**PARTIE 1 : Protection contre les tentatives de force brute**

Pour compléter la protection contre les tentatives de piratage des comptes par force brute, il a été décidé de limiter le nombre de tentatives successives échouées de connexion à 5 maximum. Au-delà de ce seuil, le compte est désactivé, et un mail est envoyé à l’utilisateur avec un lien à usage unique pour débloquer son compte. Le lien de réactivation que recevra l’utilisateur par mail prendra la forme suivante : <http://ip_serveur_web/reactivation?code=UnCodeGenereAUsageUnique>

Dès que l’utilisateur aura cliqué sur le lien, on affichera un message de confirmation de la réactivation de son compte.

Pour gérer cela, il faudra donc mémoriser dans la base de données ce code qui sera généré aléatoirement au moment de sa création, sous la forme d’une suite de 32 caractères alphanumériques, associé au compte utilisateur concerné. Le lien généré expire au bout de 24 heures après sa création. Autrement dit, il faudra enregistrer la date et l’heure d’expiration du lien.

Lorsque la connexion est réussie, il faudra sauvegarder en variable SESSION l’identifiant du compte utilisateur pour la partie A2F qui sera gérée plus tard. **Vous appellerez cette variable SESSION « connexion ».**

Pour rappel : toute tentative de connexion (échec ou réussite) devra être notifié dans les logs (cf fiche 1).

***Documentation annexe :***

* A propos de l’attaque par force brute : <https://www.cnil.fr/fr/definition/force-brute-attaque-informatique>

***AVANT DE FAIRE LES QUESTIONS : vous installerez la librairie PHPMailer via la commande : composer require phpmailer/phpmailer***

**TRAVAIL A FAIRE :**

1. Rappelez en quoi consiste une attaque par force brute.

Une attaque par force brute consiste à tester l’une après l’autre toutes les combinaison possible pour trouver un mot de passe.

1. Proposer une modification de la base de données existante afin de permettre la gestion des comptes désactivés et des liens de réactivation des comptes.

Il faut créer une table réactivation et une ta ble réactivation.

1. Coder la partie authentification : vérification du couple email/mot de passe, enregistrement dans les logs et comptabilisation du nombre de tentatives échouées, création de la variable session si authentification réussie. Veillez à afficher le nombre de tentatives restantes de connexions en cas d’échec. Pensez à gérer l’envoi du mail avec le lien de réactivation dès le blocage du compte.

***Pour l’envoi de mails, utilisez la méthode statique envoyerEmail du contrôleur Email.***

1. Coder la partie réactivation du compte utilisateur. Veillez à bien vérifier la validité du code passé dans l’URL avant d’afficher la page. On supprimera le code de réactivation dans la base une fois le compte réactivé. N’oubliez pas les logs!

**PARTIE 2 : Authentification à double facteur (A2F)**

Il est maintenant temps de gérer la double authentification d’un compte utilisateur.

Le formulaire de connexion est déjà établi, reste à développer la partie vérification du couple login/mot de passe puis la partie vérification du code envoyé par l’application d’authentification. Pour rappel, la gestion de cette vérification passera par la librairie pragmarx/google2fa.

***Documentation annexe :***

* Documentation de la librairie PHP pragmarx/google2fa : <https://github.com/antonioribeiro/google2fa/blob/8.x/README.md#usage>
* Documentation complémentaire pour la librairie PHP pragmarx/google2fa :  
  <https://medium.com/@richb_/easy-two-factor-authentication-2fa-with-google-authenticator-php-108388a1ea23>

**TRAVAIL A FAIRE :**

1. Coder la partie vérification du code de l’application A2F. Une fois le code validé, il faudra détruire la variable SESSION (on s’occupera de la suite dans la partie suivante). On devra gérer également le cas où le code est incorrect en retournant un message d’erreur sur la page d’authentification à double facteur.

**PARTIE 3 : JSON Web Token (JWT)**

Lorsque la connexion aura réussi, on génèrera un JSON Web Token (JWT) qui sera généré via la librairie firebase/php-jwt. Ce JWT sera ensuite renvoyé à l’utilisateur sous la forme d’un cookie qui sera valable pendant 30 jours. **Vous nommerez ce cookie « auth ».**

Le JWT devra stocker dans son payload, les données suivantes :

**{**

**"name": "email\_de\_l\_utilisateur",**

**"sub": "l\_ID\_de\_l\_utilisateur",**

**"iat": date\_heure\_creation\_du\_jwt\_sous\_la\_forme\_d\_un\_timestamp**

**}**

Pour chiffrer et déchiffrer le JWT dans le cookie utiliserez la clé suivante : T3mUjGjhC6WuxyNGR2rkUt2uQgrlFUHx

***Documentation annexe :***

* A propos des cookies (CNIL) : <https://www.cnil.fr/fr/definition/cookie>
* A propos des timestamps – Page Wikipédia : <https://fr.wikipedia.org/wiki/Horodatage>
* Fonction setcookie – Documentation PHP : <https://www.php.net/manual/fr/function.setcookie.php>
* Variables $\_COOKIE – Documentation PHP : <https://www.php.net/manual/fr/reserved.variables.cookies.php>
* Documentation officielle de la librairie firebase/php-jwt : <https://github.com/firebase/php-jwt/blob/main/README.md>

**TRAVAIL A FAIRE :**

1. Qu’est-ce qu’un cookie et à quoi sert-il dans le cas d’une connexion pour un utilisateur ?

Un **cookie** est un petit fichier de texte stocké sur le navigateur d’un utilisateur par un site web. Il contient des données que le serveur peut récupérer lors des visites ultérieures de l'utilisateur. Ces données permettent de maintenir des informations entre les requêtes HTTP.

Lorsqu'il s'agit d'une **connexion utilisateur**, les cookies sont utilisés pour gérer l'authentification et maintenir l'état de session entre les requêtes.

1. Que représente un timestamp ? Recherchez la fonction permettant de générer un timestamp en PHP.

Un **timestamp** représente un moment précis dans le temps, généralement exprimé sous forme d'un nombre entier, qui indique le nombre de secondes écoulées depuis une référence spécifique dans le passé.

C’est la fonction **time()** qui permet de générer un timestamp

Exemple : $timestamp\_actuel = time();

1. Installer et configurer la librairie firebase/php-jwt via composer.
2. Modifier la partie connexion faite précédemment afin de générer un JWT une fois le code de l’application A2F est validé. N’oubliez pas de renvoyer le JWT sous la forme d’un cookie que l’on nommera « auth ». On redirigera enfin l’utilisateur connecté sur la page de son profil.