Documentation technique

TOTP

Matthieu Campagna

11/12/2025

Sommaire

Présentation du projet

Ce projet, réalisé dans le cadre de la deuxième année du BTS SIO, consiste à développer un générateur de codes de sécurité à usage unique.

L'objectif principal est d'approfondir les notions fondamentales de la cybersécurité en mettant en œuvre un système d'authentification à deux facteurs (2FA). Concrètement, l'application génère un code à usage unique qui se renouvelle automatiquement à chaque connexion.

Cette solution permet de renforcer la sécurité des accès en ajoutant une couche de sécurité supplémentaire, rendant le vol de mot de passe insuffisant pour compromettre un compte. Le projet explore ainsi des concepts clés tels que les algorithmes de hachage et la synchronisation avec la base de données.

Arborescence des Fichiers

```
/tp-totp
│
├── config
│   └── database.php            (Connexion à la base de données)
│
├── controller
│   └── RegisterController.php          (Contrôleur : Logique de traitement)
│
├── models
│   └── TOTPmodel.php                   (Modèle : Gestion des données)
│
├── view
│   ├── footer.php              (Pied de page commun)
│   ├── header.php              (En-tête commun)
│   ├── index.php               (Page d'accueil)
│   ├── login.php               (Interface de connexion)
│   ├── logout.php              (Script de déconnexion)
│   └── register.php            (Interface d'inscription)
│
├── bdtotp.sql          (Script SQL de la base de données)
└── README.MD           (Documentation d’installation)
```

Config

database.php

```
<?php
// Variables de connexion à la base de données
$host = 'localhost';
$dbname = 'bdtotp';
$username = 'myroot';
$password = 'root123*';

// Création d'une connexion PDO
$pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username, $password);
```

Ce code PHP sert à se connecter à une base de données MySQL. Il définit d'abord les informations nécessaires pour la connexion :

- l'adresse du serveur (localhost)
- le nom de la base (bdtotp)
- le nom d'utilisateur (myroot)
- le mot de passe (root123*)

Une instance PDO est créée pour gérer la connexion à la base de données, permettant l'interaction et l'exécution des commandes SQL.

Controller

RegisterController.php

```
<?php
require '../models/TOTPmodel.php';

if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $prenom = $_POST['firstname'];
    $nom = $_POST['lastname'];
    $email = $_POST['email'];
    $password = $_POST['password'];

    if (!$prenom || !$nom || !$email || !$password) {
        echo "Veuillez remplir tous les champs.";
        exit;
    }

    if (getRegister($prenom, $nom, $email, $password)) {
        header("Location: ../view/login.php");
        exit;
    } else {
        echo "Cet email est déjà utilisé.";
        exit;
    }
}
```

Ce code PHP gère l'inscription d'un utilisateur. Il commence par inclure le modèle TOTPmodel.php qui contient les fonctions pour interagir avec la base de données. Puis, il vérifie si le formulaire a été envoyé en méthode POST et récupère les informations saisies :

- prénom
- nom
- email
- mot de passe

Il vérifie que tous les champs sont remplis et affiche un message d'erreur si ce n'est pas le cas. Ensuite, il appelle la fonction getRegister pour enregistrer l'utilisateur. Si l'enregistrement réussit, il redirige vers la page de connexion. Sinon, il affiche un message indiquant que l'email est déjà utilisé.

Models

TOTPmodel.php

```
function generateTotpSecret()
{
    return str_pad(rand(0, 999999), 6, '0', STR_PAD_LEFT);
}
```

Cette fonction PHP génère un code TOTP à 6 chiffres.

Elle utilise rand(0, 999999) pour créer un nombre aléatoire entre 0 et 999999, puis str_pad complète avec des zéros à gauche si le nombre a moins de 6 chiffres.

Le résultat est toujours une chaîne de 6 chiffres, utilisée comme code pour l'authentification.

```
function getRegister($prenom, $nom, $email, $password)
{
    global $pdo;

    $stmt = $pdo->prepare("SELECT * FROM utilisateur WHERE email = ?");
    $stmt->execute([$email]);
    if ($stmt->fetch()) {
        return false;
    }

    $totp_secret = generateTotpSecret();
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);

    $stmt = $pdo->prepare(
        "INSERT INTO utilisateur (email, password, totp_secret, created_at, nom, prenom)
         VALUES (?, ?, ?, NOW(), ?, ?)"
    );
    $success = $stmt->execute([$email, $hashed_password, $totp_secret, $nom, $prenom]);
    return $success;
}
```

Cette fonction PHP enregistre un nouvel utilisateur dans la base de données.

Le code vérifie d'abord si l'adresse email est déjà présente dans la base de données si c'est le cas, la fonction retourne false.

Sinon, elle génère un code secret TOTP avec generateTotpSecret() et sécurise le mot de passe avec password_hash().

Ensuite, elle insère les informations de l'utilisateur (email, mot de passe, secret TOTP, prénom, nom et date de création) dans la table utilisateur.

Enfin, elle renvoie true si l'insertion a réussi, ou false sinon.

```
function updateUserTotp($userId)
{
    global $pdo;

    $newTotp = str_pad(rand(0, 999999), 6, '0', STR_PAD_LEFT);

    $update = $pdo->prepare("UPDATE utilisateur SET totp_secret = ? WHERE id = ?");
    $update->execute([$newTotp, $userId]);

    return $newTotp;
}
```

Cette fonction PHP met à jour le code TOTP d'un utilisateur existant.

Elle crée un nouveau code aléatoire à 6 chiffres avec rand() et str_pad, puis met à jour le champ totp_secret dans la table utilisateur pour l'utilisateur correspondant à l'id fourni.

Enfin, elle retourne le nouveau code TOTP généré.

```
function getLogin()
{
    global $pdo;

    if ($_SERVER["REQUEST_METHOD"] === "POST") {
        $email = $_POST['email'];
        $password = $_POST['password'];
        $totp = $_POST['totp'];

        $stmt = $pdo->prepare("SELECT * FROM utilisateur WHERE email = ?");
        $stmt->execute([$email]);
        $user = $stmt->fetch();

        if (!$user) {
            echo "Aucun compte n'a été trouvé avec cet email.";
            return;
        }

        if (!password_verify($password, $user['password'])) {
            echo "Mot de passe incorrect.";
            return;
        }

        if ((string)$totp !== (string)$user['totp_secret']) {
            echo "Code TOTP incorrect.";
            return;
        }

        $_SESSION['user_id'] = $user['id'];
        $_SESSION['email'] = $user['email'];
        $_SESSION['totp'] = $totp;

        $newTotp = updateUserTotp($user['id']);
        $_SESSION['totp'] = $newTotp;

        header("Location: index.php?prenom=" . urlencode($user['prenom']) . "&nom=" .
urlencode($user['nom']));

        exit;
    }
```

Cette fonction PHP gère la connexion d'un utilisateur avec TOTP lorsqu'un formulaire est envoyé en POST.

Elle récupère l'email, le mot de passe et le code TOTP saisis puis elle cherche l'utilisateur dans la base de données avec l'email si aucun compte n'est trouvé.

Elle affiche un message d'erreur, et vérifie ensuite que le mot de passe soit correct avec password_verify() et que le code TOTP corresponde au code stocké.

Si tout est valide, elle crée des variables de session pour l'utilisateur, génère un nouveau code TOTP avec updateUserTotp() et redirige vers la page d'accueil en passant le prénom et le nom de l'utilisateur dans l'URL.

View

footer.php

```
</body>

</html>
```

Ce fichier footer.php contient simplement la fin d'une page HTML : il ferme les balises </body> et </html>.

Il sert à être inclus à la fin de toutes les pages pour terminer correctement le document HTML.

header.php

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="./public/css/style.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.8/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-sRIl4kxILFvY47J16cr9ZwB07vP4J8+LH7qKQnuqkuIAvNWLzeN8tE5YBujZqJLB" crossorigin="anonymous">
    <title>Projet TOTP</title>
</head>
```

Ce fichier header.php contient le début d'une page HTML.

Il définit le type de document (<!DOCTYPE html>), la langue (fr) et l'encodage (UTF-8).

Il inclut également :

- Un meta viewport pour que la page soit responsive sur mobile.
- Une feuille de style locale (style.css).
- La bibliothèque Bootstrap pour le design et les composants.

Enfin, il définit le titre de la page avec <title>Projet TOTP</title>.

Ce fichier est inclus au début de chaque page pour standardiser l'en-tête HTML.

index.php

```
<?php
require '../view/header.php';

$prenom = $_GET['prenom'] ?? 'Utilisateur';
$nom = $_GET['nom'] ?? '';
?>

<body class="bg-light d-flex justify-content-center align-items-center vh-100">
    <div class="card p-4 text-center" style="width: 100%; max-width: 400px;">
        <h2 class="mb-3">Bonjour</h2>
        <h3><?=($prenom . ' ' . $nom) ?></h3>
        <p class="text-muted">Bienvenue sur votre espace utilisateur.</p>
        <a href="logout.php" class="btn btn-danger mt-3 w-100">Se déconnecter</a>
    </div>

    <?php
    require '../view/footer.php';
    ?>
```

Ce fichier index.php affiche la page d'accueil après la connexion.

Il inclut d'abord header.php pour le début du HTML.

Il récupère le prénom et le nom de l'utilisateur depuis l'URL ($_GET) ou utilise des valeurs par défaut.

Dans le <body>, il affiche une carte centrée avec un message de bienvenue et le nom complet de l'utilisateur.

Il propose un bouton de déconnexion qui redirige vers logout.php.

Enfin, il inclut footer.php pour terminer la page HTML.

En résumé, c'est la page utilisateur principale après connexion.

login.php

```
<?php
require '../view/header.php';
require '../models/TOTPmodel.php';
?>

<body class="bg-white d-flex justify-content-center align-items-center vh-100">

    <div class="card p-4" style="width: 100%; max-width: 400px;">
        <h2 class="text-center mb-4">Connexion utilisateur</h2>

        <form method="POST" action="login.php">
            <div class="mb-3">
                <label for="email" class="form-label">Adresse e-mail</label>
                <input type="email" class="form-control" name="email" id="email"
placeholder="exemple@domaine.com" required>
            </div>

            <div class="mb-3">
                <label for="password" class="form-label">Mot de passe</label>
                <input type="password" class="form-control" name="password" id="password"
placeholder="••••••••" required>
            </div>

            <div class="mb-3">
                <label for="totp" class="form-label">Code TOTP</label>
                <input type="text" class="form-control" name="totp" id="totp" placeholder="Entrez le
code TOTP" required>
            </div>

            <button type="submit" class="btn btn-success w-100">Se connecter</button>
        </form>

        <?php
        // Traiter le formulaire de connexion
        getLogin();
        ?>

        <div class="text-center mt-2">
            <a href="./register.php" class="d-block mb-1">Créer un compte</a>
        </div>

    </div>

    <?php require '../view/footer.php'; ?>
```

Ce fichier login.php affiche le formulaire de connexion pour les utilisateurs.

Il inclut header.php pour le début du HTML et TOTPmodel.php pour les fonctions liées à la base de données et au TOTP.

Le formulaire demande l'email, le mot de passe et le code TOTP et envoie les données en POST vers login.php.

Après l'envoi du formulaire, la fonction getLogin() traite la connexion : vérification de l'email, du mot de passe et du code TOTP, création des sessions et redirection vers la page d'accueil.

Il propose aussi un lien vers register.php pour créer un compte.

Enfin, footer.php est inclus pour terminer la page HTML.

En résumé, c'est la page où l'utilisateur se connecte avec TOTP.

logout.php

```
<?php
session_start();
session_unset(); // Supprimer les données de la session
session_destroy(); // Supprime la session
header('Location: login.php');
exit;
```

Ce fichier logout.php gère la déconnexion de l'utilisateur.

- session_start() démarre la session
- session_unset() supprime toutes les variables de la session
- session_destroy() détruit la session elle-même

En résumé, il redirige l'utilisateur vers la page de connexion (login.php).

register.php

```
<?php
require '../view/header.php';
?>

<body class="bg-white d-flex justify-content-center align-items-center vh-100">

    <div class="card p-4" style="width: 100%; max-width: 400px;">
        <h2 class="text-center mb-4">Créer un compte</h2>

        <form method="POST" action="../controller/RegisterController.php">
            <div class="mb-3">
                <label for="firstname" class="form-label">Prénom</label>
                <input type="text" class="form-control" id="firstname" name="firstname"
placeholder="Votre prénom" required>
            </div>

            <div class="mb-3">
                <label for="lastname" class="form-label">Nom</label>
                <input type="text" class="form-control" id="lastname" name="lastname"
placeholder="Votre nom" required>
            </div>

            <div class="mb-3">
                <label for="email" class="form-label">Adresse e-mail</label>
                <input type="email" class="form-control" id="email" name="email"
placeholder="exemple@domaine.com" required>
            </div>

            <div class="mb-3">
                <label for="password" class="form-label">Mot de passe</label>
                <input type="password" class="form-control" id="password" name="password"
placeholder="••••••••" required>
            </div>

            <button type="submit" class="btn btn-success w-100">Créer un compte</button>
        </form>

        <div class="text-center mt-2">
            <a href="./login.php" class="d-block mb-1">Déjà un compte ? Se connecter</a>
        </div>

    </div>

    <?php require '../view/footer.php'; ?>
```

Ce fichier register.php affiche le formulaire d’inscription pour créer un compte utilisateur.

Il inclut header.php pour le début du HTML.

Le formulaire demande prénom, nom, email et mot de passe et envoie les données en POST vers RegisterController.php, qui gère l’enregistrement dans la base de données.

Un bouton permet de soumettre le formulaire pour créer le compte.

Un lien en bas permet aux utilisateurs déjà inscrits d’accéder à la page de connexion (login.php).

Enfin, footer.php est inclus pour fermer correctement la page HTML.

En résumé, c’est la page où un nouvel utilisateur peut s’inscrire.

Guide d’Installation

Pour déployer le projet localement, veuillez suivre les étapes suivantes :

1. Base de données

- Ouvrir phpMyAdmin.
- Créer une nouvelle base de données.
- Importer le fichier SQL fourni à la racine du projet (bdtotp.sql).

2. Configuration

- Ouvrir le fichier de configuration config/database.php.
- Mettre à jour les variables $username et $password avec les identifiants de votre serveur local (WAMP/XAMPP).

3. Installation de l'environnement

- Ouvrir Visual Studio Code.
- Se rendre dans l'onglet Extensions et installer Live Server.

4. Lancer le projet

- Ouvrir le dossier complet du projet dans VS Code.
- Faire un clic droit sur la page view/login.php.
- Sélectionner l'option Open with Live Server.