SLH 24
Lab #2

5 décembre 2024

— Laboratoire noté.
— Veuillez rendre **votre code** et un **rapport** répondant aux questions de la Section 2.
— La qualité du code est notée.
— Le code doit obligatoirement être écrit en Rust.
— La **validation des entrées** est primordiale.
— Nous nous attendons à ce que vous testiez votre code.
— Vous trouverez dans le code fourni les fichiers à remplir. La partie frontend est déjà fournie dans son entièreté.
— Veillez à utiliser `cargo clean` avant de rendre votre code.
— La crate `webauthn-rs` nécessite d'avoir `openssl` d'installé.
— **Ne pas modifier la version des dépendances de cargo.toml**. Vous pouvez cependant ajouter des crates si nécessaire.
— Il est normal que les photos publiées ne s'affichent pas.
— Le fichier `.gitignore` contient les dossiers / fichiers à ne pas inclure dans le rendu.

1 Description

Le but de ce laboratoire est de gérer l'authentification d'un site web. L'authentification doit être gérée avec des *passkeys* [1] [2]. Les fonctionnalités du site sont les suivantes :
— Création d'un compte.
— Connexion au compte.
— Récupération du compte.
— Publier une image et une courte description.

2 Questions

1. Les passkeys, comme l'authentification unique (SSO), permettent toutes deux à un utilisateur d'utiliser un seul mécanisme de sécurité pour tous ses services, sans s'exposer à une attaque de "credentials stuffing". Mais quelle est la différence en termes de risque ?
2. Concernant la validation d'entrées pour les images, quelles sont les étapes que vous avez effectuées ? Et comment stockez vous les images ?
3. Que pensez-vous de la politique de choix des noms de fichiers uploadés ? Y voyez-vous une vulnérabilité ? Si oui, suggérez une correction.

1. https://www.passkeys.com/fr/passkey-cest-quoi
2. https://crates.io/crates/webauthn-rs

3 Tâche principale

1. Complétez les **TODOs** dans `backend/handlers_unauth.rs` et `utils/webauthn.rs`.
2. Gérez la validation des entrées et stockez correctement les images postées par les utilisateurs dans `backend/handlers_auth.rs`. Nous acceptons seulement les fichiers `.jpg` et la crate `image`[3].
3. **Ne modifiez pas** `backend/router.rs` et `backend/model.rs`.
4. Libre à vous d'ajouter des modules dans `utils`. Ajoutez également le nom des modules ajoutés dans le fichier `utils.rs`.
5. La base de données est simulée avec des fichiers `.yaml`.

4 Explications sur le template

La plupart du serveur est déjà implémenté. La librairie principalement utilisées est `axum`[4], elle permet de gérer les endpoints, la gestion de la "database" et les fichiers `.hbs` pour le HTML et le javascript.

4.1 axum

Le serveur est démarré dans le `main.rs` et fonctionne sur `localhost:8080`. Deux routes principales sont définies dans `src/backend/router.rs`.

```rust
/// Routes accessibles sans authentification
fn auth_routes() -> Router {
    Router::new() {
        // Liste des routes
    }
}

/// Routes accessibles avec authentification
fn unauth_routes() -> Router {
    Router::new() {
        // Liste des routes
    }
}
```

Dans l'application, les routes sont divisées en deux catégories principales :
- **Routes sans authentification** (`unauth_routes`) : Ces routes sont accessibles à tous les utilisateurs, qu'ils soient connectés ou non. Elles permettent de gérer des actions comme l'inscription, la connexion, la validation de compte, ou encore la récupération de compte.
- **Routes nécessitant une authentification** (`auth_routes`) : Ces routes sont réservées aux utilisateurs authentifiés. Elles permettent d'accéder aux fonctionnalités principales de l'application, comme la gestion des posts (création, suppression, interaction).

Les routes nécessitant une authentification (`auth_routes`) intègrent un middleware (`SessionUser`) pour vérifier que l'utilisateur est bien connecté avant de lui permettre d'accéder aux fonctionnalités protégées.

3. https://crates.io/crates/image
4. https://docs.rs/axum/latest/axum/index.html

4.2 Handlers

Les handlers associés aux routes sont des fonctions qui permettent de récupérer différents attributs d'une requête. Par exemple :

- `Json(params)` : Permet de récupérer le corps de la requête au format JSON et de le convertir en un objet de type T. Utile pour traiter les données envoyées via une requête POST ou PUT.
- `Path(params)` : Permet de récupérer les segments dynamiques de l'URL. Par exemple, pour une route `/user/:id`, `Path` permet d'extraire l'`id`.
- `Query(params)` : Permet de récupérer les paramètres de requête passés dans l'URL (par exemple, `?key=value`) et de les mapper dans une structure ou une table de hachage.
- `Multipart` : Permet de gérer les fichiers/formulaires envoyés via une requête `multipart/form-data`, comme pour l'upload de fichiers.
- `Extension(params)` : Permet de partager des objets globaux comme une base de données, un moteur de templates, ou d'autres extensions nécessaires à l'application.

5 Comment utiliser les Passkeys

Dans ce laboratoire, vous serez amenés à créer et utiliser des passkeys pour tester votre application. Voici un aperçu non exhaustif des outils compatibles :

Navigateurs compatibles

- **Google Chrome/Brave**[5] : Compatibilité complète sur Windows, macOS, Linux, Android et iOS.
- **Microsoft Edge** : Fonctionne avec Windows Hello pour l'authentification biométrique ou via PIN.
- **Safari** : Intégration native avec Apple Passkeys sur macOS et iOS, synchronisation via iCloud.
- **Mozilla Firefox** : Non compatible pour l'instant, sauf avec un token hardware (YubiKey par exemple).

Systèmes d'exploitation compatibles

- **Windows** : Support via Windows Hello (empreinte digitale, reconnaissance faciale, PIN).
- **macOS et iOS** : Gestion native des passkeys via le trousseau iCloud.
- **Linux** : Utilisation possible via des gestionnaires tiers tels que Bitwarden.

Applications tierces compatibles

Certains gestionnaires de mots de passe permettent de gérer les passkeys sur les différents navigateurs.

- **Bitwarden** : Compatible avec ce projet.
- **Proton Pass** : Non compatible avec ce projet.
- **Autres** : Non testés.

5. https://developer.chrome.com/docs/devtools/webauthn?hl=fr