Documentation du projet - MyWatchingCompanion

# Présentation du projet

# MyWatchingCompanion est une application web conçue dans le cadre de l’épreuve E5 du BTS Services Informatiques aux Organisations (option SLAM). Le but principal de ce projet est de proposer un outil simple, intuitif et personnalisable permettant à chaque utilisateur de gérer ses séries et films en fonction de son avancement personnel.

# Chaque utilisateur peut ainsi créer une ou plusieurs watchlists (listes de visionnage) personnalisées selon ses goûts, ses envies ou des thématiques spécifiques (par exemple : “À regarder entre amis”, “Films primés”, “Séries Netflix à finir”, etc.). L’utilisateur peut ensuite ajouter des titres à ces listes et indiquer leur statut d’avancement parmi les suivants :

# Pas vu

# En cours

# Fini

# Ce système permet de visualiser rapidement ce qui reste à regarder ou ce qui a déjà été visionné, tout en gardant une trace organisée de ses habitudes de consommation audiovisuelle.

# L’un des points forts du projet est la fonction de partage, qui permet de rendre ses watchlists visibles à ses amis. Cela favorise l’échange de recommandations, la découverte de nouveaux contenus et une approche plus sociale du suivi des séries et films.

# Accessible depuis une adresse web dédiée (https://mywatchingcompanion.com), l’application offre une interface dynamique, responsive et conviviale, facilitant la navigation, la création et la gestion des watchlists.

# Ce projet a été développé avec des technologies modernes du web, en mettant l'accent sur l’organisation des données, la modularité du code et l’expérience utilisateur.

# Cahier des charges

# L’application MyWatchingCompanion a été pensée pour répondre aux besoins essentiels d’un utilisateur souhaitant organiser son suivi de séries et de films de manière claire et interactive. Elle regroupe plusieurs fonctionnalités clés permettant une gestion fine et personnalisée des contenus audiovisuels. Voici le détail des principales fonctionnalités :

# Ajout, modification et suppression de titres (séries ou films)

# L’utilisateur peut enrichir ses watchlists en y ajoutant de nouveaux titres manuellement. Chaque ajout est associé à des informations essentielles (titre, type – série ou film – et statut). Une fois un titre ajouté, il est possible à tout moment de le modifier (changement de statut, de nom, etc.) ou de le supprimer définitivement de la liste si besoin.

# Suivi de l’avancement des contenus

# Pour chaque titre enregistré, l’utilisateur peut indiquer son statut de progression selon trois niveaux :

# Pas vu : contenu non commencé.

# En cours : contenu actuellement en train d’être regardé.

# Fini : contenu complètement visionné.

# Ce système de suivi permet d’avoir une vision globale sur l’état d’avancement de chaque liste et facilite la planification de futures sessions de visionnage.

# Création de plusieurs watchlists personnalisées

# L’un des atouts majeurs de l’application est de permettre la création illimitée de watchlists, chacune pouvant porter un nom personnalisé. Cela permet à l’utilisateur de structurer ses contenus selon ses propres critères : thématiques, plateformes de streaming, types de contenus, genres, etc.

# Partage de watchlists avec des amis

# Afin d’encourager les interactions sociales, MyWatchingCompanion propose une fonctionnalité de partage. Chaque liste peut être rendue publique et partagée via un lien avec d’autres utilisateurs. Cela permet d’échanger facilement des recommandations de séries ou de films, ou encore de suivre l’avancement d’amis sur des contenus communs.

# Accessibilité en ligne via une plateforme dédiée

# L’application est hébergée en ligne et accessible via l’adresse sécurisée suivante :

# https://mywatchingcompanion.com

# L’utilisateur peut ainsi accéder à son espace personnel à tout moment et depuis n’importe quel appareil disposant d’une connexion Internet, sans installation locale requise.

# 3. Technologies utilisées

# Le projet MyWatchingCompanion repose sur un ensemble de technologies web modernes, permettant de garantir à la fois performance, modularité et maintenabilité. Le choix de ces technologies a été fait en tenant compte des compétences acquises en formation, de la compatibilité entre les outils, et de la simplicité de mise en œuvre pour une application dynamique côté serveur.

# ▸ Front-end : EJS (Embedded JavaScript Templates)

# Le moteur de template EJS a été utilisé pour générer dynamiquement les pages HTML côté serveur. Contrairement aux frameworks front-end comme React ou Vue.js, EJS permet de combiner facilement du HTML classique avec du JavaScript côté serveur, ce qui est idéal pour des projets de taille moyenne comme celui-ci.

# Grâce à EJS, les vues s’adaptent aux données reçues du serveur (par exemple, la liste des titres à afficher), permettant de personnaliser le contenu affiché pour chaque utilisateur tout en maintenant une structure de code claire et réutilisable.

# ▸ Back-end : JavaScript avec Express.js

# Le serveur de l’application a été développé avec Node.js et le framework Express.js, qui offre une architecture souple et rapide à mettre en place. Express permet de gérer les routes HTTP, les requêtes utilisateurs, la gestion des sessions, et la logique métier de manière modulaire.

# Ce choix permet également d’assurer une bonne communication entre la base de données et les vues EJS, en traitant les données côté serveur avant leur affichage.

# ▸ Base de données : SQL Server

# Pour le stockage des données, le projet s’appuie sur SQL Server, un système de gestion de base de données relationnelle robuste et performant.

# La base de données gère plusieurs entités clés comme :

# Les utilisateurs

# Les watchlists

# Les titres (films / séries)

# Les statuts d’avancement

# Des relations bien définies entre ces entités (par exemple, entre un utilisateur et ses watchlists) permettent une intégrité des données et une gestion efficace des interactions.

# L'utilisation de requêtes SQL complexes permet également d'extraire facilement des informations comme les titres partagés, ou les contenus "en cours" d’un utilisateur.

# Architecture du projet

# L’application MyWatchingCompanion suit une architecture MVC (Model – View – Controller), un modèle d’architecture largement utilisé dans les projets web pour organiser le code de manière structurée, claire et maintenable. Cette séparation des responsabilités facilite l'évolution du projet et permet de mieux organiser les différentes couches de l'application.

# ▸ Model (Modèle)

# La couche modèle correspond à la gestion des données et des interactions avec la base de données SQL Server.

# Elle comprend :

# La structure des tables (utilisateurs, watchlists, contenus, statuts)

# Les relations entre ces entités (un utilisateur possède plusieurs watchlists, une watchlist contient plusieurs contenus, etc.)

# Les requêtes SQL utilisées pour insérer, mettre à jour, supprimer ou lire les données nécessaires à l'affichage côté client

# Ce modèle assure la cohérence et la persistance des données, et permet au contrôleur d’interagir efficacement avec le stockage des informations.

# ▸ View (Vue)

# Les vues sont générées dynamiquement grâce à EJS (Embedded JavaScript Templates).

# Lorsqu’un utilisateur accède à une page, le serveur utilise EJS pour injecter les données dans les templates HTML avant de les envoyer au navigateur.

# Par exemple :

# La liste des watchlists de l’utilisateur est injectée dans une page HTML

# Les titres associés à une liste sont affichés dynamiquement selon leur statut

# Cette approche permet de générer des pages personnalisées et dynamiques, tout en conservant la simplicité d’un rendu côté serveur.

# ▸ Controller (Contrôleur)

# Le contrôleur, développé avec Express.js, agit comme une interface entre le modèle et la vue.

# Il est chargé de :

# Recevoir et interpréter les requêtes des utilisateurs (ex. : ajout d’un titre, création d’une watchlist, mise à jour d’un statut)

# Appeler les fonctions du modèle pour récupérer ou modifier les données

# Passer les résultats à la vue EJS pour affichage

# Chaque route Express représente une fonctionnalité de l'application. Par exemple :

# GET /watchlists pour afficher toutes les listes

# POST /watchlist/:id/add pour ajouter un titre dans une liste donnée

# POST /watchlist/:id/share pour générer un lien de partage

# Cette structure MVC garantit une séparation claire entre les données, la logique de traitement, et l’affichage, ce qui rend le projet plus facilement modifiable, testable et évolutif.

# 5. Rôle dans le projet

# Dans le cadre du développement de MyWatchingCompanion, j’ai occupé un rôle central axé principalement sur la gestion de la base de données et le développement back-end, tout en contribuant également à l’interface utilisateur via le moteur de templates EJS.

# ▸ Conception et mise en place de la base de données

# J’ai assuré l’ensemble du travail de modélisation de la base de données relationnelle. Cela comprend :

# L’identification des entités nécessaires (utilisateur, watchlist, contenu, statut…)

# La création des relations entre les tables, en respectant les règles d’intégrité référentielle (clefs primaires, clefs étrangères)

# La rédaction et l’exécution des scripts SQL pour générer la structure de la base (tables, contraintes, index, etc.) dans SQL Server

# La rédaction de requêtes complexes pour interroger efficacement les données, notamment pour les fonctionnalités de partage et de filtrage des contenus par statut

# Ce travail a été essentiel pour assurer une base solide au fonctionnement global de l’application.

# ▸ Développement du back-end avec Express.js

# J’ai ensuite pris en charge une grande partie de la logique métier en développant les routes Express permettant de faire le lien entre l’utilisateur, la base de données et les vues.

# Parmi les tâches réalisées :

# Création des routes pour l’ajout, la modification et la suppression de contenus dans les watchlists

# Mise en place de la logique de gestion des statuts d’avancement

# Développement de la fonctionnalité de création et de partage de watchlists

# Gestion des réponses HTTP, des redirections et de l’affichage dynamique des données

# J’ai veillé à ce que les routes soient claires, sécurisées et efficaces, en structurant le code de façon modulaire pour simplifier la maintenance.

# ▸ Participation à l’interface utilisateur avec EJS

# Bien que mon travail ait été centré sur le back-end, j’ai aussi participé à la création des vues dynamiques avec le moteur de templates EJS.

# J’ai notamment :

# Intégré les données récupérées depuis la base de données dans les vues HTML

# Contribué à la structuration des pages (navigation, affichage des listes, formulaires, statuts…)

# Travaillé sur l’adaptation de l’affichage selon les actions de l’utilisateur

# Cette collaboration entre back-end et front-end m’a permis d’avoir une vision globale de l’application et d'assurer la cohérence entre les données, le traitement serveur et leur présentation à l’utilisateur.

# 6. Déploiement

# L’application MyWatchingCompanion est actuellement hébergée sur un serveur personnel (VPS) que j’ai entièrement configuré dans le cadre du projet. Ce choix d’infrastructure m’a permis de maîtriser toutes les étapes du déploiement, de la mise en ligne jusqu’à la sécurisation de l’accès à l’application.

# Le projet est accessible publiquement via une adresse sécurisée :

# 🔗 https://mywatchingcompanion.com

# L’hébergement sur un VPS (Virtual Private Server) présente plusieurs avantages :

# Contrôle total sur l’environnement serveur (système d’exploitation, services, ports, sécurité…)

# Personnalisation de la configuration web selon les besoins de l’application (reverse proxy, gestion SSL, nom de domaine personnalisé…)

# Expérience concrète dans l’administration système, compétence essentielle en BTS SIO

# Le serveur web a été configuré pour accueillir l’application Node.js, assurer le routage via Express.js, et servir les vues dynamiques générées avec EJS. Un certificat SSL a également été mis en place pour garantir un accès sécurisé via HTTPS.

# Grâce à ce déploiement, l’application est disponible à tout moment depuis n’importe quel appareil connecté à Internet, ce qui facilite les démonstrations, les tests utilisateurs et les retours en conditions réelles.

# 7. Bilan personnel

Le développement de MyWatchingCompanion a représenté une expérience particulièrement enrichissante et formatrice dans le cadre de ma formation en BTS SIO, spécialité SLAM. Ce projet m’a permis de mettre en pratique de nombreuses compétences acquises en cours, tout en me confrontant aux exigences d’un projet web complet, depuis la conception jusqu’à la mise en production.

J’ai particulièrement renforcé mes compétences en :

Développement back-end, grâce à l’utilisation du framework Express.js pour la gestion des routes, des données, et de la logique métier.

Gestion de base de données relationnelle, avec SQL Server pour modéliser, structurer et interroger efficacement les données.

Création d’interfaces dynamiques, à travers le moteur de templates EJS, permettant d’adapter les pages HTML en fonction des données et des actions des utilisateurs.

Déploiement et hébergement, en configurant moi-même un VPS personnel, incluant le déploiement de l’application, la configuration du nom de domaine, la mise en place du HTTPS, et la gestion du serveur.

Ce projet m’a aussi permis de développer une vision globale d’un développement web structuré, en prenant en compte :

Les besoins fonctionnels de l’utilisateur final (simplicité, personnalisation, partage)

Les contraintes techniques (sécurité, performance, évolutivité)

L’organisation du code selon une architecture MVC claire et maintenable

Ce travail a nécessité un investissement personnel important, tant en temps qu’en autonomie, mais il m’a offert l’opportunité de consolider mes acquis et de gagner en confiance dans mes capacités à gérer des projets techniques de A à Z.

Enfin, pour compléter cette documentation, des captures d’écran de l’interface utilisateur, ainsi que des schémas de la base de données ou de l’architecture du projet, pourront être ajoutés afin d’illustrer concrètement le fonctionnement de l’application.

# 8. Annexes

Une image contenant texte, logiciel, Page web, Site web

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, logiciel, Page web

Le contenu généré par l’IA peut être incorrect.