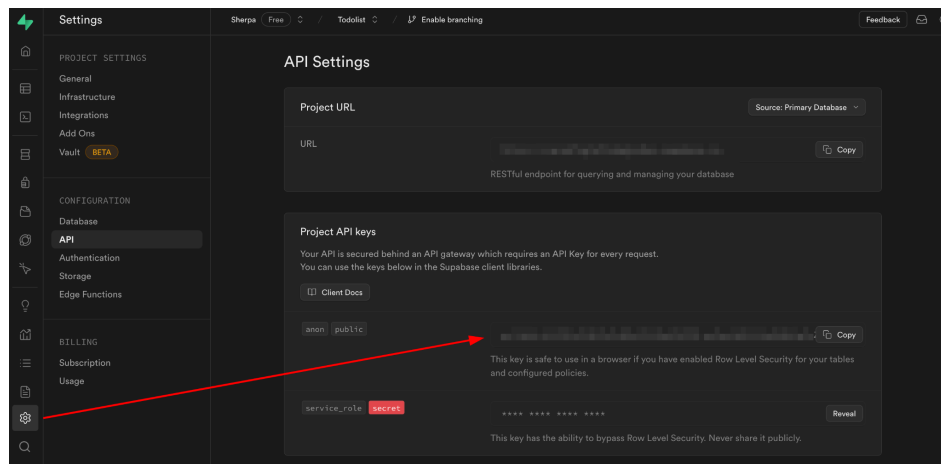
Cancel Save policy

Configuration d'une règle de sécurité indiquant que l'accès en lecture et écriture à la table *tasks* est autorisé uniquement pour les utilisateurs authentifiés



The 'Create a new user' modal is shown. It has fields for 'User Email' (john@doe.com) and 'User Password'. The 'Auto Confirm User?' checkbox is checked and highlighted with a pink box. Below it, a green 'Create user' button is visible.

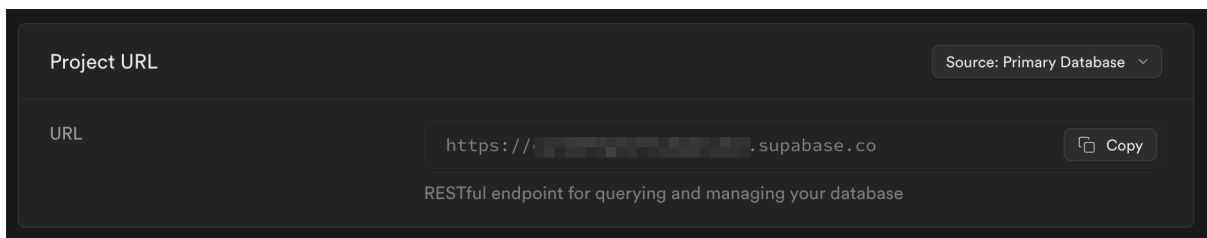
Création d'un compte utilisateur de l'API



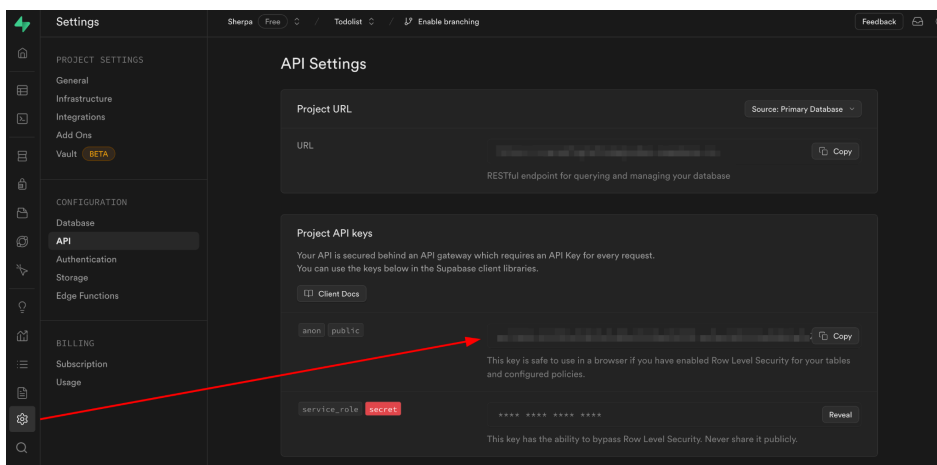
- Créez une branche **etape09** dans votre dépôt Git et passez sur cette branche.
- Créez une nouvelle classe de widget **SignIn** (`./lib/screens/signin.dart`) disposant d'un widget **Scaffold** contenant un formulaire d'authentification.
Cet écran permettra à l'utilisateur de s'authentifier auprès de l'**API REST** par l'intermédiaire d'un service nommé **AuthService**.
- Faites en sorte que l'application affiche par défaut l'écran **SignIn**.
- Un utilisateur authentifié doit disposer d'un **access_token** (JWT, cf. <https://supabase.com/docs/guides/auth/jwts>) retourné par l'**API REST** lors de l'étape d'authentification. Ce token a une durée de validité. Par conséquent, il est donc nécessaire de vérifier sa validité.
- Mettez en place une nouvelle classe Dart **AuthService** (`./lib/services/auth_service.dart`) disposant des méthodes suivantes (classées par ordre de priorité de réalisation) :
 - **signIn** (authentification par login + password, obtention d'un access_token et refresh_token),
 - **signOut** (déconnexion correspondant à l'invalidation du refresh_token),
 - **me** (obtention des données sur l'utilisateur connecté),
 - **checkToken** (vérification de la validité de l'access_token),
 - **refreshToken** (obtention d'un nouvel access_token en communiquant un refresh_token),
 - **signUp** (création de compte, méthode facultative).
- La classe **AuthService** effectuera les requêtes **HTTP** auprès de l'API en employant le package **dio**.
- Créer une classe **UserProvider** (en reprenant les principes vus avec **TasksProvider**) afin de partager les informations de l'utilisateur courant à toute l'application. Adaptez la méthode `main` (`./lib/main.dart`) afin de disposer d'un widget **MultiProvider** en tant que wrappeur de l'application **ToDoListApp**, permettant d'employer à la fois **TasksProvider** et **UserProvider** en tant que providers.
- Le widget **SignIn** emploiera la classe **AuthService** pour authentifier l'utilisateur.

- Installez le package **go_router** aux dépendances du projet (https://pub.dev/packages/go_router). Configurez des routes associées aux différents écrans de l'application.
- Redirigez automatiquement l'utilisateur authentifié vers l'écran **TasksMaster**.
- Redirigez automatiquement l'utilisateur non-authentifié vers l'écran **SignIn**.

API



Adresse de l'API Supabase



Clé de l'API Supabase

Une documentation est automatiquement générée par **Supabase** :

- pour l'**authentification** :
<https://supabase.com/dashboard/project/<supabase id>/api?page=auth>

- pour la **gestion des utilisateurs** :
https://supabase.com/dashboard/project/<supabase id>/api?page=users
- pour **chaque table de données / collection de ressources** :
https://supabase.com/dashboard/project/<supabase id>/api?resource=<nom de la table>

Sign up

🚩 Après création du compte via la requête HTTP, nécessite une validation manuelle de l'utilisateur dans le dashboard de *Supabase* pour activer l'utilisateur (onglet **Authentification**).

```
POST /auth/v1/user
```

```
Apikey: <apikey>
```

```
Host:<Supabase id>.supabase.co
```

Body : format **JSON** : `{"email":"john@doe.com", "password":"azerty"}`

Header HTTP : **apikey** / clé de l'API *Supabase*

Sign in

```
POST /auth/v1/token?grant_type=password HTTP/1.1
```

```
Apikey: <apikey>
```

```
Host:<Supabase id>.supabase.co
```

Body : format **JSON** : `{"email":"john@doe.com", "password":"azerty"}`

Header HTTP : **apikey** / clé de l'API *Supabase*

Sign out

```
POST /auth/v1/logout HTTP/1.1
```

```
Apikey: <apikey>
```

```
Host:<Supabase id>.supabase.co
```

+

Header HTTP **Authorization (Bearer)** : <access_token>

Header HTTP : **apikey** / clé de l'API **Supabase**

Refresh

```
POST /auth/v1/token?grant_type=password HTTP/1.1
```

```
Apikey: <apikey>
```

```
Host:<Supabase id>.supabase.co
```

Body : format **JSON** : `{"email":"john@doe.com", "password":"azerty"}`

Header HTTP : **apikey** / clé de l'API **Supabase**

Current user

```
GET /auth/v1/user HTTP/1.1
```

Apikey: **<apikey>**

Host: **<Supabase id>**.supabase.co

```
{"email": "<email>", "password": "<password>"}
```

+

Header HTTP **Authorization (Bearer)** : **<access_token>**

Header HTTP : **apikey** / clé de l'API **Supabase**

Read collection (tasks, tags...)

```
GET /rest/v1/tasks HTTP/1.1
```

Apikey: **<apikey>**

Host: **<Supabase id>**.supabase.co

+

Header HTTP **Authorization (Bearer)** : **<access_token>**

Header HTTP : **apikey** / clé de l'API **Supabase**

Read item (tasks, tags...)

```
GET /rest/v1/tasks?id=<task id>&select=* HTTP/1.1
```

Apikey: **<apikey>**

Host: **<Supabase id>**.supabase.co

+

Header HTTP **Authorization (Bearer)** : **<access_token>**

Header HTTP : **apikey** / clé de l'API **Supabase**