Rapport TP Archiapp

Liens

Projet GitHub: Ishirui/tp-archiapp

Version en ligne: tp-archiapp.veyrenc.xyz

Fonctionnement général et déploiement

Organisation générale

Tout le site est basé sur un serveur écrit avec la librarie ExpressJS. En utilisant le middleware `static`, le serveur est également capable de renvoyer les fichiers du front-end sur la route `/`.

Ces fichiers pour le front sont dans le dossier `client/`, avec le triplet standard: un `index.html`, un peu d'interactivé grâce au `script.js` et le style grâce au fichier `style.css`. Le code serveur est stocké

Toute la stylisation a été réalisée "à la main", sans bibliothèques comme Tailwind par souci de simplicité. La fonctionnalité "dark mode" est réalisée purement en CSS, en ajoutant une pseudo-classe `darkMode` via un peu de JS lorsqu'on appuie sur le bouton. Une règle CSS change les couleurs des éléments ayant cette sous-classe.

API

Format des messages

```
{
    msg: string,
    time: string, // Au format ISO 8601
    author: string
}
```

> Les champs `time` sont transformés en objet `Date` à la fois sur le client et le serveur, pour les manipuler plus facilement.

Les messages sont stockés dans un simple tableau en mémoire côté serveur. Pour plus de robustesse (actuellement ils disparaissent tous lorsque le serveur redémarre !), il faudrait les stocker dans une BDD SQL, type SQLite par exemple.

Routes

Je me suis un peu écarté du format proposé dans le TP pour se rapprocher d'une API REST plus classique:

- les erreurs sont indiquées par des codes HTTP en `4xx` ou `5xx` plutôt qu'un champ `code`
- les nouveaux messages sont envoyés dans le corps d'une requête POST plutôt que simplement dans l'URL.
- la route `/get` prend un paramètre d'url (`/get?id=xxx`) plutôt qu'un autre segment pour identifier le message à récupérer.

Déploiement

J'ai choisi de déployer le site sur mon NAS/serveur personnel plutôt que d'utiliser un service tiers (pourquoi pas !). Pour cela j'ai simplement eu à créer une image de conteneur pour le site et un rapide fichier compose pour le déploiement.

Vulnérabilités et points d'amélioration

- Il y a une énorme vulnérabilité de XSS au niveau du front: le contenu des messages est intégré au DOM de la page sans la moindre sanitisation ou contrôle. Il serait très facile pour un attaquant d'injecter un *payload* pour exécuter du code sur les navigateurs des autres utilisateurs du site.
- On pourrait améliorer la performance du front: actuellement, à chaque *refresh*, le site extrait la liste totale des messages depuis l'API et re-dessine l'intégralité des messages. Il suffirait de récupérer et d'afficher les *nouveaux* messages.