1 La méthode OAuth

1.1) Explication :

La méthode OAuth permet aux utilisateurs de se connecter à une application en utilisant leurs comptes existants sur différents fournisseurs d'identité.

On peut ainsi se connecter grâce à son compte google, facebook, twitter, instagram et encore bien d'autres.

Cette méthode est pratique car elle évite aux utilisateurs de devoir mémoriser de nouveaux identifiants. La sécurité est également renforcée via cette méthode, car les informations d'identification sont gérées par le fournisseur d'identité directement, et non par l'application tierce.

1.2) Tutoriel :

Etape 1 : Création d'un identifiant OAuth :

Cette étape est une étape préparatoire pour la suite. On crée un identifiant client pour notre application à travers la console API Google ; ce qui va nous permettre d'utiliser les services de google dans notre application. En l'occurrence, ici nous cherchons juste à pouvoir nous connecter grâce au système d'authentification de google.

- Cliquer sur ce lien : https://developers.google.com/identity/gsi/web/guides/get-google-api-clientid?hl=fr
- Ouvrir le lien (console des API Google)
- Cliquer sur tutoriel → Créer un projet
- Aller dans identifiants → Créer des identifiants → ID client OAuth
- Configuration écran de consentement :

  - Cocher externe puis suivre la configuration

- Aller dans identifiants → Créer des identifiants → ID client OAuth

  - Dans Origines Javascript autorisées, rentrer le lien du serveur

  - Dans URI de redirection autorisés rentrer le lien vers lequel on veut être redirigé après avoir cliqué sur le bouton de connexion

Etape 2 : Charger la librairie JS Google :

Ici, nous allons copier coller le code javascript et html déjà préconçu par google pour la connexion google. Après avoir intégré ce code dans notre application, la connexion côté client sera déjà fonctionnelle, mais il restera une dernière étape.

- Cliquer sur ce lien : https://developers.google.com/identity/gsi/web/guides/client-library?hl=fr
- Copier le script JS et aller le coller sur la page de connexion

```
62
63
64   <script src="https://accounts.google.com/gsi/client" async></script>
```

- Cliquer sur le lien : https://developers.google.com/identity/gsi/web/tools/configurator?hl=en
- Rentrer l'ID client précédemment obtenu puis l'URI de connexion
- Choisir le type de bouton qu'on veut puis faire obtenir le code
- Coller le code sur la page de connexion avant le script JS

```
<div class="offset-5">

    <div id="g_id_onload"
      data-client_id="266973138588-hab87f1c0cc4i9jt3el82f8ajcqnpp32.apps.googleusercontent.com"
      data-context="signin"
      data-ux_mode="popup"
      data-login_uri="https://sohappynas.synology.me/sohappy-heb/index.php?uc=accueil&action=accueil"
      data-auto_prompt="false">
    </div>

    <div class="g_id_signin"
      data-type="standard"
      data-shape="rectangular"
      data-theme="outline"
      data-text="signin_with"
      data-size="large"
      data-logo_alignment="center"
      data-width="400px">
    </div>

</div>
```

Etape 3 : Traitement PHP de la connexion :

A cette étape là on peut déjà se connecter via notre compte google, mais on va vérifier côté serveur la tentative de connexion pour éviter le piratage.

- Cliquer sur ce lien : https://github.com/googleapis/google-api-php-client.

Le dépôt d'un SDK sur github c'est-à-dire d'un ensemble de fonctions déjà codées pour nous permettre de traiter la connexion côté serveur existe déjà. On va donc l'implémenter dans notre code.

- Installer le code avec composer dans un terminal de VS Code à l'aide du lien fourni sur github.

- Vérifier sur VS Code dans composer.json que la librairie est bien implémentée

- Ecrire ce code dans index.php pour charger les dépendances et pouvoir ensuite utiliser le SDK dans notre code :

```php
<?php

require('vendor/autoload.php');
```

- cliquer : https://developers.google.com/identity/gsi/web/guides/verify-google-id-token?hl=fr.

Grâce à ce lien on va récupérer le code qui permet de valider le jeton d'ID google côté serveur. Le code est préconçu pour Java, Node.js, PHP et Python. Dans notre cas nous allons prendre celui pour PHP et nous allons l'introduire dans le fichier php où l'utilisateur est redirigé après sa connexion, puis nous allons le modifier.

```php
<?php

if (!empty($_POST['credential'])){

    if(empty($_COOKIE['g_csrf_token']) || empty($_POST['g_csrf_token']) || $_COOKIE['g_csrf_token'] != $_POST['g_csrf_token']){
        echo "Erreur vérification jeton CSRF";
        exit();
    }
```

'credential' est l'identifiant envoyé par google vers l'application qui permet de valider l'authentification. Puis un token est généré par google après la connexion. On va donc vérifier que le token généré par google et celui qu'on a dans les cookies est le même pour vérifier l'intégrité et la provenance de la connexion. Dans le code ci-dessus, on traite tous les cas où la connexion ne doit pas avoir lieu et on envoie un message d'erreur dans le cas où l'une de ces conditions est vérifié.

```php
        $clientId = "266973138588-hab87f1c0cc4i9jt3el82f8ajcqnpp32.apps.googleusercontent.com";
        $client = new Google_Client(['client_id' => $clientId]);
        $idToken = $_POST['credential'];
        $user = $client->verifyIdToken($idToken);
```

Ici, on crée un objet avec notre ID Client qu'on stocke dans une variable pour pouvoir nous servir des méthodes de la classe Google_Client qui permet d'intéragir avec les API Google et notament le service google qui permet de gérer l'authentification. Ensuite la méthode verifyIdToken permet à l'application de vérifier l'authenticité et l'intégrité du jeton d'identification fourni par google , ce qui est essentiel pour garantir la sécurité de l'authentification des utilisateurs.

```
    if ($user) {
        $_SESSION['user'] = $user;
        header('location:https://sohappynas.synology.me/sohappy-heb/index.php?uc=accueil&action=accueil');
        exit();
    } else {

        echo "Erreur lors de l'authentification";
    }

}
```

Ici si l'authentification du jeton d'identification a réussie, on stocke $user dans une session comme ça lorsqu'il se déconnecte, l'utilisateur doit se reconnecter pour de nouveau avoir accès à l'application.

Puis on redirige l'utilisateur vers la page d'accueil de l'application.

Si l'authentification a échouée, on affiche un message d'erreur.

1.3.) Code Source

Trois fichiers sont importants :

- v_connexion.php
- v_accueil.php
- index.php

Ces fichiers sont dans le dossier sohappy-heb sur le serveur web.