



Dossier de projet

Développeur web et web mobile

Session d'examen 2025

Arthur Descourvieres

Sommaire

1. [À propos de moi](#)
2. [Introduction générale](#)
3. [Liste des compétences mises en œuvre](#)
4. [Expression des besoins](#)
5. [Environnement technique](#)
6. [Réalisations côté front-end](#)
7. [Réalisations côté back-end](#)
8. [Déploiement de l'application](#)
9. [Jeu d'essai et tests](#)
10. [Veille technologique](#)
11. [Conclusion](#)
12. [Annexes](#)

À propos de moi

Je m'appelle **Arthur**, j'ai 22 ans et je vis actuellement à Marseille.

Mon parcours a débuté dans le domaine du graphisme, avec trois années d'alternance de 2020 à 2023 pour préparer un bac professionnel en **Artisanat et Métiers d'Art, option communication visuelle**.

Cette expérience m'a permis de développer ma sensibilité à l'esthétique, à la structure visuelle et logique, mais aussi à la création numérique.

Curieux de nature, j'ai ensuite suivi une courte formation aux Petits Débrouillards, qui proposait une initiation à plusieurs domaines du numérique.

C'est à partir de là que j'ai découvert le développement web – un domaine qui a immédiatement suscité mon intérêt. C'est cette première découverte qui m'a motivé à intégrer la formation **Développeur Web et Web Mobile (DWWM)** proposée par [LaPlateforme](#), une école du numérique à Marseille.

Tout au long de cette formation, j'ai acquis des bases solides en développement *front-end* et *back-end*, et réalisé plusieurs projets pratiques.

Si je ne me considère pas encore comme passionné au sens strict, je peux dire que j'apprécie sincèrement ce métier et que je ressens une vraie motivation à apprendre toujours plus. Je prends plaisir à coder, à découvrir de nouvelles technos, et à construire des projets complets, de l'idée jusqu'au déploiement.

À l'avenir, je souhaite continuer à travailler sur des projets fullstack, variés, et tester différentes technologies modernes. Je m'intéresse particulièrement à des outils comme Next.js, Render, GSAP, ou encore PostgreSQL, et je suis enthousiaste à l'idée de continuer à explorer l'univers du **développement web**.

Introduction générale

Dans le cadre de la formation **Développeur Web et Web Mobile (DWWM)**, il nous a été demandé de concevoir un projet personnel permettant de mettre en œuvre l'ensemble des compétences acquises au cours du parcours. J'ai choisi de développer **CineTech**, une application web de gestion de films et séries. Ce projet permet aux utilisateurs de rechercher, consulter et organiser des œuvres audiovisuelles grâce à l'intégration de l'API TMDB (The Movie Database).

Le choix de ce thème n'est pas anodin : passionné par le cinéma et les technologies du web, j'ai souhaité allier ces deux domaines afin de créer une application à la fois utile, intuitive et agréable à utiliser. Le développement de Cinetech m'a permis de travailler sur un cas concret, en reproduisant des conditions similaires à celles rencontrées en entreprise : définition des besoins, choix techniques, gestion de données externes, responsive design, optimisation des performances, et mise en place d'un système d'authentification sécurisé.

Ce projet a été l'occasion de mobiliser les compétences techniques et méthodologiques acquises durant la formation, notamment en **PHP (Laravel 10)**, **MySQL**, **Blade**, **Tailwind CSS**, **Alpine.js** et **Axios**. J'ai également utilisé **Vite** pour la gestion des assets, ce qui m'a permis d'aborder des pratiques de développement modernes.

Ce dossier a pour objectif de présenter de manière claire les différentes étapes qui ont conduit à la réalisation du projet Cinetech. J'y détaille les choix techniques que j'ai effectués, les difficultés rencontrées au cours du développement, ainsi que les solutions que j'ai mises en œuvre pour aboutir à une application web fonctionnelle, organisée et évolutive.

Liste des compétences mises en œuvre

Dans le cadre du projet **Cinetech**, les compétences du référentiel DWWM suivantes ont été mobilisées :

1. Installer et configurer l'environnement de travail

- Mise en place de Laravel 10 (PHP 8.1+), MySQL, Composer, NPM et Vite pour le build front.
- Configuration des variables d'environnement (.env), des services (TMDB API).

2. Maquetter des interfaces utilisateur web ou web mobile

- Création de wireframes et prototypes (desktop & mobile) pour les pages de listing, détails et espace utilisateur.
- Définition de la charte graphique (couleurs, typographies) dans tailwind.config.js.

3. Réaliser des interfaces utilisateur statiques web ou web mobile

- Intégration HTML 5 avec Blade et Tailwind CSS 3.1 pour bâtir les gabarits statiques (layouts, partials).

4. Développer la partie dynamique des interfaces

- Ajout d'interactions légères avec Alpine.js 3.4 (menus, modals).
- Gestion des appels AJAX pour les avis via Axios 1.6.

5. Développer des composants métier côté serveur

- Conception d'une API REST sécurisée avec Laravel (routes, contrôleurs, policies).
- Utilisation de Sanctum pour l'authentification par token.

6. Mettre en place une base de données relationnelle

- Modélisation MySQL (schéma utilisateurs, films, avis).
- Gestion des migrations et seeders via Artisan.

7. Développer des composants d'accès aux données

- Requêtes Eloquent ORM pour MySQL.
- Appels externes via GuzzleHTTP à l'API TMDB.

8. Documenter le déploiement d'une application dynamique

- Rédaction d'un guide de déploiement : commandes Artisan, build Vite, configuration Nginx/Apache ou CI/CD (GitHub Actions → déploiement sur serveur/forge).

9. Mettre en œuvre une démarche de résolution de problème

- Debugging avec Spatie Laravel Web Tinker et logs Laravel.
- Tests fonctionnels manuels et itératifs pour valider chaque feature.

10. Apprendre en continu (veille technologique)

- Suivi des mises à jour Laravel 10, Tailwind CSS et Alpine.js via blogs officiels, GitHub et vidéos Youtube.

Expression des besoins

1. Présentation du besoin général

Avec la multitude de films et séries disponibles sur les différentes plateformes de streaming, il devient de plus en plus difficile pour les utilisateurs de suivre, retrouver et organiser leurs visionnages. Les solutions existantes sont souvent limitées ou dispersées, ce qui nuit à l'expérience utilisateur.

CineTech a été conçu pour répondre à ce besoin : il s'agit d'une application web centralisée permettant de rechercher, consulter et classer des œuvres audiovisuelles. Grâce à l'intégration de l'API TMDB, l'utilisateur accède à un large catalogue d'œuvres tout en bénéficiant d'une interface simple à utiliser.

2. Objectifs du projet

Le projet Cinetech poursuit plusieurs objectifs principaux :

- Permettre aux utilisateurs de rechercher des films et des séries à partir de l'API TMDB.
- Offrir la possibilité de consulter des informations détaillées sur chaque œuvre : synopsis, casting, bande-annonce, notation, etc.
- Proposer un espace personnel où l'utilisateur peut organiser ses contenus en listes personnalisées (films vus, à voir).
- Assurer une interface ergonomique, responsive et accessible sur tous types de supports (ordinateur, tablette, smartphone).

3. Fonctionnalités attendues

Afin d'atteindre ces objectifs, plusieurs fonctionnalités ont été prévues et développées dans l'application :

- **Recherche avancée** de films et séries avec suggestions dynamiques.
- **Consultation détaillée** des œuvres (fiche complète avec synopsis, casting, note, image).
- **Système d'authentification sécurisé** (inscription, connexion, déconnexion).
- **Interface responsive**, adaptée aux différents formats d'écran.
- **Navigation fluide**, grâce à un chargement dynamique des contenus via Axios.
- **Gestion des rôles utilisateur**, avec un **rôle administrateur** disposant de privilèges supplémentaires (ex. : modération, gestion des contenus).

User stories

En tant que...	Je veux...	Afin de...
Utilisateur anonyme	rechercher un film ou une série	obtenir des informations rapidement
Utilisateur authentifié	ajouter un commentaire sur un média	partager mon avis avec les autres
Utilisateur authentifié	voir la liste des films que j'ai marqués "à voir"	retrouver facilement ce que je veux voir
Admin	supprimer un commentaire inapproprié	modérer le contenu

4. Limites du projet

Le projet Cinetech, bien que fonctionnel et abouti sur de nombreux aspects, présente plusieurs limites dues au périmètre défini, aux contraintes de temps, et aux choix techniques :

- **Pas de système de notation personnalisée**

Les utilisateurs ne peuvent pas attribuer une note personnelle aux films ou séries. Seule la note issue de l'API TMDB est affichée. Cette fonctionnalité a été écartée pour privilégier l'implémentation stable des listes personnelles.

- **Absence d'interactions sociales entre utilisateurs**

Il n'est pas possible de suivre d'autres utilisateurs, de partager ses listes ou de commenter publiquement des œuvres. Ces aspects, bien qu'intéressants, sortaient du cadre initial et auraient impliqué une architecture plus complexe (modération, notifications, etc.).

- **Pas de système de recommandations personnalisées**

Il n'y a pas d'algorithme pour suggérer des œuvres en fonction des préférences ou de l'historique. L'API TMDB permet ce type de données, mais leur exploitation intelligente nécessite un traitement plus avancé (machine learning, scoring, etc.).

- **Pas d'internationalisation / traduction**

L'application est conçue uniquement en français, sans possibilité de basculer vers d'autres langues. Cela a été fait pour se concentrer sur la logique fonctionnelle principale.

5. Contraintes et limites

Le développement du projet Cinetech s'est inscrit dans un cadre précis, impliquant plusieurs contraintes :

- **Technologiques** : utilisation obligatoire de Laravel 10, MySQL, Blade, Tailwind CSS, Alpine.js, Axios.
- **Pédagogiques** : le projet a été réalisé dans un cadre individuel, avec un temps limité correspondant au temps restant en formation.
- **Fonctionnelles** : certaines fonctionnalités avancées (système de notation par utilisateur, interactions sociales) n'ont pas été implémentées, afin de respecter les délais et le périmètre initial défini.

6. Public cible

L'application Cinetech s'adresse principalement :

- Aux **cinéphiles** et amateurs de séries qui souhaitent organiser leur visionnage.
- Aux **utilisateurs en quête d'une alternative** légère et personnalisée aux plateformes commerciales souvent trop complexes ou surchargées.
- À un **public jeune et connecté**, habitué à la consommation numérique et aux interfaces modernes.

Environnement Technique

1. Outils de développement et configuration

- **IDE utilisé :** Visual Studio Code
- **Serveur local :** Laragon (Apache, MySQL, PHP 8.1+)
 - MySQL est exécuté via Laragon.
 - phpMyAdmin est utilisé pour visualiser et manipuler la base de données.
- **Lancement du projet :**
 - Serveur Laravel démarré avec php artisan serve
 - Compilation des assets front-end avec npm run dev (Vite)
- **Outils de versionning :** Git (en local) et GitHub (hébergement distant)
- **Gestion des dépendances :**
 - PHP : Composer
 - Front-end : npm
- **Outil de build :** Vite
- **Outil de test :** Vitest
- *(Pas d'outil de conteneurisation utilisé dans ce projet)*

2. Technologies back-end

- Langage : PHP 8.1+
- Framework : Laravel 10
- API REST / authentification : Sanctum
- Appels externes : GuzzleHTTP (TMDB)
- Gestion des rôles, sécurité (CSRF, validation, etc.)

3. Base de données

- MySQL
- Migrations, seeders
- Respect des règles de sécurité, confidentialité (RGPD)
- Sauvegardes, restauration

4. Technologies front-end

- Blade (Laravel)
- Tailwind CSS 3.1
- Alpine.js 3.4
- Axios 1.6 (appels API, interactions dynamiques)
- Embla Carousel (composant UX)
- Accessibilité (ARIA, responsive, RGAA)
- Tests unitaire avec Vitest

5. Services tiers

- API TMDB : récupération dynamique des données

6. Sécurité

- Authentification tokenisée (Sanctum)
- Protection CSRF
- Validation des entrées
- Gestion des rôles et accès

7. Responsive & accessibilité

- Interface responsive (desktop, tablette, mobile)
- Respect du RGAA

Réalisations côté front-end

Maquettes & enchaînement

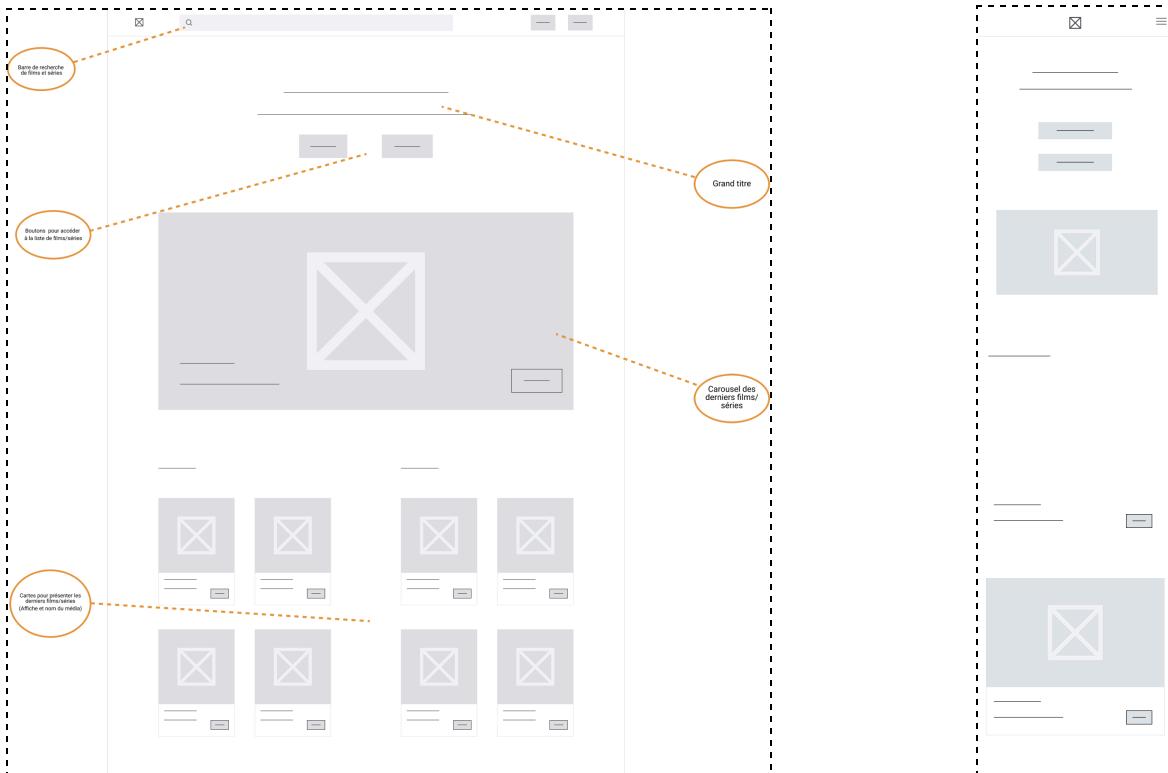
Avant de commencer le développement de l'interface utilisateur, j'ai réalisé des maquettes haute fidélité de l'application **CineTech**. Ces maquettes ont été conçues pour refléter une interface moderne, intuitive et responsive, adaptée aussi bien aux supports desktop qu'aux écrans mobiles.

Les maquettes ont été réalisées à l'aide de **Figma**, en respectant les principes d'accessibilité (RGAA), les bonnes pratiques de l'ergonomie et les recommandations en matière d'éco-conception.

Le parcours utilisateur principal a été réfléchi de manière à réduire les frictions et offrir une navigation fluide entre les différentes pages de l'application. **J'ai également pris en compte la compatibilité mobile (responsive) dès la conception des maquettes.**

L'enchaînement des interfaces a été modélisé sous forme d'arborescence, permettant de visualiser l'organisation des vues et la logique de navigation.

Page d'accueil (Desktop et mobile)



Page détail d'un média (Desktop et mobile)

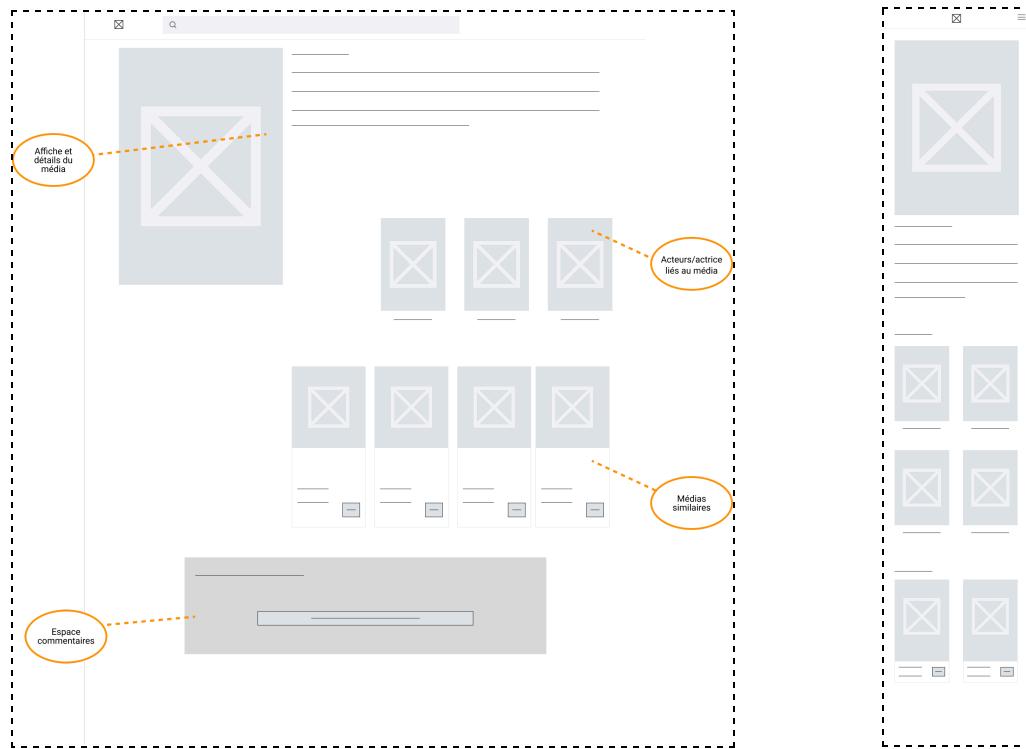
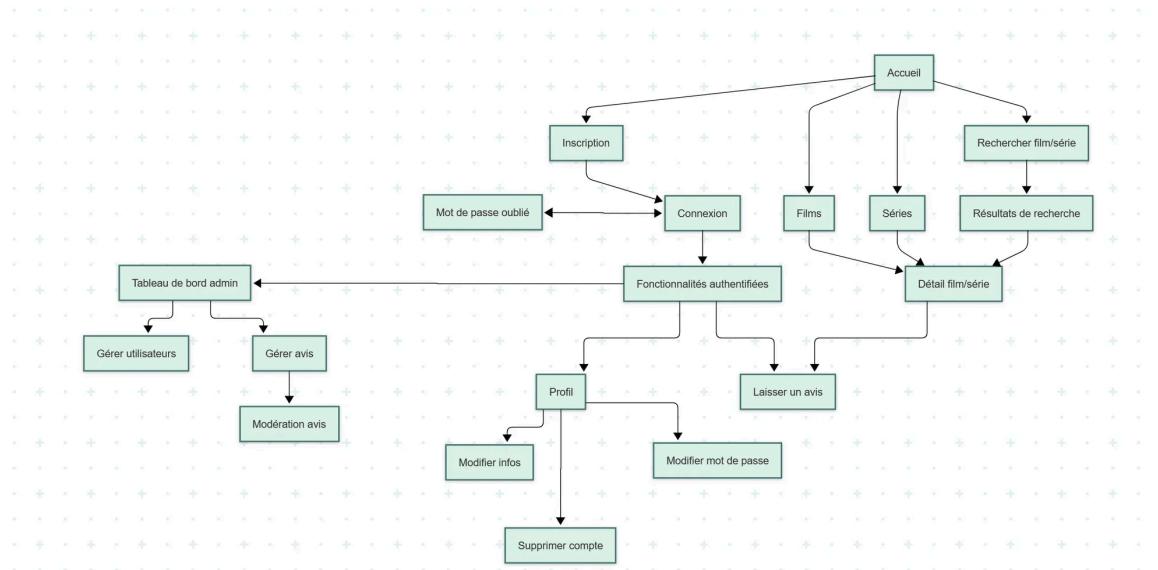


Schéma du parcours utilisateur



3.2 Interfaces statiques

Les interfaces statiques de l'application **CineTech** ont été développées à l'aide du moteur de templates **Blade** de Laravel et stylisées avec **Tailwind CSS**. Ce choix m'a permis de gagner en rapidité de développement tout en respectant une architecture modulaire et claire.

J'ai conçu plusieurs vues statiques réutilisables :

- La page d'accueil, et les pages de détails affichant des sélections de contenus populaires via l'API TMDB
- Les pages d'authentification (connexion, inscription)
- La structure des layouts (entête, navigation, pied de page) qui s'adapte selon l'état de connexion de l'utilisateur

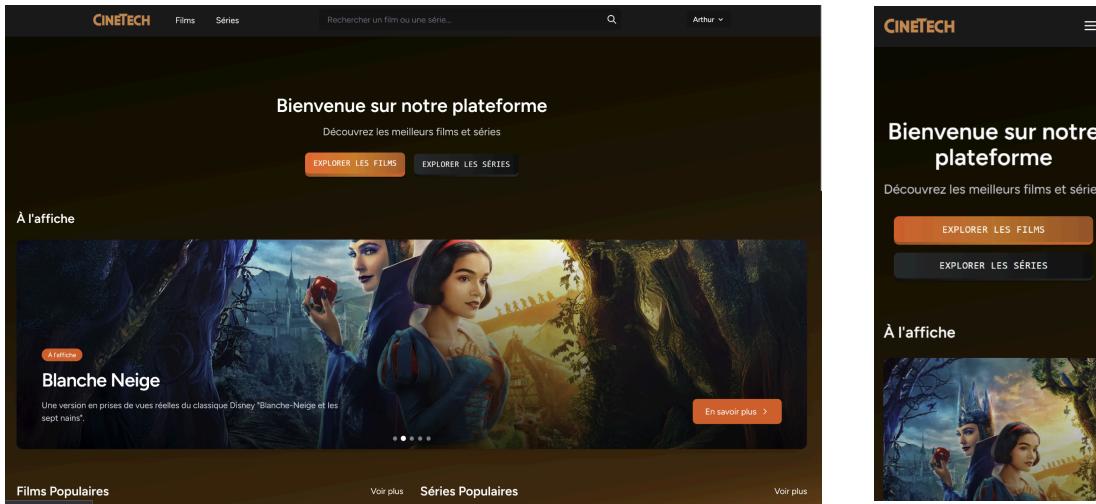
L'**organisation des vues respecte une structure claire**, avec un répertoire par fonctionnalité dans resources/views/ :

- movies/, tv-shows/ pour les fiches
- auth/ pour l'authentification
- profile/ pour le compte utilisateur
- components/ pour les composants réutilisables
- layouts/ pour les gabarits d'affichage

Ces interfaces ont été conçues en **mobile-first**, avec une attention particulière portée à :

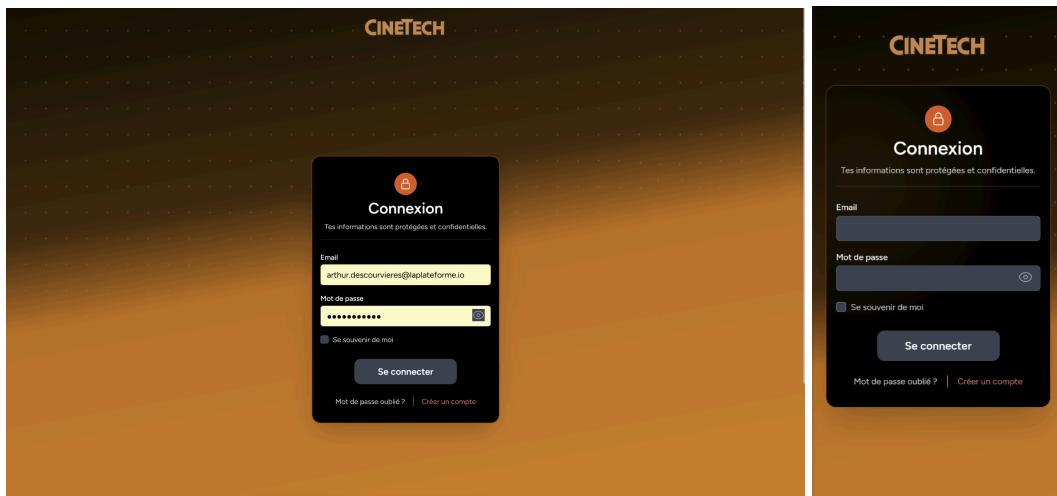
- La **responsivité** (flexbox, grid, breakpoints Tailwind)
- L'**accessibilité** : balises ARIA, contrastes conformes WCAG 2.1, **navigation au clavier testée**

Capture de la page d'accueil du site



The image displays two side-by-side screenshots of the CINE TECH website's homepage. Both versions feature a dark background with a large, stylized movie poster for "Blanche Neige" (Snow White) as the central focus. At the top, there is a navigation bar with the CINE TECH logo, a search icon, and a user profile icon for "Arthur". Below the banner, there are two prominent orange buttons labeled "EXPLORER LES FILMS" and "EXPLORER LES SÉRIES". The left screenshot is a desktop view, showing additional navigation links for "Films Populaires" and "Séries Populaires" along with "Voir plus" buttons. The right screenshot is a mobile view, showing a simplified layout with just the main banner and the exploration buttons.

Capture de la page connexion du site



The image displays two side-by-side screenshots of the CINE TECH website's login page. Both versions feature a dark background with a light-colored login form in the center. The form has a header with a padlock icon and the word "Connexion". It includes fields for "Email" (containing "arthur.descourvieres@plateforme.io") and "Mot de passe" (containing a redacted password). There is also a "Se souvenir de moi" checkbox and a "Se connecter" button. Below the form are links for "Mot de passe oublié ?" and "Créer un compte". The left screenshot is a desktop view, while the right one is a mobile view.

Extrait de code du fichier app.blade.php (layout principal)

```
<body class="font-sans antialiased">
  <div class="min-h-screen">
    @include('layouts.navigation')

    <!-- Page Heading -->
    @if (isset($header))
      <header>
        <div class="max-w-7xl mx-auto py-6 px-4 sm:px-6 lg:px-8">
          <h1 class="text-white">
            {{ $header }}
          </h1>
        </div>
      </header>
    @endif

    <!-- Page Content -->
    <main class="min-h-screen pb-24">
      @hasSection('content')
        @yield('content')
      @else
        {{ $slot ?? '' }}
      @endif
    </main>
  </div>
  <x-footer />
  @stack('scripts')
</body>
```

3.3 Interfaces dynamiques

Autocomplétion dans la barre de recherche

Objectif :

Offrir à l'utilisateur une expérience fluide en affichant des suggestions de films ou séries en temps réel lors de la saisie dans la barre de recherche.

Fonctionnement côté client

1. L'utilisateur commence à taper dans la barre de recherche.
2. Dès que la saisie atteint 2 caractères, une requête AJAX est automatiquement envoyée vers une route spécifique.
3. Le front-end reçoit une liste de 4 suggestions contenant le titre, le type, une image et un identifiant.
4. La liste est affichée dynamiquement sous le champ de recherche.
5. L'utilisateur peut parcourir les suggestions à l'aide du clavier ou de la souris.
6. La sélection d'une suggestion redirige vers la page correspondante du film ou de la série.

Fichiers concernés (front-end uniquement)

- **Blade + Alpine.js :**
 - resources/views/layouts/navigation.blade.php
 - Barre de recherche pour version desktop
 - Composant Alpine.js autocompleteSearch()
 - resources/views/components/mobile-menu.blade.php
 - Barre de recherche mobile avec autocompleteSearchMobile()
- **Styles :**
 - resources/css/app.css : Personnalisation de l'apparence des suggestions et de la scrollbar

Détail technique (front-end)

- Déclenchement de la méthode `fetchSuggestions()` via Alpine.js dès que l'utilisateur entre au moins 2 caractères.
- Envoi d'une requête GET vers une URL dédiée avec le paramètre `query`.
- Réception d'une réponse JSON contenant une liste filtrée de suggestions.
- Affichage immédiat des résultats sous le champ, avec navigation clavier/souris et surbrillance de la sélection.
- Un clic ou appui sur Entrée redirige l'utilisateur vers une fiche détaillée selon l'élément choisi.

Accessibilité et UX

- Prise en charge complète de la navigation clavier ($\uparrow \downarrow$ Entrée Échap)
- Mise en évidence visuelle de l'élément sélectionné
- Affichage d'un message en cas d'absence de résultat
- Indicateur de chargement lors de la requête

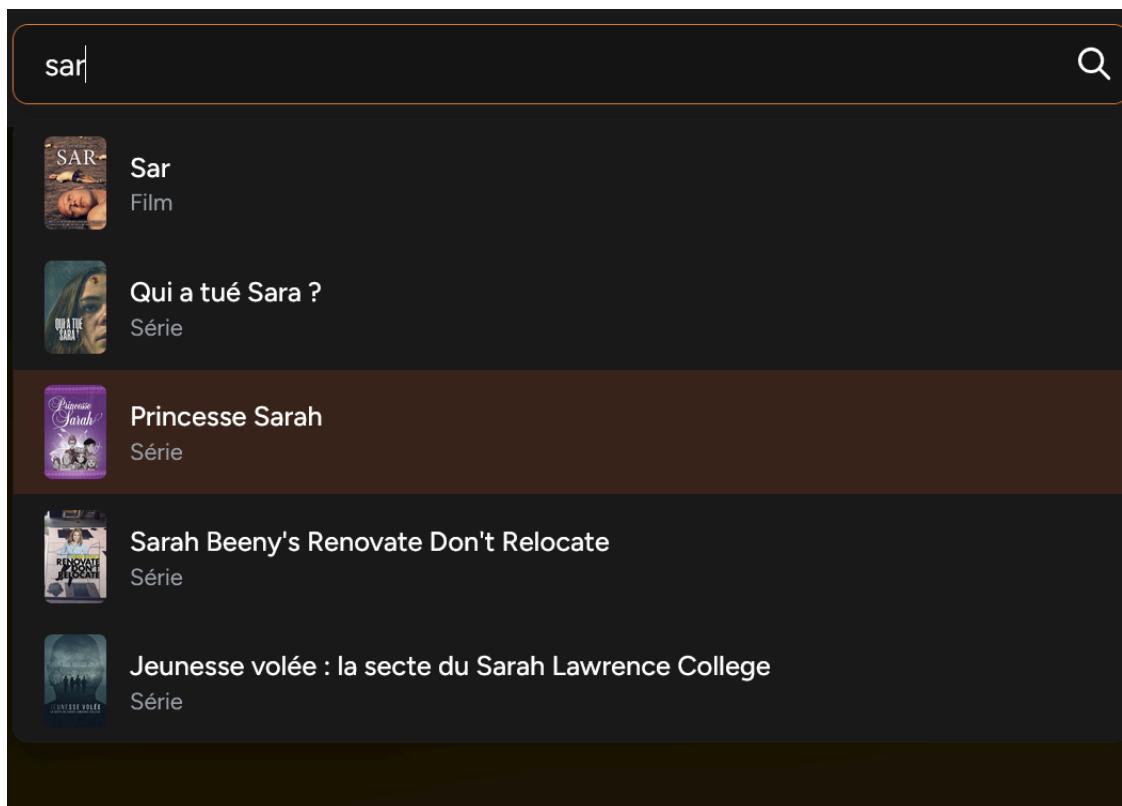
Fonction utilisé pour l'autocomplétion

```

1  function autocompleteSearch() {
2    return {
3      query: '',
4      suggestions: [],
5      open: false,
6      loading: false,
7      highlightedIndex: -1,
8      fetchSuggestions() {
9        if (this.query.length < 2) {
10          this.suggestions = [];
11          this.open = false;
12          return;
13        }
14        this.loading = true;
15        fetch(`api/autocomplete?query=${encodeURIComponent(this.query)}`)
16          .then(res => res.json())
17          .then(data => {
18            this.suggestions = data;
19            this.loading = false;
20            this.open = true;
21            this.highlightedIndex = -1;
22          })
23          .catch(() => {
24            this.suggestions = [];
25            this.loading = false;
26            this.open = true;
27          });
28        },
29        openList() {
30          if (this.suggestions.length > 0) this.open = true;
31        },
32        closeList() {
33          this.open = false;
34          this.highlightedIndex = -1;
35        },
36        closeListWithDelay() {
37          setTimeout(() => this.closeList(), 100);
38        },
39        highlightNext() {
40          if (!this.open || this.suggestions.length === 0) return;
41          this.highlightedIndex = (this.highlightedIndex + 1) % this.suggestions.length;
42        },
43        highlightPrev() {
44          if (!this.open || this.suggestions.length === 0) return;
45          this.highlightedIndex = (this.highlightedIndex - 1 + this.suggestions.length) % this.suggestions.length;
46        },
47        goToHighlighted() {
48          if (this.highlightedIndex >= 0 && this.suggestions[this.highlightedIndex]) {
49            this.goTo(this.highlightedIndex);
50          }
51        },
52        goTo(index) {
53          const item = this.suggestions[index];
54          if (item) {
55            window.location.href = item.media_type === 'movie' ? `/movies/${item.id}` : `/tv-shows/${item.id}`;
56          }
57        }
58      }
59    }
}

```

Affichage des résultats dynamique



Réalisations côté back-end

Dans cette partie, nous décrivons en détail la mise en œuvre de la couche serveur de l'application Cinetech. L'objectif est de présenter comment l'architecture back-end répond aux besoins fonctionnels (gestion des utilisateurs, commentaires, intégration de l'API TMDB, etc.) tout en garantissant performance, évolutivité et sécurité.

Nous commencerons par la modélisation de la base de données et son organisation via les migrations Laravel, puis nous détaillerons les composants d'accès aux données (repositories et modèles Eloquent) qui assurent la persistance et la récupération des informations. Ensuite, nous aborderons la logique métier encapsulée dans les services et contrôleurs, en illustrant le découpage en responsabilités claires. Enfin, nous présenterons les mécanismes de sécurité mis en place (authentification via Sanctum, gestion des rôles, protection CSRF, validation serveur) pour garantir l'intégrité des données et la robustesse de l'application.

4.1 Base de donnée



Objectif : donner une vue d'ensemble avant d'entrer dans le détail des tables.

4.1.1 Description des tables principales

Table users

Contient les informations relatives aux comptes utilisateurs et à leurs rôles.

- Chaque utilisateur possède un identifiant unique (id).
- Les champs nickname, email et password sont obligatoires (email est en plus unique).
- is_admin gère le rôle administrateur.
- remember_token, email_verified_at et timestamps servent à la sécurité et à la traçabilité.

Champ	Type	Contraintes	Commentaire
<code>id</code>	BIGINT	PK, AI	Identifiant unique de l'utilisateur
<code>nickname</code>	VARCHAR	NOT NULL	Pseudo de l'utilisateur
<code>email</code>	VARCHAR	UNIQUE, NOT NULL	Adresse email de l'utilisateur
<code>email_verified_at</code>	TIMESTAMP	NULLABLE	Date de vérification de l'email
<code>password</code>	VARCHAR	NOT NULL	Mot de passe hashé
<code>is_admin</code>	BOOLEAN	DEFAULT 0	1 si administrateur, 0 sinon
<code>remember_token</code>	VARCHAR(100)	NULLABLE	Token pour la reconnexion automatique
<code>created_at</code>	TIMESTAMP	NOT NULL	Date de création du compte
<code>updated_at</code>	TIMESTAMP	NOT NULL	Date de dernière modification du compte

Table comments

Stocke les avis laissés par les utilisateurs sur les médias TMDB.

- Référence l'utilisateur via user_id.
- tmdb_id et type précisent la cible du commentaire.

Champ	Type	Contraintes	Commentaire
<code>id</code>	BIGINT	PK, AI	Identifiant unique du commentaire
<code>user_id</code>	BIGINT	FK → users.id, NOT NULL	Référence à l'utilisateur auteur du commentaire
<code>tmdb_id</code>	BIGINT	NOT NULL	Identifiant du film ou de la série dans TMDB
<code>type</code>	VARCHAR	NOT NULL	Type de média : 'movie' ou 'tv'
<code>content</code>	TEXT	NOT NULL	Contenu du commentaire
<code>created_at</code>	TIMESTAMP	NOT NULL	Date de création du commentaire
<code>updated_at</code>	TIMESTAMP	NOT NULL	Date de dernière modification du commentaire

Table password_reset_tokens

Champ	Type	Contraintes	Commentaire
<code>email</code>	VARCHAR	PK, NOT NULL	Adresse email de l'utilisateur
<code>token</code>	VARCHAR	NOT NULL	Jeton de réinitialisation du mot de passe
<code>created_at</code>	TIMESTAMP	NOT NULL	Date de création du jeton

Table personal_access_tokens

Champ	Type	Contraintes	Commentaire
<code>id</code>	BIGINT	PK, AI	Identifiant unique du token
<code>tokenable_type</code>	VARCHAR	NOT NULL	Type de l'entité associée (ex : App\Models\User)
<code>tokenable_id</code>	BIGINT	NOT NULL	ID de l'entité associée
<code>name</code>	VARCHAR	NOT NULL	Nom du token
<code>token</code>	VARCHAR	UNIQUE, NOT NULL	Valeur du token (hashée)
<code>abilities</code>	TEXT	NULLABLE	Permissions associées au token
<code>last_used_at</code>	TIMESTAMP	NULLABLE	Dernière utilisation du token
<code>expires_at</code>	TIMESTAMP	NULLABLE	Date d'expiration du token
<code>created_at</code>	TIMESTAMP	NOT NULL	Date de création du token
<code>updated_at</code>	TIMESTAMP	NOT NULL	Date de dernière modification du token

Table password_reset_tokens

Champ	Type	Contraintes	Commentaire
<code>email</code>	VARCHAR	PK, NOT NULL	Adresse email de l'utilisateur
<code>token</code>	VARCHAR	NOT NULL	Jeton de réinitialisation du mot de passe
<code>created_at</code>	TIMESTAMP	NOT NULL	Date de création du jeton

4.1.2 Contraintes d'intégrité et index

- **Clés étrangères :**
 - comments.user_id → users.id (ON DELETE CASCADE)
- **Index :**
 - Index UNIQUE sur users.email
 - Index sur comments.tmdb_id, comments.type pour accélérer les requêtes de récupération
- **Justification :** améliore les performances et garantit l'intégrité référentielle.

4.1.3 Conclusion

- La base est normalisée (1NF, 2NF, 3NF), évitant redondance et anomalies.
 - Séparation claire entre tables métier et tables techniques.
 - Schéma facilement extensible (nouveaux médias, rôles, fonctionnalités).
 - Sauvegarde et restauration rapides grâce au script SQL.
-

4.2 Composants d'accès aux données

L'application Cinetech utilise **Laravel 10** et son ORM **Eloquent** pour gérer les interactions entre le code et la base de données. Ce système permet un accès sécurisé, structuré et maintenable aux données.

Modèles Eloquent

Les modèles représentent les tables principales :

- User : gère les utilisateurs, leur authentification et leurs rôles.
- Comment : permet à un utilisateur de laisser un avis sur un média (film ou série).

Les relations sont définies avec hasMany, belongsTo, etc., assurant une cohérence logique des entités.

Contrôleurs et logique d'accès

Les **contrôleurs** orchestrent les opérations entre l'utilisateur, les modèles et la base :

- Validation des données entrantes (\$request->validate()).
- Application des règles de sécurité (auth()->id() pour l'auteur du commentaire).
- Renvoi d'une réponse JSON ou d'une vue avec pagination.

Appels à l'API TMDB

L'API TMDB est interrogée via `Http::get()` pour récupérer dynamiquement les informations des médias (titre, image, synopsis), évitant ainsi de les stocker en base.

Sécurité et autorisations

La sécurité est assurée à plusieurs niveaux :

- **Middleware Laravel** pour l'authentification.
- **Policies** pour restreindre certaines actions (ex. : seuls l'auteur ou un admin peuvent supprimer un commentaire).
- **Validation serveur** systématique.

Administration

Une interface dédiée permet aux administrateurs de consulter et modérer les commentaires, accessible uniquement via un middleware isAdmin.

Voici quelques fichiers concernés par le chapitre :

Extrait de Models/User :

```
1 <?php
2
3 namespace App\Models;
4
5 // use Illuminate\Contracts\Auth\MustVerifyEmail;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8 use Illuminate\Notifications\Notifiable;
9 use Laravel\Sanctum\HasApiTokens;
10
11 class User extends Authenticatable
12 {
13     use HasApiTokens, HasFactory, Notifiable;
14
15
16     public function isAdmin()
17     {
18         return $this->is_admin;
19     }
20
21     public function sendPasswordResetNotification($token)
22     {
23         $this->notify(new \App\Notifications\ResetPassword($token));
24     }
25 }
```

Méthode index de CommentController :

```

15  /**
16   * Afficher les commentaires pour un film/série
17  */
18  public function index(Request $request)
19  {
20      $perPage = 3;
21      $comments = Comment::forMedia(
22          $request->tmdb_id,
23          $request->type
24      )->latest()
25      ->paginate($perPage);
26
27      return response()->json([
28          'comments' => $comments->items(),
29          'current_page' => $comments->currentPage(),
30          'last_page' => $comments->lastPage(),
31          'total' => $comments->total(),
32      ]);
33  }
34

```

- **Validation** : serveur via `$request->validate()`.
- **Sécurité** : `auth()->id()` garantit l'auteur.
- **Pagination** : standard Laravel (`->paginate(...)`).

Page d'administration des commentaires :

Gestion des commentaires							
ID	Utilisateur	Type	Média	Contenu	Date	Action	
41	Arthur	Série	Gute Zeiten, schlechte Zeiten	dqsqdqsdqs	03/05/2025 12:41	<button>Supprimer</button>	
40	Arthur	Série	Gute Zeiten, schlechte Zeiten	dqsdqsdqs	03/05/2025 12:41	<button>Supprimer</button>	
39	Arthur	Série	Gute Zeiten, schlechte Zeiten	qdqdqsdqs	03/05/2025 12:41	<button>Supprimer</button>	
38	Arthur	Film	Flight Maayam	dqsdqdd	03/05/2025 12:41	<button>Supprimer</button>	
37	Arthur	Film	Flight Maayam	gsfsdfdsfd	03/05/2025 12:40	<button>Supprimer</button>	
36	Arthur	Film	Flight Maayam	dqsdqsdq	03/05/2025 12:40	<button>Supprimer</button>	
35	Arthur	Film	Flight Maayam	stfgd d	03/05/2025 12:40	<button>Supprimer</button>	
31	Arthur	Série	Gute Zeiten, schlechte Zeiten	ddsqdqsdq	03/05/2025 12:32	<button>Supprimer</button>	
23	Arthur	Film	Exteriorial	dqsdqsdqs	03/05/2025 11:39	<button>Supprimer</button>	
22	Arthur	Film	Exteriorial	sqdqsdqsdqs	03/05/2025 11:34	<button>Supprimer</button>	
21	Arthur	Film	Minecraft, Le Film	dsdgds	28/04/2025 19:51	<button>Supprimer</button>	
3	Arthur	Film	Flying Saucer Rock 'N' Roll	Iste sint saepe delectus voluptas minima recusandae iu...	28/04/2025 19:51	<button>Supprimer</button>	
5	Arthur	Série	Head Case	Eos quisquam in qui id ea quo nam dolore minima.	28/04/2025 19:51	<button>Supprimer</button>	
7	Arthur	Série	Find My Family	Omnis aut et quia nihil ea dolores consequatur est fugi...	28/04/2025 19:51	<button>Supprimer</button>	
8	Arthur	Film	Crossed	Quia est voluptatem sed omnis enim accusamus quis.	28/04/2025 19:51	<button>Supprimer</button>	

Page 1 sur 2 Suivant >

4.3 Composants métiers

Les composants métiers de l'application **CineTech** encapsulent la logique de traitement propre aux règles fonctionnelles de l'application, notamment la gestion des commentaires utilisateurs et les contrôles d'accès.

1. Services applicatifs

La logique métier principale repose sur des méthodes spécifiques aux **contrôleurs**, complétée par des **Policies Laravel** et des **middlewares** :

- **Gestion des commentaires** : seuls les utilisateurs connectés peuvent publier ou supprimer un commentaire, à condition d'en être l'auteur ou d'avoir un rôle d'administrateur.
- **Mise en œuvre des règles d'accès** : via isAdmin et auth() pour filtrer les autorisations.
- **Validation des données** : avant chaque traitement, les données utilisateur sont validées côté serveur (\$request->validate()).

2. Découpage métier et lisibilité

Chaque responsabilité métier est isolée dans son contrôleur respectif (ex. : CommentController, ProfileController), suivant une architecture MVC propre à Laravel. Cela favorise la lisibilité du code et la maintenance future.

3. Respect des bonnes pratiques

- Utilisation de la **programmation orientée objet** (POO).
- **Tests manuels et jeux d'essai** pour valider les cas d'usage.
- **Sécurisation** des traitements : vérification des droits avant suppression ou modification.

Extrait montrant une logique conditionnelle métier

```

48  /**
49   * Check if the user is an admin.
50   */
51  public function isAdmin()
52  {
53      return $this->is_admin;
54 }

```

Exemples de routes api utilisées

Méthode	Endpoint	Description	Authentification
GET	/api/user	Retourne les infos de l'utilisateur connecté	Requise (Sanctum)
GET	/api/comments	Liste paginée des commentaires	Non requise
POST	/api/comments	Création d'un commentaire	Requise (utilisateur)
DELETE	/api/comments/{id}	Suppression d'un commentaire	Requise (auteur/admin)

4.4 Sécurité de l'application

La sécurité de l'application **CineTech** a été pensée à plusieurs niveaux pour garantir l'intégrité des données, la protection des utilisateurs, et la prévention des failles courantes côté serveur comme côté client.

1. Authentification et gestion des rôles

- Authentification basée sur **Sanctum** avec jetons sécurisés.
- Gestion des rôles via le champ `is_admin` dans la table `users`, accessible via la méthode métier `isAdmin()`.
- Middleware `IsAdmin` appliqué aux routes réservées à l'administration (ex. : modération).

2. Validation et filtrage des données

- Toutes les entrées utilisateur sont **validées côté serveur** avec `$request->validate()`.
- Les champs sensibles (ex. : mot de passe) sont hashés avec **bcrypt**.
- Les politiques (Policies) limitent les actions autorisées selon le rôle et l'identité de l'utilisateur.

3. Protection contre les attaques courantes

- **CSRF** : les formulaires HTML utilisent les protections automatiques de Laravel (`@csrf`).
- **XSS** : les données affichées dans les vues Blade sont automatiquement échappées.
- **Injection SQL** : évitée grâce à l'utilisation de l'ORM Eloquent et des requêtes préparées.
- **Rate limiting** possible à activer via les middlewares Laravel pour certaines routes sensibles.

4. Sécurité API et confidentialité

- L'API TMDB est appelée côté serveur via **Guzzle**, évitant toute exposition de la clé publique dans le front.
- Les identifiants et tokens sont stockés dans le fichier `.env`, non versionné dans GitHub.
- Les pages privées sont protégées par middleware auth.

5. Exemples d'implémentation

- Middleware : app/Http/Middleware/IsAdmin.php
- Policy : CommentPolicy.php pour autoriser ou refuser la suppression d'un commentaire
- Route sécurisée.

Extrait de /Middleware/IsAdmin.php

```
public function handle(Request $request, Closure $next): Response
{
    if (!auth()->check() || !auth()->user()->is_admin) {
        abort(403);
    }
    return $next($request);
}
```

Extrait de /admin/comments.blade.php

```
<form id="deleteForm" method="POST" class="inline">
    @csrf
    @method('DELETE')
    <button type="submit" class="px-4 py-2 bg-red-600 text-white rounded hover:bg-red-700">
        Confirmer
    </button>
</form>
```

4.5 Ouverture NoSQL : projet complémentaire *NetDart*

En complément du projet principal **CineTech** basé sur une base de données relationnelle (MySQL), j'ai réalisé un mini-projet intitulé *NetDart* afin d'explorer les bases de données NoSQL, et plus particulièrement MongoDB.

Ce projet est un **livre d'or numérique** développé avec **Next.js** (frontend en React), **TypeScript** et **MongoDB** pour la gestion des données. Il permet aux utilisateurs de laisser des messages, qui sont stockés et affichés dynamiquement via une base NoSQL. L'interface est responsive, moderne, et enrichie par des animations fluides grâce à GSAP.

Objectifs pédagogiques

- Découvrir le fonctionnement des bases NoSQL (documents JSON, absence de schéma strict).
- Expérimenter une base de données orientée documents (MongoDB) avec Mongoose.
- Mettre en place un système simple de stockage et récupération de messages sans structure relationnelle.

Caractéristiques principales

- **Stockage NoSQL** des messages avec les champs : nom, email, message, date.
- **Validation** des données côté client et serveur.
- **Affichage dynamique** des messages avec actualisation en temps réel.
- Utilisation de **MongoDB Compass** pour la visualisation et la gestion de la base.

Intérêt pour ma formation

Ce projet m'a permis d'appréhender un autre paradigme de gestion de données que celui utilisé dans **CineTech**. Il complète mes compétences en me familiarisant avec des cas d'usage plus souples et documentaires, adaptés à certains types d'applications comme les microservices ou les contenus non structurés.

Exemple de donnée dans la collection messages

The screenshot shows the MongoDB Compass interface for a collection named 'messages'. At the top, there are tabs for 'Documents' (1), 'Aggregations', 'Schema', 'Indexes' (1), and 'Validation'. Below the tabs is a search bar with placeholder text 'Type a query: { field: 'value' } or [Generate query](#)'. To the right of the search bar are buttons for 'Explain', 'Reset', 'Find' (which is highlighted in green), and 'Options'. Below the search bar are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. At the bottom, there is a preview area showing a single document:

```

_id: ObjectId('6824c2b9527dc791da3857ca')
name: "Halla Berger"
email: "calamailinator.com"
message: "Consequat Excepturi quibusdam ad vero repellendus Est unde itaque"
createdAt: 2025-05-14T16:20:09.370+00:00
__v: 0
  
```

Below the preview area, there are navigation controls for pages 1-1 of 1, and icons for search, sort, and refresh.

Déploiement de l'application

5. Déploiement de l'application

5.1 Objectifs du déploiement

Le but du déploiement de Cinetech est de rendre l'application **accessible publiquement** via un hébergeur web, en assurant un bon fonctionnement, une configuration adaptée à Laravel, et un **niveau minimal de sécurité**. Le déploiement a été réalisé **chez Hostinger**, un hébergeur mutualisé compatible avec PHP 8.1+ et MySQL.

5.2 Environnement de production

- **Hébergeur** : [Hostinger](#)
 - **Type** : Hébergement mutualisé PHP/MySQL
 - **Nom de domaine** : (ex. cinetech.derroce.com)
 - **Langage serveur** : PHP 8.1
 - **SGBD** : MySQL 8
 - **Gestionnaire de fichiers** : File Manager Hostinger / FTP (FileZilla)
 - **Accès base de données** : phpMyAdmin
-

5.3 Étapes du déploiement

1. Préparation de l'application en local

- Suppression des fichiers inutiles pour la prod (node_modules, fichiers de dev)
- Compilation des assets avec Vite :
“npm run build”

2. Export du projet vers le serveur

- Upload du projet via FTP dans /public_html/
- Déplacement du contenu de public/ vers public_html/ (spécificité des hébergeurs mutualisés)

Mise à jour des chemins dans index.php avec :

```
require __DIR__.'/../cinetech/vendor/autoload.php';  
  
$app = require_once __DIR__.'/../cinetech/bootstrap/app.php';
```

3. Installation des dépendances PHP

- Connexion au terminal Web (via SSH)

Exécution de :

- “composer install --no-dev”
- “php artisan config:cache”
- “php artisan route:cache”

4. Configuration de l'environnement

- Création du fichier .env.prod avec :
 - Accès DB (host, user, password)
 - TMDB API KEY
 - URL de l'application (APP_URL)

Génération de la clé :

“php artisan key:generate”

5. Migration de la base de données

- Création de la base MySQL via le tableau de bord Hostinger
- Import du schéma via phpMyAdmin

Lancement des migrations artisan :

- php artisan migrate --seed

6. Sécurisation

- Activation du HTTPS via Let's Encrypt (offert par Hostinger)

Protection du fichier .env via .htaccess :

<Files .env>

Order allow,deny

Deny from all

</Files>

5.4 Limites et pistes d'amélioration

- Le déploiement est **manuel**, sans automatisation continue (CI/CD)
- Pas de Dockerisation ni environnement de staging
- Aucune supervision automatique des erreurs (pas d'intégration Sentry, etc.)

Jeu d'essai et tests

Objectif

Cette section présente un jeu d'essai fonctionnel et des tests automatisés centrés sur la fonctionnalité principale du projet **CineTech** : la gestion des commentaires sur les films et séries.

L'objectif est de démontrer que cette fonctionnalité a été testée de manière rigoureuse, à travers des tests manuels et automatisés, couvrant les cas d'usage principaux, les erreurs, et le bon fonctionnement de l'interface.

Fonctionnalité : Commentaires

La gestion des commentaires implique plusieurs aspects techniques :

- Authentification utilisateur via **Laravel Sanctum**
- Requêtes **AJAX** avec **Axios**
- Contrôleurs Laravel sécurisés par **middleware**
- Sauvegarde et suppression en base de données (table comments)
- Feedback dynamique et réactif via Alpine.js
- Tests automatisés avec :
 - PHPUnit côté back-end
 - Vitest côté front-end

Jeu d'essai fonctionnel + tests automatisés

Cas de test	Données en entrée	Résultat attendu	Testé via	Résultat obtenu
Ajout d'un commentaire valide	Texte = "Excellent film", ID TMDB = 550	Commentaire enregistré en base, affiché dans l'interface	PHPUnit + Vitest	OK
Envoi d'un commentaire vide	Texte = "", ID TMDB = 550	Message d'erreur de validation côté serveur	PHPUnit	OK
Suppression d'un commentaire par son auteur	ID du commentaire = 42	Suppression en base + disparition immédiate dans le DOM	Vitest	OK
Suppression d'un commentaire depuis le backoffice	Rôle admin connecté, ID du commentaire = 43	Suppression autorisée + retour au tableau de modération	PHPUnit + Vitest	OK
Suppression par un utilisateur non autorisé	ID commentaire = 42, utilisateur ≠ auteur	Refus (HTTP 403), commentaire non supprimé	PHPUnit	OK
Affichage des commentaires	Film avec plusieurs commentaires	Liste correctement affichée au chargement	Vitest	OK
Gestion de la pagination	>10 commentaires sur un même film	Affichage par pages avec navigation (suivant/précédent)	Vitest	OK

Résumé des résultats

Tous les tests effectués ont donné les résultats attendus, et aucune anomalie fonctionnelle n'a été observée.

Limites des tests

- Les tests de sécurité (XSS, injections) n'ont pas été simulés en profondeur.
 - Pas de test de performance sous charge élevée.
 - Pas encore de tests end-to-end automatisés simulant le parcours complet utilisateur.
-

Perspectives d'amélioration

- Ajouter des tests e2e avec Cypress ou Laravel Dusk pour automatiser les parcours utilisateur.
- Effectuer des tests d'accessibilité avec des outils comme **WAVE** ou **Axe**.
- Intégrer des tests de robustesse pour des cas limites (ex. déconnexion inattendue, suppression simultanée).

Exécution du fichier CommentControllerTest.php (Backend)

```
● $ php artisan test tests/Feature/CommentControllerTest.php

  PASS  Tests\Feature\CommentControllerTest
    ✓ ajout commentaire valide reussi          4.13s
    ✓ ajout commentaire vide echoue          0.17s

  Tests:    2 passed (6 assertions)
  Duration: 4.47s
```

Exécution du fichier comments.test.js (Frontend)

```
$ npm run test comments.test.js

> test
> vitest comments.test.js

  DEV  v3.1.3 C:/Docker/dev-environment/Projects/laravel-projects/cinetech

  ✓ tests/Unit/comments.test.js (4 tests) 46ms
    ✓ Composant Comments > charge et affiche les commentaires correctement 19ms
    ✓ Composant Comments > gère correctement la pagination 10ms
    ✓ Composant Comments > permet d'ajouter un nouveau commentaire 8ms
    ✓ Composant Comments > permet de supprimer un commentaire 7ms

  Test Files 1 passed (1)
    Tests 4 passed (4)
    Start at 09:45:57
    Duration 3.60s (transform 68ms, setup 0ms, collect 935ms, tests 46ms, environment 706ms, prepare 331ms)

  PASS  Waiting for file changes...
        press h to show help, press q to quit
```

Veille technologique

La veille technologique consiste à s'informer régulièrement des évolutions des technologies, outils, frameworks et bonnes pratiques dans le domaine du développement web. Elle permet de rester compétent et pertinent dans un secteur en constante évolution.

1. Sujets suivis pendant la formation et mon projet

- **Laravel 10** : nouveautés de la version (notamment l'évolution des systèmes d'authentification comme Sanctum).
- **Tailwind CSS** : montée de version, nouvelles fonctionnalités, gestion des breakpoints, dark mode, etc.
- **Alpine.js** : syntaxe simplifiée, comparaison avec d'autres micro-frameworks comme Stimulus.
- **Axios** : bonnes pratiques pour les appels API, gestion des erreurs.
- **API TMDB** : documentation, limites d'utilisation, mise à jour de l'API.

Sécurité et vulnérabilités courantes

J'ai suivi des contenus sur les failles de sécurité web les plus fréquentes, comme les attaques XSS, CSRF ou l'injection SQL. Cela m'a permis de mieux comprendre comment les éviter dans un projet Laravel. J'ai notamment renforcé la validation des données entrantes, activé les protections CSRF dans les formulaires Blade, et utilisé l'ORM Eloquent pour prévenir les injections SQL. Des ressources comme la cheatsheet OWASP ou les articles de Laravel Daily m'ont été utiles pour intégrer ces pratiques dans Cinetech.

2. Outils de veille utilisés

- Blogs : [Laravel News](#), [Tailwind Blog](#)
 - YouTube / Conférences : Laracon, vidéos de tutoriels.
 - Newsletters : Laravel Daily, Daily.dev, Medium
-

3. Impact sur le projet

Par exemple, j'ai découvert grâce à Laravel News l'intérêt de Sanctum pour l'authentification SPA/API, ce qui m'a permis de sécuriser les commentaires dans Cinetech de manière moderne et efficace.

De même, j'ai suivi l'évolution de Tailwind CSS pour optimiser la structure responsive de l'interface utilisateur et mieux gérer les breakpoints.

4. Perspectives

Je souhaite continuer cette démarche de veille au quotidien, notamment autour de technologies comme Next.js, PostgreSQL, ou encore les outils de déploiement comme Vercel et Render, que je compte expérimenter après la formation.

Conclusion

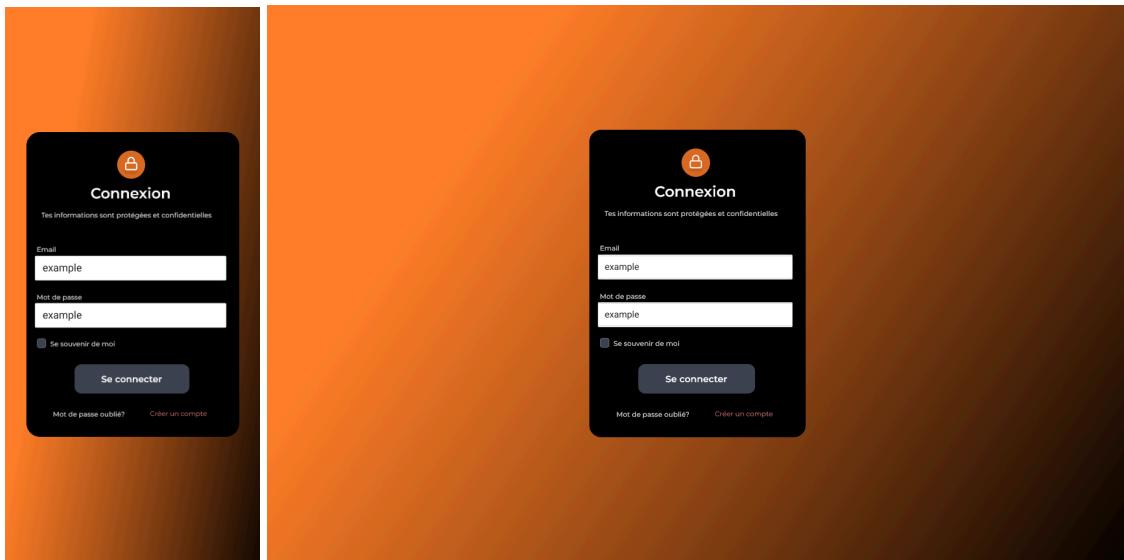
La réalisation de **CineTech** a représenté pour moi bien plus qu'un simple exercice pédagogique. Ce projet m'a permis de mettre en œuvre l'ensemble des compétences acquises au cours de la formation DWWM, tout en me confrontant à une logique de développement réaliste : expression des besoins, structuration du code, choix techniques pertinents, gestion des priorités, et résolution autonome de problèmes.

En développant cette application, j'ai pu approfondir l'utilisation de technologies modernes telles que Laravel, Tailwind CSS, Alpine.js ou encore Axios, tout en m'initiant à des pratiques professionnelles : appels API externes, sécurisation des données, responsive design, et déploiement en ligne. J'ai également appris à mieux structurer un projet complet, à anticiper les contraintes, et à produire une documentation claire.

Même si certaines fonctionnalités ont été mises de côté pour des raisons de périmètre ou de temps, je suis satisfait du résultat obtenu. Cinetech est une application fonctionnelle, cohérente, et évolutive, qui pourrait facilement être enrichie à l'avenir.

Ce projet m'a confirmé mon intérêt pour le développement web fullstack et ma volonté de continuer à progresser dans ce domaine. Je ressors de cette expérience avec une meilleure compréhension des enjeux du métier, et une motivation renforcée pour aborder de nouveaux défis techniques, que ce soit en entreprise ou dans des projets personnels.

Annexes



Movie Details:

- Title:** Warfare
- Year:** 2006
- Genre:** Action, Drama
- Rating:** 7/10
- Director:** Michael Winterbottom
- Actors:** Sharlto Copley, Will Poulter, Cillian Murphy, Kit Connor, Fionn Whitehead
- Plot Summary:** 2006 pendant la Guerre d'Irak. Un peloton de Navy SEALs américains est en planque dans la maison d'une famille irakienne pour une mission dangereuse à Ramadi. Avec des snipers, les militaires surveillent tous les mouvements de « l'ennemi » dans les moindres détails mais ils ne peuvent empêcher une embuscade. Une pluie de balles va causer de graves blessures et la mission se retrouve complètement bouleversée. Tandis que les hommes mettent tout en œuvre pour sauver les blessés en attendant désespérément d'être secourus, leur situation est de plus en plus désespérée. Combien de temps pourront-ils encore tenir?

Similar Movies:

- Platoon
- Bullitt
- Full Metal Jacket
- À armes égales

Comments:

Votre commentaire... (max 1000 caractères)

Nombre de commentaires: 0

Publier

