

Arthur Blairon

L3 CPES Informatique
2025/26

TP SuperHero Manager

Tomczak Robert

Table des matières

1 1. Présentation générale du projet.....	3
2 2. Architecture et technologies.....	3
♦ Backend — Node.js / Express / TypeScript.....	3
♦ Frontend — React / TypeScript.....	4
3 3. Fonctionnalités réalisées.....	5
✓ Authentification sécurisée.....	5
✓ Gestion complète des super-héros.....	5
✓ Filtrage avancé.....	5
✓ Interface utilisateur responsive.....	5
✓ Page de détails enrichie.....	6
4 4. Difficultés rencontrées.....	6
♦ Gestion des chemins d'images.....	6
♦ Migration ESMmodules / TypeScript.....	6
♦ Structure MongoDB incompatible.....	7
♦ Appels réseau du frontend.....	7
♦ Gestion des permissions JWT.....	7
5 5. Axes d'amélioration.....	7
♦ Améliorations techniques.....	7
♦ Améliorations UX/UI.....	7
♦ Extension fonctionnelle.....	8
6 6. Conclusion.....	8

1. Présentation générale du projet

Le projet *SuperHero Manager* est une application web complète permettant de consulter, rechercher, filtrer et administrer une base de données de super-héros.

Elle inclut :

- Un **backend Node.js + Express + MongoDB** pour la gestion des données.
- Un **frontend React + TypeScript** moderne et responsive.
- Un **système d'authentification JWT** permettant différents rôles (admin / editor / user).
- Une interface utilisateur permettant :
 - La **consultation détaillée** des informations d'un héros.
 - Le **filtrage avancé** par statistiques, éditeur et recherche textuelle.
 - La **gestion CRUD complète** (ajout, modification, suppression) réservée à l'administrateur.

Ce projet simule une application professionnelle de gestion de base de données avec API REST sécurisée et interface d'administration.

2. Architecture et technologies

L'application est constituée de deux modules indépendants communiquant via HTTP :

◆ Backend — Node.js / Express / TypeScript

- **Express.js** : création des routes API.
- **Mongoose** : modèle de données et interaction avec MongoDB.
- **JSON Web Tokens (JWT)** : authentification et gestion des permissions.
- **bcrypt.js** : hachage des mots de passe.
- **cors** : gestion de la sécurité des requêtes cross-origin.
- **ts-node-dev** : exécution du serveur en live reload.

Le backend expose des routes REST propres permettant :

Route	Méthode	Description
/api/auth/register	POST	Inscription utilisateur
/api/auth/login	POST	Connexion JWT
/api/auth/me	GET	Informations du compte
/api/heroes	GET	Liste filtrée + pagination

Route	Méthode	Description
/api/heroes/:id	GET	Détails d'un héros
/api/heroes	POST	Création d'un héros (admin)
/api/heroes/:id	PUT	Modification (admin)
/api/heroes/:id	DELETE	Suppression (admin)

◆ Frontend — React / TypeScript

- **React Router** pour la navigation.
- **Hooks** (useState, useEffect) pour la gestion du state.
- **Fetch API** pour les appels réseau.
- **CSS inline / responsive**.
- Gestion du token dans `localStorage`.

Le frontend propose :

- Une **page d'accueil** avec recherche, filtres et pagination.
- Une **page de détails du héros**, enrichie avec :
 - Images multi-résolution
 - Biographie détaillée
 - Apparence
 - Occupation
 - Relations
 - Statistiques visuelles
- Une interface **d'administration** pour modifier et supprimer un héros.
- Une page de **connexion** sécurisée.

3. Fonctionnalités réalisées

✓ Authentification sécurisée

- Token JWT généré à la connexion.
- Middlewares Express pour vérifier :
 - L'authentification
 - Le rôle de l'utilisateur (admin / editor)

- Protection des routes sensibles.

✓ Gestion complète des super-héros

- Import automatique de **563** héros depuis un JSON.
- Images disponibles en plusieurs tailles.
- CRUD complet :

Fonction	Lire la liste	Lire un héros	Ajouter un héros	Modifier un héros	Supprimer un héros
Accessible	Tous	Tous	Admin	Admin	Admin

✓ Filtrage avancé

- Recherche textuelle
- Filtre par éditeur (Marvel, DC...)
- Filtrage dynamique par statistiques (force, vitesse, intelligence...)
- Pagination personnalisée (12 / 24 / 48 héros par page)
- Saut direct à une page précise

✓ Interface utilisateur responsive

- Cartes adaptatives selon taille d'écran
- Images responsives
- Barre de statistiques colorée
- Animation hover

✓ Page de détails enrichie

- Photo HD
- Biographie complète
- Apparence physique
- Activité professionnelle
- Groupe d'affiliation
- Barres de statistiques colorées
- Boutons d'édition pour l'admin

4. Difficultés rencontrées

♦ Gestion des chemins d'images

Le JSON fournissait des URLs différentes (md, lg, sm...).

Le backend a dû :

- Servir un dossier `images/` statique.
- Corriger les chemins avec `http://localhost:4000/images/...`

♦ Migration ESM Modules / TypeScript

L'utilisation simultanée de :

- `"type": "module"`
- `ts-node`
- extensions `.ts`

a provoqué des erreurs `ERR_MODULE_NOT_FOUND` ou `Cannot use import.meta`.

La solution a été :

- Utiliser `ts-node-dev`
- Retirer les extensions `.js`
- Configurer Node en mode ESM dans `tsconfig.json`.

♦ Structure MongoDB incompatible

Le modèle initial ne stockait qu'un champ `image`.

Il a fallu ajouter :

`images: { xs, sm, md, lg }`



et réparer l'importation des données.

♦ Appels réseau du frontend

Certaines images n'apparaissaient pas car la route n'était pas :

`/images/md/hero.jpg`

Il a fallu reconstruire `imageUrl` dans le frontend.

	Technologie du Web	
	SuperHero Manager	

♦ Gestion des permissions JWT

Les routes admin étaient accessibles sans vérification.

Un middleware de rôle a été ajouté.

5. Axes d'amélioration

♦ Améliorations techniques

- Ajouter **une base de données Redis** pour accélérer la recherche.
- Mettre en place des **tests unitaires Jest** pour sécuriser le backend.
- Ajouter un système de **refresh token**.
- Implémenter **upload d'images** pour les héros créés manuellement.
- Ajouter une **documentation Swagger** automatique.

♦ Améliorations UX/UI

- Ajouter un **mode sombre**.
- Inclure un **graphique radar** pour les statistiques.
- Ajouter un **tri par ordre alphabétique ou par force**.
- Mettre en place une **barre latérale de filtres**.
- Ajouter l'historique des modifications pour les admins.

♦ Extension fonctionnelle

- Ajouter une **fonction de favoris** pour les utilisateurs.
- Ajouter des **combats simulés** basés sur les statistiques.
- Intégrer une API externe pour enrichir les données.

6. Conclusion

Ce projet a permis de construire une application web complète, moderne et réaliste intégrant :

- Un backend sécurisé avec gestion d'authentification,
- Un frontend ergonomique et responsive,
- Une base de données structurée et optimisée,
- Un workflow complet de gestion d'utilisateurs et de données.

Le résultat est une plateforme solide, extensible, et prête pour une démonstration professionnelle ou un hébergement en ligne.