Documentation Technique – Chrono Explorer

1. Présentation du projet

Chrono Explorer est une application web Angular dédiée à la découverte d'événements historiques à travers une interface interactive et enrichie. Les utilisateurs peuvent explorer des événements, consulter des archives multimédias associées (images, vidéos, PDF, audios), commenter, et sauvegarder leurs favoris.

2. Architecture du projet

- **Frontend** : Angular 19 (Standalone Components, Reactive Forms, Routing)
- **Backend** : Node.js + Express
- **Base de données** : MySQL
- **Middleware** : Multer (upload de fichiers)
- **Sécurité** : Authentification JWT
- **Upload fichiers** : géré dans `MediaService` côté frontend, `/uploads` dans Express

3. Fonctionnalités principales côté Utilisateur

Utilisateur /admin non connecté

- Visualisation de la ligne de temps

Utilisateur connecté

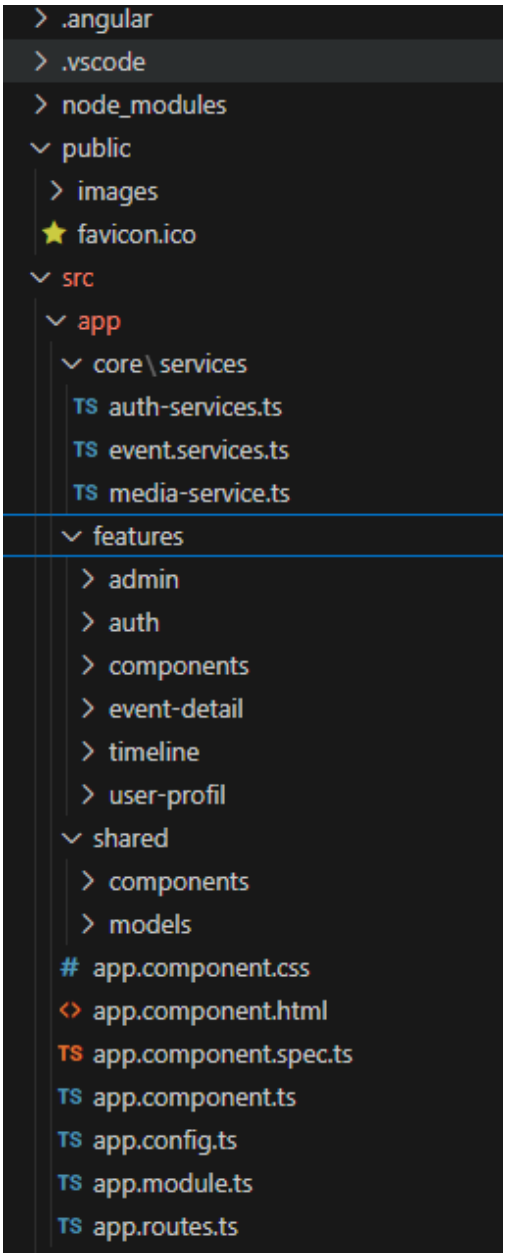
- Parcourir la ligne de temps
- Voir les details des evenements

- Accéder au archive des evenements
- Ajout de commentaires
- Ajout des favoris

4. Fonctionnalités principales côté Admin

- Gestion des évenements :
    - Ajout
    - Suppression
    - Modification
    - Affichage
- Ajout de commentaires
- valider/refuser des commentaire
- Ajout de favoris
- Parcourir la ligne de temps
- Faire des recherches
- Ajout des commentaires

5. Structure des composants Angular



- src/core: pour gerer l'appel de mes differents services

- src/features: pour les pages de l'application

- src/shared: pour mes composant reutilisable (header, footer)

6. Modèles (Models)

CollectionEvent (pour l’ajout et la recuperation des evenements)

```
export interface CollectionEvent {
  id_evenement: number;
  titre_evenement: string;
  description_evenement: string;
  id_civilisation: number;
  id_lieu: number;
  id_thematique: number;
  id_categorie: number;
  id_periode: number;
  date_debut: string;
  date_fin: string;
}
```

MediaPost(pour le post des evenements)

```
export interface mediaPost {
  nom_fichier: File;
  description_archive: string;
  type_archive: 'image' | 'archive';
  id_evenement: number;
  principal: string;
  url: string;
}
```

7. Upload de fichiers

- Géré via `FormData` dans `MediaService`
- Champs envoyés :
    - `fichier`: le fichier (Blob)
    - `id_evenement`

- `type_archive`
- `description_archive`
- `principal` (booléen)

8. Sécurité

- L'API vérifie le JWT dans les headers via un middleware `verifyToken()`
- Le champ `admin` est utilisé pour autoriser certaines actions

9. Installation locale

Backend

```
cd backend
npm install
npm start
```

Frontend

```
cd frontend
npm install
ng serve --open
```

10. Auteurs

- **TEMGOUA Carine**
- **KENFACK Ariol**
- **FONKUI William**