



O'COMICS

Dossier de projet

Projet réalisé dans le cadre de la présentation au Titre
Professionnel Développeur Web et Web Mobile - DWWM -

Présenté par

Benjamin GRECO

Promotion Vega - 2023

Organisme de formation

Ecole O'Cock

Session du jeudi 26 octobre 2023

SOMMAIRE

SOMMAIRE	1
INTRODUCTION	3
RÉSUMÉ DU PROJET	4
CAHIER DES CHARGES	5
Présentation	5
I. Description	5
II. User Stories (v1)	5
III. Evolution	6
IV. User Stories (v2)	6
V. M.V.P (Minimum Viable Product)	6
VI. Cible	7
Description fonctionnelles	7
I. Arborescence	8
II. Listes des routes	8
III. MCD (Modèle Conceptuel de Données)	10
IV. MLD (Modèle Logique des Données)	11
V. MPD (Modèle Physique de Données)	11
Wireframes	14
Dictionnaire de données	15
Convention de nommage	21
I. VS Code	21
II. Git & Github	22
SPÉCIFICATIONS TECHNIQUES DU PROJET	23
I. Technos	23
A. Front - Framework React	23
B. Back - Framework Symfony	23
II. Besoins	23
III. Navigateurs compatibles.	23
IV. Versionning	23
V. Gestion de projet	24

VI. L'équipe et les rôles	24
VII. Les sprints	24
COMPÉTENCES DU RÉFÉRENTIEL	25
I. Développer la partie front-end d'une application web ou web mobile	25
A. CP 1. Maquetter une application	25
B. CP 2. Réaliser une interface utilisateur web statique et adaptable	26
C. CP 3. Développer une interface utilisateur web dynamique	29
II. Développer la partie back-end d'une application web ou web mobile	33
A. CP 5. Créer une base de données	33
B. CP 6. Développer les composants d'accès aux données	35
C. CP 7. Développer la partie back-end d'une application web ou web mobile	39
RÉALISATIONS TECHNIQUE DU PROJET	44
I. Front-end	44
A. Main.tsx	44
B. Index.tsx	44
C. App.tsx	46
D. Home.tsx	48
II. Back-end	50
A. Entity	50
B. Controller	51
RÉALISATIONS PERSONNELLES	53
A. MarvelApiUrlGenerator	53
B. ApiRegisterController	61
C. Access Control	63
JEU D'ESSAI	65
VULNÉRABILITÉ DE SÉCURITÉ	70
UTILISATION DE SITES ANGLOPHONES	75
CONCLUSION	76
ANNEXES	77

INTRODUCTION

Mon parcours atypique m'a conduit à une reconversion professionnelle, j'ai commencé ma carrière dans l'industrie de la restauration en tant que directeur et propriétaire de restaurant. Mon expérience dans ce secteur m'a permis de développer des compétences précieuses en gestion, en communication et en résolution de problèmes. Cependant, la pandémie de COVID-19 a mis à l'épreuve l'industrie de la restauration, et j'ai été confronté à la nécessité de réfléchir à de nouvelles opportunités.

Depuis longtemps, j'ai un intérêt profond pour l'informatique. En dehors de mes heures de travail, j'ai consacré du temps à découvrir le monde de la programmation et du développement web. Cette curiosité s'est rapidement transformée en une véritable vocation, et j'ai décidé de poursuivre une carrière dans le domaine du développement web.

Le développement web m'a particulièrement attiré en raison de son potentiel à combiner la créativité avec la résolution de problèmes techniques. Créer des sites web interactifs et dynamiques est à la fois un défi stimulant et une opportunité de laisser libre cours à la créativité. Je suis convaincu que le web est l'avenir, et je souhaite faire partie de cette révolution technologique.

Mon objectif dans le cadre de ce projet est d'acquérir une solide compréhension des technologies web, de développer des compétences techniques de pointe et de contribuer à la création de solutions web innovantes. Je suis déterminé à apprendre et à m'améliorer constamment pour exceller dans ce domaine.

J'ai choisi de suivre ma formation en développement web à l'école O'Clock, une institution réputée pour son enseignement de qualité. Le format télé présentiel de la formation m'a permis de bénéficier d'une éducation de premier plan, tout en ayant la flexibilité de suivre les cours à distance. Cette approche novatrice correspond parfaitement à mon désir d'apprendre de manière efficace tout en m'adaptant aux contraintes de la situation actuelle.

Dans le cadre de cette formation de plus de 5 mois, j'ai eu 3 mois de formation socle, un mois de spécialisation et un dernier mois de projet professionnel appelé "Apothéose" en équipe de 4 développeurs. C'est ce projet que je vais vous présenter dans les pages qui suivent. Je précise que ce projet a été développé sous environnement **Linux**.

RÉSUMÉ DU PROJET

Étant moi-même féru de comics dans mes jeunes années et possédant des collections incomplètes, je me suis mis à la recherche de sites ou plateformes sur lesquelles je pourrais chercher les numéros manquants de mes collections. Il existe effectivement des sites où il est possible d'acheter des comics et d'autres où il est possible de troquer des bandes dessinées classiques. J'ai donc eu l'idée de créer le projet O'Comics.

Le projet O'Comics a pour objectif de créer une plateforme en ligne dédiée aux passionnés de l'univers des comics. Cette plateforme permettra aux utilisateurs de découvrir de nouveaux comics, d'interagir avec d'autres membres de la communauté pour échanger des informations et des recommandations, et en fin de compte, de compléter leurs collections de comics. Les utilisateurs auront la possibilité de créer deux listes de favoris lorsqu'ils sont connectés en tant qu'utilisateurs enregistrés : l'une pour les comics qu'ils recherchent et l'autre pour les comics qu'ils possèdent déjà.

À terme, la plateforme vise à faciliter les échanges de comics entre les utilisateurs, en favorisant les transactions et les interactions au sein de la communauté. Le public cible de ce projet est principalement composé d'hommes âgés de 25 à 55 ans, partageant une passion pour l'univers riche et diversifié des comics.

CAHIER DES CHARGES

Présentation

I. Description

Le projet O'Comics a pour objectif de mettre à disposition une plateforme en ligne dédiée aux passionnés de l'univers des comics, où ils pourront découvrir de nouveaux comics et en échanger avec d'autres membres afin de compléter leurs collections respectives.

O'comics se veut un espace d'échange et de partage entre les membres d'une communauté qui se retrouve autour d'une passion commune, que ce soit pour échanger ou simplement discuter!

Côté front

Espace Visiteur

Les visiteurs peuvent:

- visualiser la liste de tous les comics enregistrés,
- visualise la liste de tous les personnages enregistrés ainsi que les comics où le personnage en question apparaît.
- Créer un compte utilisateur afin d'accéder à des fonctionnalités avancées.

Espace Utilisateur

- Les utilisateurs peuvent créer un compte personnel pour ajouter, dans leur collection, les comics qu'ils possèdent ainsi que les comics qui les intéressent.
- Les utilisateurs ont accès à une fonctionnalité d'échanges via un formulaire de contact entre utilisateurs, pour pouvoir échanger des comics entre eux.

Côté back

Espace admin

- Les administrateurs peuvent ajouter, éditer et supprimer des comics
- Les administrateurs peuvent gérer les comptes utilisateurs

II. User Stories (v1)

- En tant que visiteur, je veux pouvoir consulter tous les comics disponibles afin d'en découvrir de nouveaux.
- En tant que visiteur, je veux pouvoir consulter le détail de chaque comics disponible.
- En tant que visiteur, je veux pouvoir consulter l'historique détaillé d'un personnage et accéder aux comics associés.
- En tant que visiteur, je veux pouvoir créer un compte utilisateur.
- En tant qu'utilisateur, je veux pouvoir ajouter la mention « je possède » ou « je recherche » sur chacun des comics.
- En tant qu'utilisateur, je veux pouvoir accéder à une page d'échange qui liste tous les utilisateurs possédant les comics que je recherche
- En tant qu'utilisateur, je veux pouvoir envoyer un message via un formulaire de contact aux autres utilisateurs afin de leur proposer un échange.
- En tant qu'utilisateur, je veux pouvoir échanger avec les autres utilisateurs.
- En tant qu'administrateur, je veux pouvoir ajouter un comics.
- En tant qu'administrateur, je veux pouvoir éditer un comics.
- En tant qu'administrateur, je veux pouvoir supprimer un comics.

III. Evolution

Côté front

Espace Utilisateur

- Tous les comics souhaités/recherchés seront listés sur une page d'échange avec le pseudo de la personne qui souhaite/recherche obtenir un comics donné.
- Mise en contact des utilisateurs via une messagerie intégrée dans notre application web

Espace admin

- Les administrateurs pourront modérer les échanges entre utilisateurs.
- Ajout du CRUD des personnages
- Ajout du CRUD des utilisateurs

IV. User Stories (v2)

- En tant que visiteur, je veux pouvoir contacter l'administrateur du site afin de pouvoir remonter d'éventuels problèmes.
- En tant qu'administrateur, je peux ajouter, éditer et supprimer les comptes utilisateurs
- En tant qu'administrateur, je veux pouvoir réinitialiser les mots de passe des utilisateurs.

V. M.V.P (Minimum Viable Product)

- Accéder à la page d'accueil du site et consulter une sélection de comics et de personnages
- Avoir accès aux comics liés à un personnage

- Accéder à la page des comics avec la possibilité de les mettre dans deux listes distinctes, les comics possédés et les comics souhaités
- Accéder à la page de ces deux listes afin de les visualiser et de pouvoir supprimer un comics d'une des listes si l'utilisateur le souhaite.

VI. Cible

Le public cible est principalement composé d'hommes âgés de 25 à 35, passionnés et collectionneurs de bandes dessinées de l'univers des Comics Marvel et DC.

1. Passionnés de Comics

Notre site vise les amateurs de bandes dessinées qui souhaitent explorer, organiser et compléter leur collection de comics. Ces utilisateurs sont intéressés par les fonctionnalités telles que la création de comptes personnels, l'ajout de comics à une liste de favoris et à une liste de comics recherchés, et la possibilité de discuter avec d'autres amateurs de comics de leurs passions communes.

2. Collectionneurs en quête d'échanges

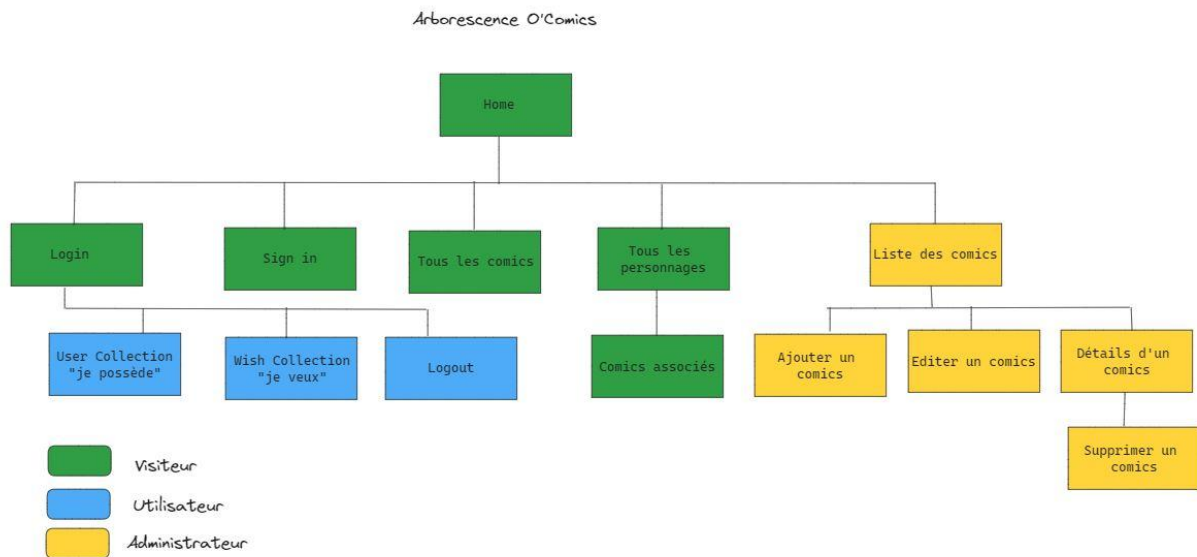
Notre fonctionnalité de liste de comics recherchés/possédés indique que nous ciblons également les collectionneurs qui sont intéressés par des échanges pour compléter leur collection.

3. Amateurs de découvertes

Les visiteurs qui ne sont pas inscrits pourraient être des personnes découvrant les univers Marvel et DC grâce aux séries ou aux films, bien plus populaires de nos jours que les Comics. Ils auront la possibilité de consulter la liste complète des comics et ainsi en découvrir plus sur ces univers, sur l'origine de leurs personnages favoris...

Description fonctionnelles

I. Arborescence



II. Listes des routes

- Route Front-office -

Description de la route	Méthodes HTTP	Endpoint
Accueil	Get	/
Connexion	Post	/login
Se déconnecter	Post	/logout
Inscription	Post	/register
Liste des comics	Get	/comics
Liste des personnages	Get	/personnages
Liste des comics possédés	Get	/ownlist
Liste des comics souhaités	Get	/wishlist
Back-office	Get	/admin

- Route back-office

Description de la route	Méthodes HTTP	Endpoint
Accueil du back-office	Get	/
Détail d'un comics	Get	/admin/comics/{id}
Créer un nouveau comics	Post	/admin/comics/add
Éditer un comics existant	Post	/admin/comics/edit/{id}
Supprimer un comics	Post	/admin/comics/delete/{id}

- Route API

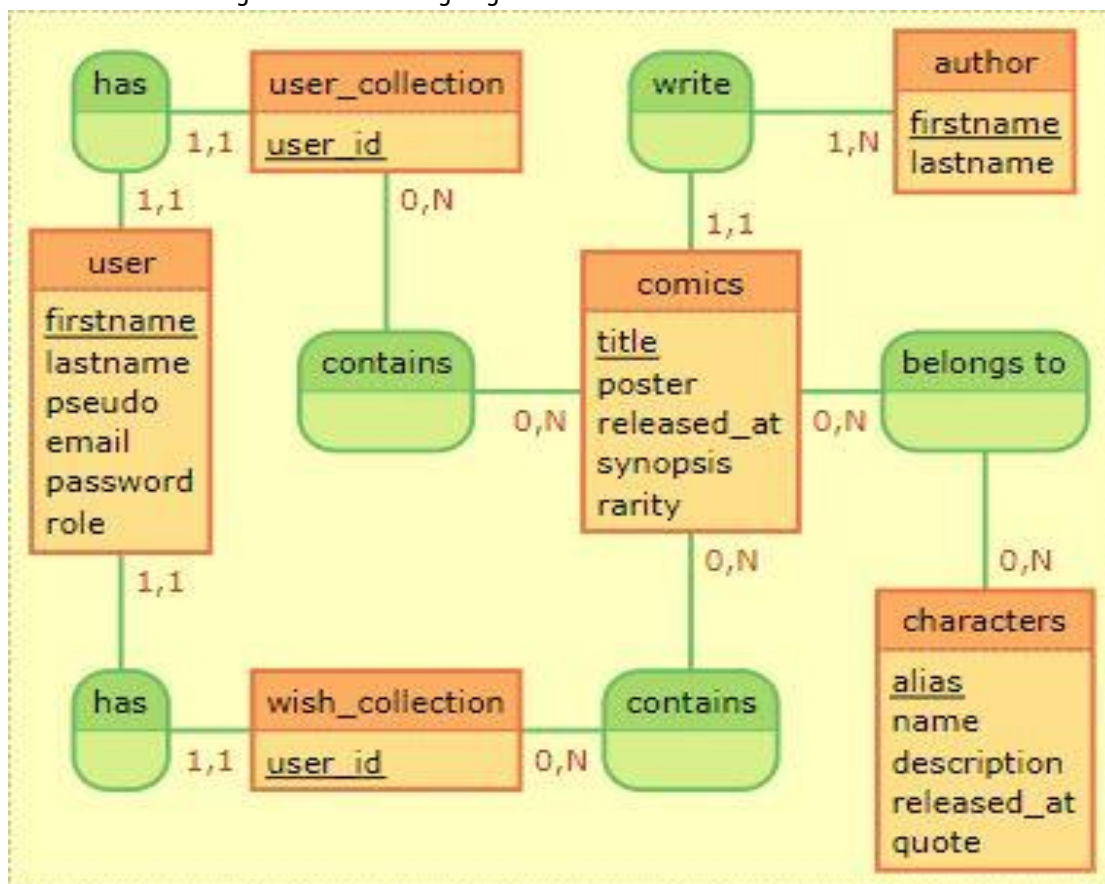
Description de la route	Méthodes HTTP	Endpoint
Liste des personnages	Get	/api/character
Détail d'un personnage	Get	/api/character/{id}
Liste des comics	Get	/api/comics
Détail d'un comics	Get	/api/comics/{id}
5 personnages pour la home	Get	/api/home-character
9 comics pour la home	Get	/api/home-comics
Ajout d'un comics de la liste des comics souhaités	Post	/api/wishlist/add/{comicsId}
Suppression d'un comics de la liste des comics souhaités	Delete	/api/wishlist/remove/{comicsId}
Ajout d'un comics de la liste des comics possédés d'un utilisateur	Post	/api/ownedlist/add/{comicsId}
Suppression d'un comics de la liste des comics possédés d'un utilisateur	Delete	/api/ownedlist/remove/{comicsId}
Récupération d'un token JWT	Get	/api/login_check
Récupérer l'utilisateur courant	Get	/api/user
Création d'un nouvel utilisateur	Post	/api/register

- Route API v2 - fonctionnelle sur Insomnia, non mise en place.

Description de la route	Méthodes HTTP	Endpoint
Création d'un comics	Post	/api/admin/comics/add
Edition d'un comics	Put	/api/admin/comics/{id}/update
Suppression d'un comics	Delete	/api/admin/comics/{id}/delete

III. MCD (Modèle Conceptuel de Données)

Le MCD a été réalisé grâce à l'outil en ligne gratuit Mocodo.



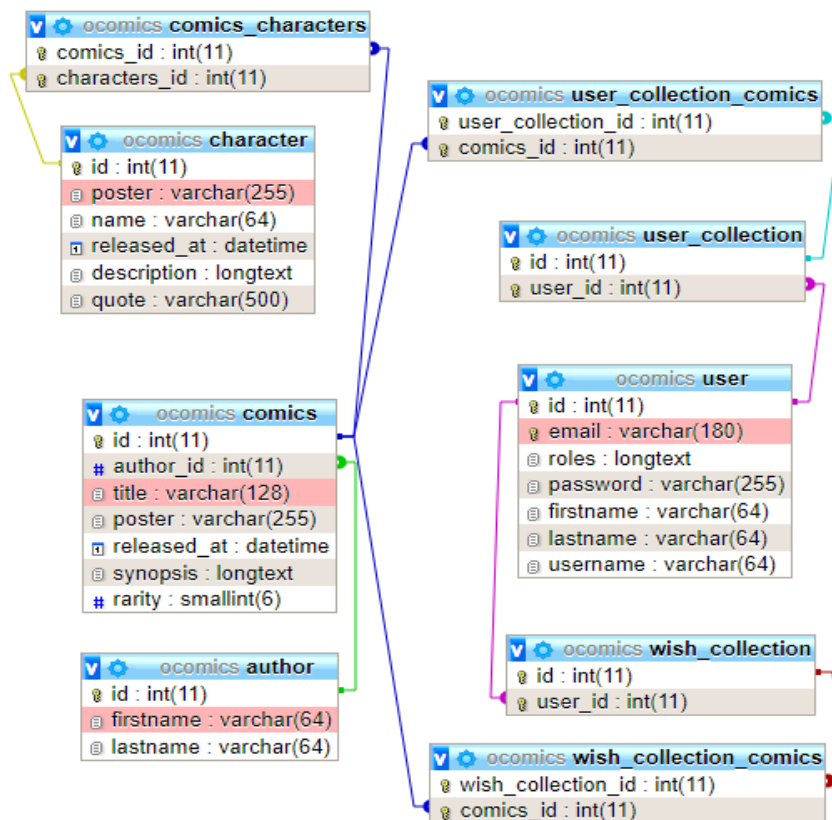
IV. MLD (Modèle Logique des Données)

Le MLD est la représentation textuelle du MPD, disponible en dessous.

```
comics (id, title, poster, released_at, synopsis, rarity, #author_id)
character (id, poster, name, released_at, description, quote)
comics_characters (#comics_id, #characters_id)
author (id, lastname, firstname)
user (id, email, roles, password, firstname, lastname, username)
user_collection (id, #user_id)
wish_collection (id, #user_id)
user_collection_comics (#user_collection_id, #comics_id)
wish_collection_comics (#wish_collection_id, #comics_id)
```

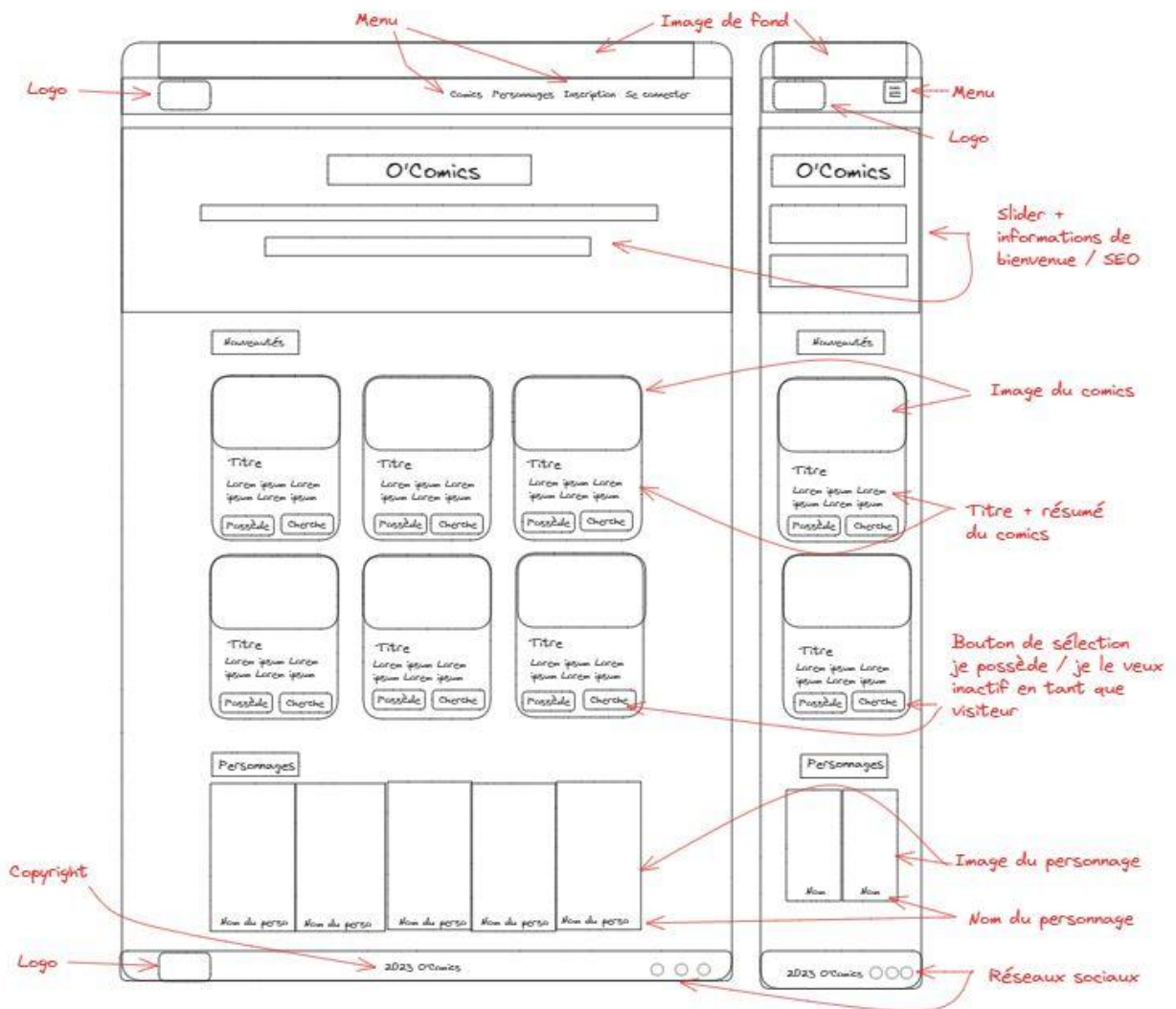
V. MPD (Modèle Physique de Données)

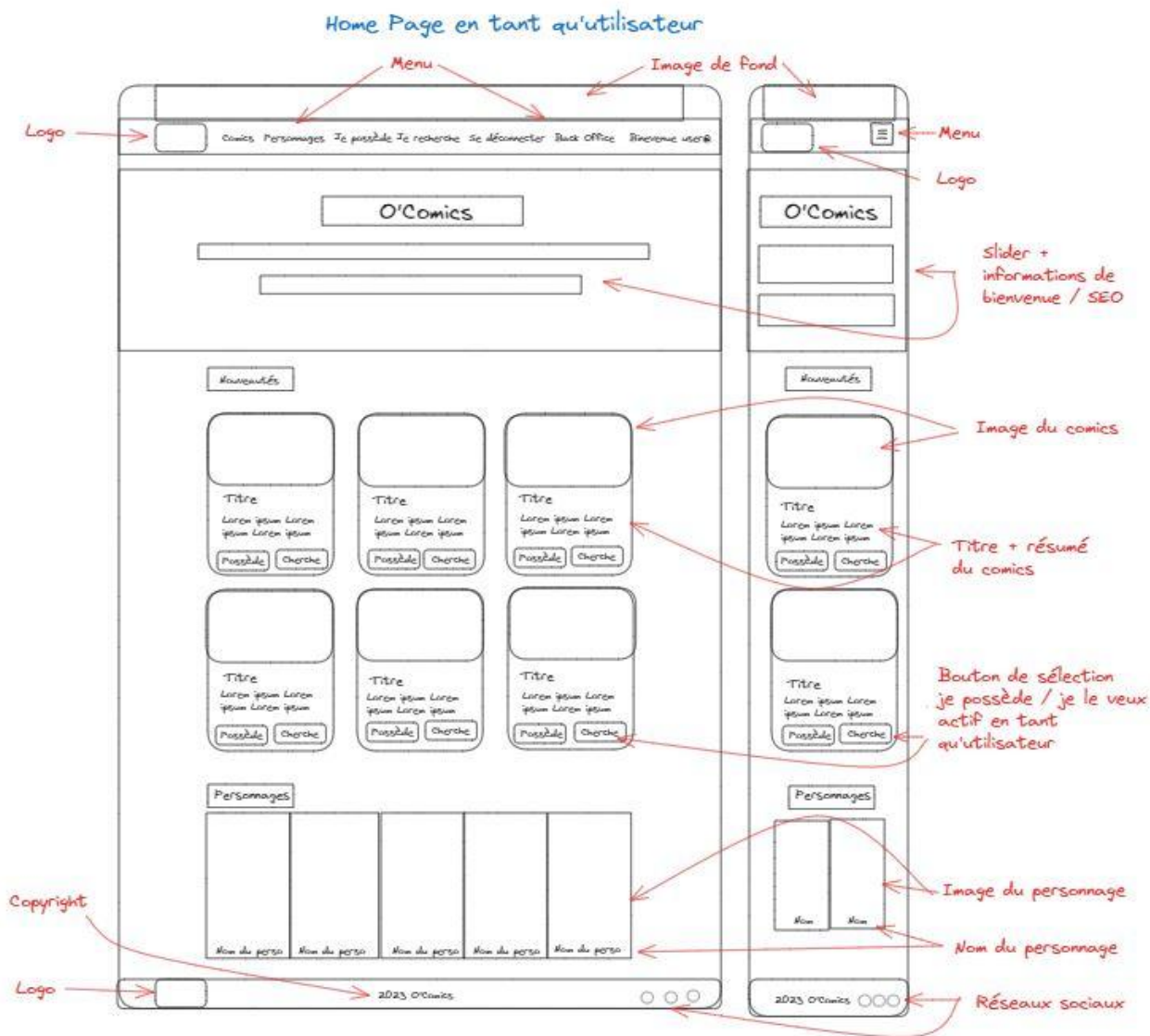
MPD généré via PHPMyAdmin, après la création de la base de données avec Doctrine.

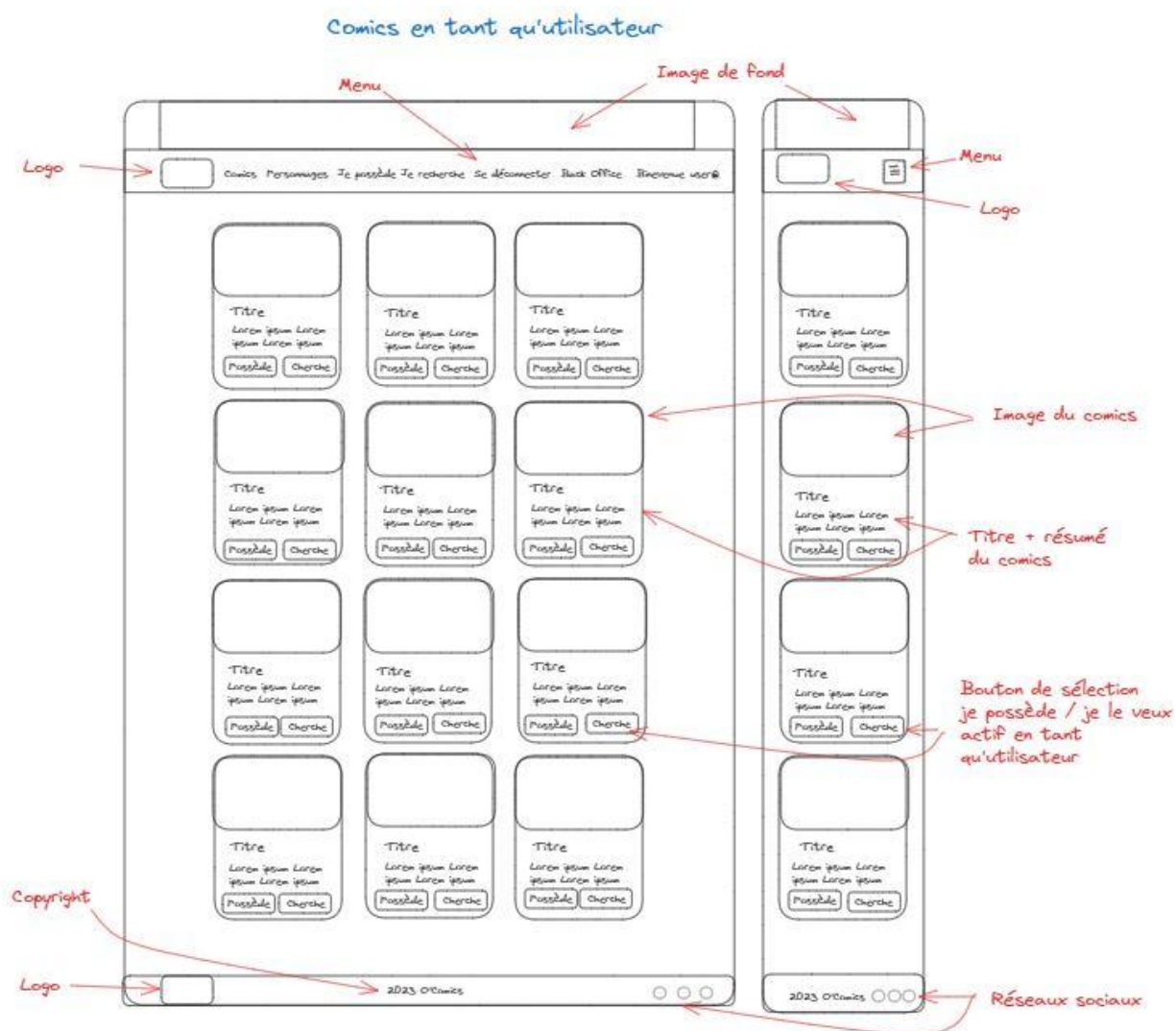


Wireframes

Home Page en tant que visiteur







Dictionnaire de données

<u>Table</u>	<u>Champ</u>	<u>Types de données</u>	<u>Spécificités</u>	<u>Description</u>
comics	id	INT	NOT NULL, PRIMARY KEY	Identifiant du comics
comics	title	VARCHAR(128)	NOT NULL	Titre du comics
comics	poster	VARCHAR(255)	NOT NULL	Couverture du comics sous forme d'url
comics	released_at	DATETIME	NOT NULL	Date de publication
comics	synopsis	TEXT	N/A	Intrigue du comics
comics	rarity	TINYINT	NULL	Rareté de la BD, de 1 à 5
comics	author_id	INT	NOT NULL,	Identifiant du

			FOREIGN KEY	comics
character	id	INT	NOT NULL, PRIMARY KEY AUTO_INCREMENT	Identifiant du personnage
character	poster	VARCHAR(255)	NOT NULL	Visuel du personnage
character	name	VARCHAR(64)	NULL	nom du personnage
character	description	longtext	NOT NULL	L'histoire résumé du personnage
character	quote	varchar(500)	NULL	Le slogan, la phrase 'mythique' du personnage

character	released_at	DATETIME	NOT NULL	Date de première apparition
user_collection	id	INT	NOT NULL, PRIMARY KEY AUTO_INCREMENT	Identifiant de la collection de l'utilisateur
user_collection	user_id	INT	NOT NULL, FOREIGN KEY	Identifiant de l'utilisateur qui détient la collection
wish_collection	id	INT	NOT NULL, PRIMARY KEY AUTO_INCREMENT	Identifiant de la collection de l'utilisateur
wish_coll	user_id	INT	NOT	Identifica

ection			NULL, FOREIGN KEY	nt de l'utilisateur ur qui détient la collectio n
author	id	INT	NOT NULL, PRIMARY KEY AUTO_IN CREMEN T	Identifia nt de l'auteur
author	firstname	VARCHA R(64)	NOT NULL	Nom de l'auteur
author	lastname	VARCHA R(64)	NOT NULL	Prénom de l'auteur
user	id	INT	NOT NULL, PRIMARY KEY AUTO_IN CREMEN	Identifia nt de l'utilisate ur

			T	
user	firstname	VARCHAR(64)	NOT NULL	Nom de l'utilisateur
user	lastname	VARCHAR(64)	NOT NULL	Prénom de l'utilisateur
user	username	VARCHAR(64)	NOT NULL	Pseudonyme de l'utilisateur
user	email	VARCHAR(128)	NOT NULL	Email de l'utilisateur
user	password	VARCHAR(255)	NOT NULL	Mot de passe de l'utilisateur hashé
users	rôle	VARCHAR(64)	NOT NULL	Rôle de l'utilisateur (utilisateur, admin..)
Table pivot				

user_collection_comics	user_collection_id	INT	NOT NULL, FOREIGN KEY	Clé primaire composite
user_collection_comics	comics_id	INT	NOT NULL, FOREIGN KEY	Clé primaire composite
wish_collection_comics	user_collection_id	INT	NOT NULL, FOREIGN KEY	Clé primaire composite
wish_collection_comics	comics_id	INT	NOT NULL, FOREIGN KEY	Clé primaire composite
comics_character	comics_id	INT	NOT NULL, FOREIGN KEY	Clé primaire composite
comics_character	character_id	INT	NOT NULL, FOREIGN KEY	Clé primaire composite

Convention de nommage

I. VS Code

Les commentaires ainsi que la documentation se feront en anglais

Côté backend :

Les noms des **variables**, des **fonctions**, des **méthodes** et des **attributs** sont écrits en camelCase.

Les classes en PascalCase,

dans la bdd, on nomme les tables et les différents champs en snake_case.

Côté Front :

Les composants commencent toujours par une **majuscule** et sont écrits en **PascalCase**.

camelCase : **Les fichiers, les states, variables** .

Les **classes** et **certaines fichiers en URL** seront en **kebab-case**.

Les types des composants :

-Composant View : affiche uniquement un rendu des données.

-Composant Button : affiche uniquement une vue actionnable

II. Git & Github

Les Branches

'master' : Branche principale du projet. C'est là que se trouve le code de production stable.

'dev' : branche de développement. Les nouvelles fonctionnalités testées sont fusionnées sur cette branche avant d'être testées et poussées vers master.

'feat/nom-de-la-fonctionnalité' : pour les nouvelles fonctionnalités en cours de dev.

'bugfix/nom-du-bug' : Pour les corrections de bugs en cours de dev.

Les Pull Requests :

Feature: Ajoute la fonctionnalité X : Pour les PR liées aux nouvelles fonctionnalités.

Bugfix: Corrige le bug Y : Pour les PR liées aux corrections de bugs.

Refactor: Réorganise le module Z : Pour les PR liées aux factorisations.

Messages de commit :

9 types sont disponibles :

build : changements qui affectent le système de build ou des dépendances externes (npm, make...)

ci : changements concernant les fichiers et scripts d'intégration ou de configuration (Travis, Ansible, BrowserStack...)

feat : ajout d'une nouvelle fonctionnalité

fix : correction d'un bug

perf : amélioration des performances

refactor : modification qui n'apporte ni nouvelle fonctionnalité ni d'amélioration de performances

style : changement qui n'apporte aucune altération fonctionnelle ou sémantique (indentation, mise en forme, ajout d'espace, renommage d'une variable...)

docs : rédaction ou mise à jour de documentation

test : ajout ou modification de tests

SPÉCIFICATIONS TECHNIQUES DU PROJET

I. Technos

● Front - Framework React

- React
- Tailwind pour le CSS du front-office
- React-Router-Dom pour le routage
- TypeScript
- Axios, pour l'authentification à notre API

La partie front-end de notre application a été réalisée avec React, framework développé Facebook utilisant Javascript et un DOM virtuel bien plus rapide que le DOM original, la possibilité de réutiliser les composants de React permet de gagner du temps, le flux de données unidirectionnel fournit un code stable, de plus la librairie est open-source ce qui facilite la recherche d'informations et de résolution de problème.

React permettant de faire ce que l'on appelle une SPA (Single Page Application), nous avons utilisé React Router afin d'avoir des routes virtuelles (seuls les composants présents sur la route sont placés dans le DOM virtuel, le résultat est direct. L'utilisateur n'a pas l'impression d'avoir subi un chargement de page).

● Back - Framework Symfony

- l'ORM Doctrine pour la gestion de la base de données
- Moteur de templating Twig pour l'affichage du back-office, avec Bootstrap pour le CSS.
- package JWT token, pour l'authentification à notre API.
- Base de données: MySQL
- Gestion de la base de données: Adminer, PHPMYAdmin
- Insomnia pour tester notre API

La partie back-end a été réalisée à l'aide du framework PHP Symfony dont l'approche se base sur des composants et du modèle MVC (Model-View-Controller).

MakerBundle a été utilisé pour permettre de générer rapidement des commandes vides, contrôleurs, classes de formulaire, etc. en une ligne de commande.

L'ORM Doctrine a été utilisé pour la gestion de base de données, en tant qu'ORM son rôle est de transformer les objets PHP en écriture SQL, et récupérer les écritures SQL et les transformer en objet PHP

Le bundle Lexik JWT Authentication pour l'utilisation de token JWT, ce jeton sera envoyé à chaque requête que l'on souhaite effectuer auprès d'une API afin de s'authentifier. Il contient toutes les informations nécessaires à notre identification.

Notre serveur Symfony est lié à une base de données MySQL, administré via le gestionnaire de base de données Adminer ou PHPMysqlAdmin selon les goûts de chacun.

II. Besoins

Nous aurons besoin d'une clé API afin de récupérer des données cohérentes pour créer notre propre API : <https://developer.marvel.com/>

Nous avons également besoin, côté back, d'un logiciel pour tester les différentes routes de notre API avant de les mettre à disposition de l'équipe Front. Pour cela, nous avons choisi de réaliser nos tests sur l'outil Insomnia.

III. Navigateurs compatibles.

Nous avons testé l'application sur Chrome v.117 version web et mobile. Firefox Version 118.0.2 web, elle fonctionne correctement sur chaque support testé.

IV. Versionning

Nous avons utilisé Git & Github pour gérer le versionning de notre application, avec une organisation sur deux repository séparé, un pour le front, un pour le back.

V. Gestion de projet

Nous avons utilisé la méthodologie Scrum, qui découle des méthodes Agile, pour la gestion de notre projet. Mise en place d'un Trello avec une organisation en sprint d'une semaine, la participation pour toute l'équipe à un daily scrum journalier afin de s'organiser de la meilleure façon possible.

Chaque jour, l'équipe se réunit pour une courte réunion pour discuter de la progression des tâches et des obstacles. Trello peut être utilisé pour mettre à jour l'état des cartes et partager les informations.

Après chaque sprint, l'équipe peut revoir et prioriser le backlog produit en fonction des retours d'information, ce qui alimente la planification du prochain sprint.

VI. L'équipe et les rôles

L'équipe est composé de 4 personnes:

Benjamin Greco : Back Dev/ Product Owner / Git Master

Nicolas Deniau : Back Dev / Lead Dev Back / Scrum Master

Hocine Aït : Front Dev / Lead Dev Front / Git Master

Thanh-binh Nguyen : Front Dev / Scrum Master

Référents Techniques : Benjamin / Thanh-binh

VII. Les sprints

Nous disposons de 4 semaines pour la réalisation de ce projet, nous avons donc découpé le projet en 4 sprints.

Sprint 0 : réalisation du cahier des charges, User stories, wireframes, Trello, MCD, MLD.

Sprint 1 : Côté back: Mise en place de l'architecture, MVC, création de la base de données avec Doctrine, création des modèles et des premiers contrôleurs de l'API, création d'une commande pour utiliser l'API de Marvel.com

Côté front : Création de la page d'accueil et page des comics

Sprint 2 : Côté front : mise en place du "login" et du "register", et du JWT pour communiquer avec le back et récupérer les infos des Users (Rôle). Mise en place de Axios pour communiquer avec la base de données. Création des routes. Création de la page des personnages.

Côté back : Création du backoffice, création du CRUD comics, création du login côté back, création des "Userlist" et "Wishlist".

Sprint 3 : Côté front : mise en place des pages "je possède" et "je recherche", finalisation du register, affichage conditionnelle de la "navbar", déploiement. (Vercel)

Côté back : Déploiement sur la VM cloud de l'école, changement de http pour https pour communiquer avec Vercel, réglages des derniers bugs avant présentation du projet.

COMPÉTENCES DU RÉFÉRENTIEL

CP 1 Maquetter une application

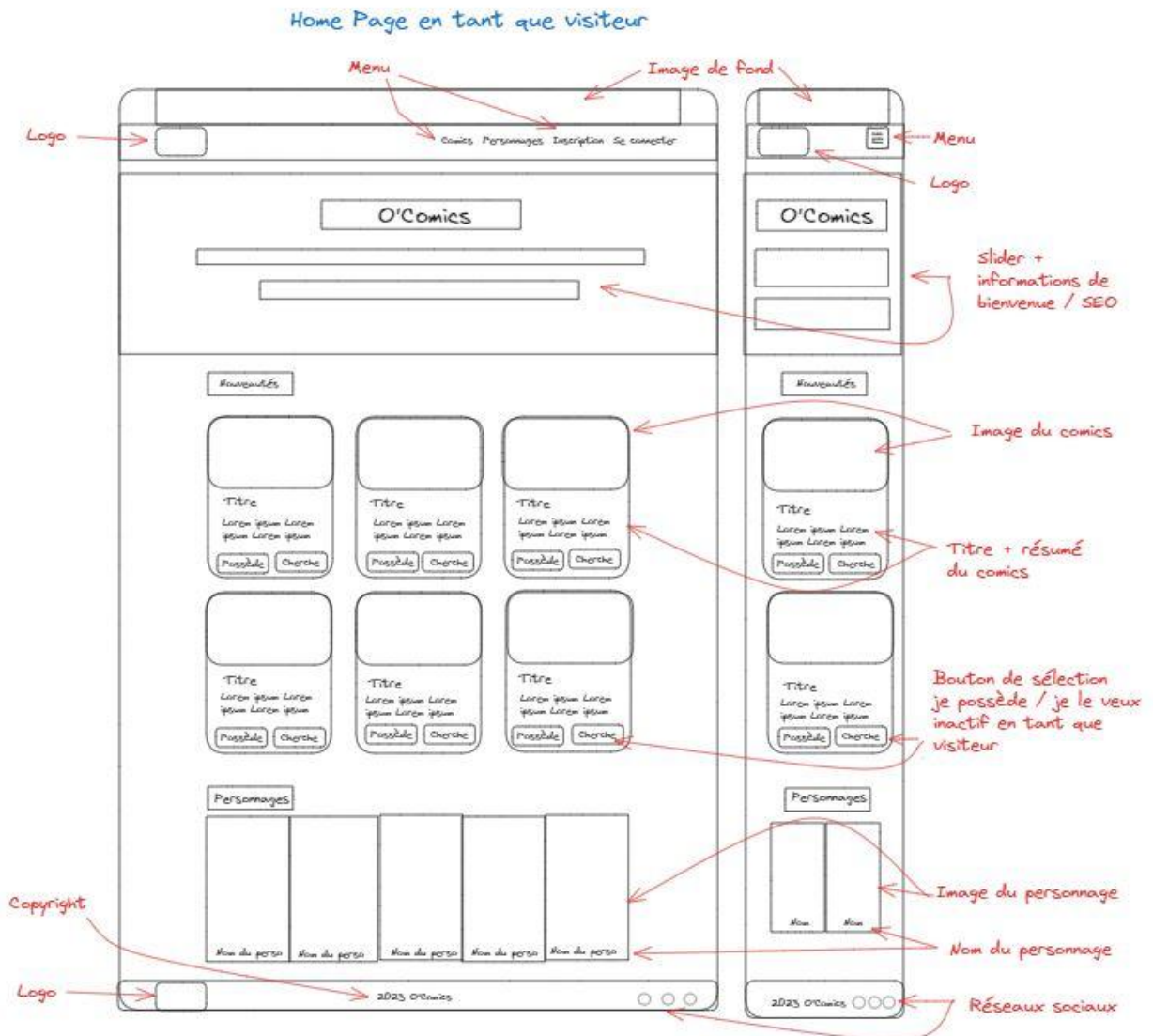
Comme indiqué dans le cahier des charges, vous trouverez ici la homepage de notre application créée avec Excalidraw. Ici, pour la réalisation de la page d'accueil visiteur, on se base sur les User Stories de la page d'accueil.

Nous avons créé deux vues, Desktop et Mobile car le site sera mobile-first

On commence par mettre en place le header avec le logo à gauche et les boutons de connexion à droite offrant un flow naturel qu'on retrouve sur la majorité des sites web.

Ensuite on met en place le slider et les messages d'accueil qui serviront pour le SEO, les différents composants de la page, comics et personnages.

On termine par le footer qui reprend le logo, le copyright et les réseaux sociaux.



CP 2 Réaliser une interface web statique et adaptable

Dans notre projet, côté front, nous avons opté pour une combinaison de technologies comprenant React, Tailwind CSS, et TypeScript pour des raisons spécifiques :

React est le cœur de notre application front-end. Il s'agit d'une bibliothèque JavaScript populaire pour la création d'interfaces utilisateur interactives. React nous permet de concevoir des composants réutilisables

qui rendent le développement d'interfaces utilisateur plus modulaire et plus facile à maintenir. Son modèle de composant permet de construire des applications complexes en assemblant des éléments plus petits.

Tailwind CSS intervient dans la gestion des styles de notre application. Il s'agit d'un framework CSS "utility-first" qui facilite la conception des éléments visuels de notre site. Au lieu de définir des styles personnalisés à partir de zéro, Tailwind CSS nous offre un ensemble complet de classes CSS préconçues que nous pouvons appliquer directement dans notre code HTML. Cela accélère le processus de développement en évitant la nécessité de créer des feuilles de style CSS distinctes. De plus, la normalisation des noms de classes avec Tailwind facilite la collaboration et garantit une cohérence dans l'apparence de notre application.

En résumé, la combinaison de React, Tailwind CSS et TypeScript dans notre projet vise à optimiser le processus de développement en offrant une architecture modulaire, une gestion des styles efficace et une sécurité accrue grâce à la détection des erreurs de typage. Ces choix technologiques sont destinés à simplifier le développement, à accélérer la création de fonctionnalités et à améliorer la qualité et la maintenabilité de notre application.

Nous avons donc commencé par concevoir nos pages en respectant les wireframes préétablis. Ces wireframes nous ont servi de guide pour organiser les éléments sur chaque page. Dans cette première étape, nous avons rempli les pages avec du texte fictif de type "Lorem ipsum" afin d'évaluer l'agencement des éléments.

Nous avons suivi une approche "mobile-first" avec Tailwind CSS. Cela signifie que nous avons commencé par concevoir la version mobile de notre site (pour des largeurs inférieures à 425px). Avec Tailwind CSS, les classes de style sont appliquées directement dans le code HTML, ce qui simplifie la gestion des styles pour chaque composant ou page.

Après avoir mis en place la version mobile, nous avons étendu notre travail pour couvrir les versions desktop et tablette. Nous avons veillé à ce que le site soit entièrement responsive, offrant une expérience optimale sur toutes les tailles d'écran.

Par exemple, dans ce code : `"mx-40 lg:mx-56"` définissent les marges horizontales pour la section principale du contenu. La classe `mx-40` s'applique à toutes les tailles d'écran, tandis que `lg:mx-56` s'applique uniquement aux écrans de taille «large» (typiquement des écrans d'ordinateurs de bureau). Cela signifie que sur des écrans plus larges, la section aura une marge plus grande pour laisser plus d'espace sur les côtés.

```

1 import React, { useState, useEffect } from 'react';
2 import Carousel from '../Components/Carousel';
3 import NavBar from '../Components/NavBar';
4 import Footer from '../Components/Footer';
5 import ComicsHome from '../Components/ComicsHome';
6 import CharactersHome from '../Components/CharactersHome';
7
8 const Home: React.FC = () => {
9   // State to manage loading status
10   const [loading, setLoading] = useState(true);
11
12   // Retrieve the user's access token from local storage
13   const token = localStorage.getItem('accessToken');
14
15   useEffect(() => {
16     // Use the useEffect hook to perform actions after the component renders
17     // Perform a request to fetch data for the authenticated user
18     fetch('https://grecohen-server.eddi.cloud/api/home-comics', {
19       method: 'GET',
20       headers: {
21         Authorization: `Bearer ${token}`,
22       },
23     })
24       .then((response) => response.json())
25       .then((data) => {
26         // Update resultAPI with received data
27         setResultAPI(data);
28         // Signal that loading is complete
29         setLoading(false);
30       })
31       .catch((err) => {
32         console.error(err);
33         // Signal that loading is complete in case of an error
34         setLoading(false);
35       });
36   }, [token]);
37
38   // You can now use resultAPI in your JSX
39   return (
40     <div>
41       <div className="flex flex-col bg-gray-800 min-h-screen">
42         <div className="">
43           { /* Render the navigation bar with authentication status */ }
44           <NavBar isAuthenticated={false} />
45         </div>
46         <div><Carousel /></div>
47         <section className="mx-40 lg:mx-56">
48           <h1 className="text-white font-bold mb-1 text-xl inline-block p-2 bg-red-700 my-5">
49             { /* Display a title for new content */ }
50             WHAT'S NEW
51           </h1>
52           { /* Render the component for displaying comics */ }
53           <ComicsHome />
54           <h1 className="text-white font-bold mb-5 text-xl inline-block p-2 bg-red-700 my-5">
55             { /* Display a title for characters */ }
56             CHARACTERS
57           </h1>
58           { /* Render the component for displaying characters */ }
59           <CharactersHome />
60           <div className="flex justify-center text-center m-4 md:my-16 text-white"></div>
61         </section>
62         <section className="">
63           { /* Render the footer component */ }
64           <Footer />
65         </section>
66       </div>
67     </div>
68   );
69 };
70
71 export default Home;

```

CP 3 Développer une interface utilisateur web dynamique

Dans la NavBar, un menu burger a été mis en place pour gérer la navigation sur les appareils mobiles. Pour ce faire, j'ai utilisé une variable d'état appelée "isMobileMenuOpen" qui contrôle la visibilité du menu burger. Cette variable débute à false, ce qui signifie que le menu burger est fermé par défaut.

Lorsque l'utilisateur clique sur l'icône du menu burger (les trois lignes) dans la barre de navigation, cela déclenche l'événement onClick, appelant la fonction "toggleMobileMenu".

La fonction "toggleMobileMenu" est responsable de changer la valeur de "isMobileMenuOpen" à chaque clic. Ainsi, chaque fois que l'utilisateur clique sur le bouton, le menu burger passe de l'état ouvert à l'état fermé, ou vice-versa.

Sous le bouton du menu burger, se trouve une liste de navigation conçue pour les écrans de bureau. Cette liste contient des éléments de menu tels que "Comics", "Personnages", "Se connecter", etc.

Pour garantir une expérience utilisateur cohérente, une vérification conditionnelle est effectuée. Si "isMobileMenuOpen" est vrai, ce qui signifie que le menu burger est ouvert, alors une deuxième liste de navigation est affichée spécialement pour les appareils mobiles.

Les éléments de menu dans le menu burger pour les écrans de bureau et les appareils mobiles sont presque identiques en termes de contenu, mais leur disposition diffère. Sur les appareils mobiles, ces éléments sont affichés sous forme de liens dans une liste verticale, offrant ainsi une meilleure expérience de navigation sur les petits écrans.

```
1 import React, { useContext, useState } from 'react';
2 import Logo from '../assets/Logo.png';
3 import { UserOutlined } from '@ant-design/icons';
4 import useAuth from '../hooks/useAuth';
5 import { useNavigate, Link } from 'react-router-dom';
6 import AuthContext from '../context/AuthProvider';
7
8 type NavbarProps = {
9   isAuthenticated: boolean;
10 }
11
12 const Navbar: React.FC<NavbarProps> = ({ isAuthenticated }) => {
13   const { auth, setAuth } = useAuth();
14   const navigate = useNavigate();
15
16   const onLogout = () => {
17     localStorage.removeItem('accessToken');
18     const authData = {
19       auth: false,
20       email: '',
21       roles: [],
22       accessToken: '',
23     };
24     setAuth(authData);
25     navigate('/');
26   };
27
28   isAuthenticated = auth.auth;
29
30   const { auth: authContext } = useContext(AuthContext);
31
32   const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);
33
34   const toggleMobileMenu = () => {
35     setIsMobileMenuOpen(!isMobileMenuOpen);
36   };
37
```



```

116 {isMobileMenuOpen && (
117   <div>
118     <ul className="bg-yellow-500 space-y-2">
119       <li>
120         <Link to="/comics" className="block py-2">Comics</Link>
121       </li>
122       <li>
123         <Link to="/Personnages" className="block py-2">Personnages</Link>
124       </li>
125       {isAuthenticated ? (
126         <li>
127           <Link to="/ownlist" className="block py-2">Je possède</Link>
128         </li>
129         <li>
130           <Link to="/wishlist" className="block py-2">Je recherche</Link>
131         </li>
132         <li>
133           <button onClick={onLogout} className="block py-2">Se déconnecter</button>
134         </li>
135         <li>
136           <Link to="https://grecoben-server.eddi.cloud/" className="block py-2">Back Office</Link>
137         </li>
138       </li>
139     ) : (
140       <li>
141         <Link to="/register" className="block py-2">Inscription</Link>
142       </li>
143       <li>
144         <Link to="/login" className="block py-2">Se connecter</Link>
145       </li>
146     </li>
147   </li>
148 </ul>
149 </div>
150 {isAuthenticated && (
151   <li>
152     <span className="block py-2" style={{ fontStyle: 'italic', fontWeight: 'lighter' }}>
153       Bienvenue <span style={{ fontStyle: 'italic', fontWeight: 'lighter' }}>{authContext.email}</span>
154     </span>
155   </li>
156 </li>
157 {isAuthenticated && (
158   <li>
159     <UserOutlined style={{ fontSize: '18px' }} />
160   </li>
161 </li>
162 </ul>
163 </div>
164 </nav>
165 </section>
166 </div>
167 </div>
168 </div>

```

Pour la sécurité côté front, des expressions régulières (regex) sont utilisées pour valider les entrées de l'utilisateur dans le composant Register. Les expressions régulières sont stockées dans les constantes "USER_REGEX", "PWD_REGEX," et "EMAIL_REGEX", et elles définissent les règles auxquelles les données d'inscription doivent se conformer. Voici comment elles sont utilisées dans le code :

"USER_REGEX" (Expression régulière pour le nom d'utilisateur) :

Cette expression régulière est utilisée pour valider le nom d'utilisateur.

Elle exige que le nom d'utilisateur commence par une lettre, suivi de lettres, de chiffres, de traits de soulignement (_) ou de traits d'union (-).

La longueur du nom d'utilisateur doit être comprise entre 4 et 24 caractères.

Si le nom d'utilisateur ne respecte pas cette expression régulière, un message d'erreur sera affiché à l'utilisateur.

"PWD_REGEX": (Expression régulière pour le mot de passe) : Cette expression régulière est utilisée pour valider le mot de passe de l'utilisateur. Elle exige que le mot de passe ait une longueur comprise entre 8 et 24 caractères. Le mot de passe doit contenir au moins une lettre minuscule, une lettre majuscule, un chiffre et un caractère spécial parmi !, @, #, \$, %. Si le mot de passe ne respecte pas cette expression régulière, un message d'erreur sera affiché à l'utilisateur.

"EMAIL_REGEX": (Expression régulière pour l'adresse e-mail) :

Cette expression régulière est utilisée pour valider l'adresse e-mail de l'utilisateur.

Elle vérifie que l'adresse e-mail suit un format valide.

Si l'adresse e-mail ne respecte pas cette expression régulière, un message d'erreur sera affiché à l'utilisateur.

Ces expressions régulières sont utilisées dans les fonctions "useEffect" du composant pour vérifier la validité des données fournies par l'utilisateur. Si les données ne respectent pas les expressions régulières, des messages d'erreur sont affichés, et l'utilisateur ne peut pas s'inscrire tant que les données ne sont pas correctes.

Par exemple, dans la fonction "handleSubmit", vous pouvez voir que les expressions régulières sont utilisées pour valider à nouveau le nom d'utilisateur (v1) et le mot de passe (v2). Si l'une de ces validations échoue ou si l'adresse e-mail n'est pas valide, le message d'erreur "Invalid Entry" est affiché à l'utilisateur, l'empêchant de s'inscrire.

```

1  import React, { useRef, useState, useEffect } from "react";
2  import { faCheck, faTimes, faInfoCircle } from "@fortawesome/free-solid-svg-icons";
3  import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
4  import { useNavigate, Link } from 'react-router-dom';
5
6  const USER_REGEX = /^[A-z][A-z0-9-_]{3,23}$/;
7  const PWD_REGEX = /^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#%]).{8,24}$/;
8  const EMAIL_REGEX = /^[A-Za-z0-9+_.-]+@(.+)$/;
9  const REGISTER_URL = 'https://grecohen-server.eddi.cloud/api/register';
10
11  const Register = () => {
12    const userRef = useRef<HTMLInputElement | null>(null);
13    const errRef = useRef<HTMLParagraphElement | null>(null);
14
15    const [username, setUser] = useState('');
16    const [validName, setValidName] = useState(false);
17    const [userFocus, setUserFocus] = useState(false);
18
19    const [password, setPwd] = useState('');
20    const [validPwd, setValidPwd] = useState(false);
21    const [pwdFocus, setPwdFocus] = useState(false);
22
23    const [matchPwd, setMatchPwd] = useState('');
24    const [validMatch, setValidMatch] = useState(false);
25    const [matchFocus, setMatchFocus] = useState(false);
26
27    const [email, setEmail] = useState('');
28    const [validEmail, setValidEmail] = useState(false);
29
30    const [firstName, setFirstName] = useState('');
31    const [lastName, setLastName] = useState('');
32

```

CP 5 Créer une base de données

Dans la phase de développement backend, nous avons entrepris la création de la base de données de notre application en utilisant PHP et le framework Symfony. Cette étape a impliqué la conception de plusieurs éléments essentiels, dont le Modèle Logique de Données (MLD), le Modèle Conceptuel de Données (MCD), et la mise en place d'un Dictionnaire de Données exhaustif. Ces ressources nous ont permis de définir avec précision la structure de notre base de données, y compris la gestion des tables et des collections, ainsi que les relations complexes entre elles.

Pour créer la base de données, j'ai utilisé la méthode Merise qui consiste à d'abord identifier les besoins du site et de concevoir ensuite le modèle conceptuel, le modèle logique et le modèle physique de données.

Pour cette plateforme j'ai d'abord définis les entités dont j'aurais besoin, qui seront par la suite les tables dans la base de données, ainsi que leurs propriétés : -

- Comics (title, poster, resume, rarity, released at)
- Characters (alias, name, description, released at)
- User (firstname, lastname, pseudo, email, password, role)
- Author (firstname, lastname)
- User collection (user id)
- Wish collection (user id)

Ensuite j'établis les relations, les cardinalités, les quantités minimum et maximum d'occurrences entre chaque table et ce dans les deux sens :

La relation entre comics et characters : -

- Un comics peut avoir au minimum 0 characters et au maximum plusieurs characters => 0,N
 - Un characters peut appartenir à au minimum 0 et au maximum plusieurs comics => 0,N
- ici je suis sur une relation Many to Many.

La relation entre comics et user collection :

- Un comics peut aller dans au minimum 0 et au maximum plusieurs user collection => 0,N
 - Une user collection peut contenir au minimum 0 et au maximum plusieurs comics => 0,N
- Ici je suis sur une relation Many to Many.

La relation entre comics et wish collection :

- Un comics peut aller dans au minimum 0 et au maximum plusieurs wish collection => 0,N
 - Une wish collection peut contenir au minimum 0 et au maximum plusieurs comics => 0,N
- Ici je suis sur une relation Many to Many.

La relation entre user et user collection :

- Un user peut avoir au minimum 1 et au maximum une user collection => 1,1
- Une user collection peut appartenir au minimum et au maximum à un user => 1,1

Je suis ici sur une relation One to One.

La relation entre user et wish collection :

- Un user peut avoir au minimum 1 et au maximum une wish collection => 1,1
- Une wish collection peut appartenir au minimum et au maximum à un user => 1,1

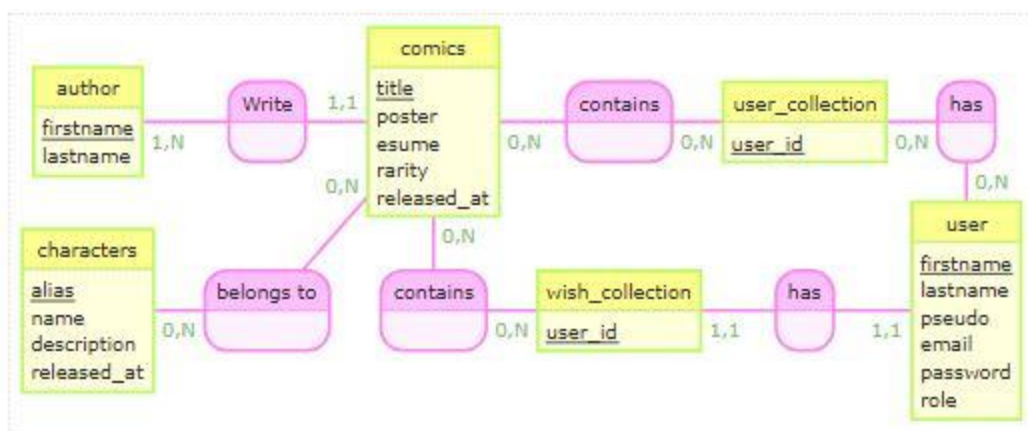
Je suis ici sur une relation One to One.

La relation entre author et comics :

- Un author peut écrire au minimum 0 et au maximum plusieurs comics => 0,N
- Un comics peut être écrit par au minimum et au maximum un author => 1,1

Je suis ici sur une relation One to Many

Une fois cet exercice terminé, je me suis servi de <https://www.mocodo.net> afin de réaliser le MCD



CP 6 Développer les composants d'accès aux données

Nous avons entrepris le développement des composants d'accès aux données. Après avoir soigneusement configuré notre fichier .env pour les paramètres de base de données, nous avons utilisé le puissant ORM (Object-Relational Mapping) Doctrine, intégré à Symfony, pour simplifier la gestion de notre base de données.

```

16
17 ###> symfony/framework-bundle ###
18 APP_ENV=dev
19 APP_SECRET=1efbbfe19151c593af94fde9ac2678b4
20 ###< symfony/framework-bundle ###
21
22 ###> doctrine/doctrine-bundle ###
23 # Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-url
24 # IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
25 #
26 # DATABASE URL="sqlite://%kernel.project_dir%/var/data.db"
27 DATABASE_URL="mysql://explorateur:Ereul9Aeng@127.0.0.1:3306/ocomics?serverVersion=10.3.38-MariaDB&charset=utf8mb4"
28 # DATABASE URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8"
29 ###< doctrine/doctrine-bundle ###
30
31 ###> lexik/jwt-authentication-bundle ###
32 JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
33 JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
34
35 JWT_PASSPHRASE=8ae04946cfef32188e7c8f1ff2192a77ae9a651de8a2e66101c9d1a6059f8c0f
36 ###< lexik/jwt-authentication-bundle ###
37
38 ###> nelmio/cors-bundle ###
39 CORS_ALLOW_ORIGIN='^https?:/(localhost|127\.\0\.\0\1)(:[0-9]+)?$'
40 ###< nelmio/cors-bundle ###
41
42 ###> symfony/mailer ###
43 # MAILER_DSN=null://null
44 ###< symfony/mailer ###
45
46 ###> symfony/mailjet-mailer ###
47 # MAILER_DSN=mailjet+api://PUBLIC_KEY:PRIVATE_KEY@api.mailjet.com
48 # MAILER_DSN=mailjet+smtp://PUBLIC_KEY:PRIVATE_KEY@in-v3.mailjet.com
49 ###< symfony/mailjet-mailer ###
50
51 ### Marvel API
52 MARVEL_PUBLIC_KEY=98ba40964d53c077c5bbf572e59d62ab
53 MARVEL_PRIVATE_KEY=d5933e5770e5cc5b2156240ee8adfaed002e3dbe
54
  
```

Ensuite nous avons configuré notre fichier “gitignore” afin de sécuriser le fichier “.env” et ne pas le sauvegarder sur Github et que personne n’accède au mot de passe de la base de données.

```

◆ .gitignore
1
2  ###> symfony/framework-bundle ###
3  /.env
4  /.env.local
5  /.env.local.php
6  /.env.*.local
7  /config/secrets/prod/prod.decrypt.private.php
8  /public/bundles/
9  /var/
10 /vendor/
11  ###< symfony/framework-bundle ###
12
13  ###> lexik/jwt-authentication-bundle ###
14  /config/jwt/*.pem
15  ###< lexik/jwt-authentication-bundle ###
16
17  ###> phpunit/phpunit ###
18  /phpunit.xml
19  .phpunit.result.cache
20  ###< phpunit/phpunit ###
21
22  ###> symfony/phpunit-bridge ###
23  .phpunit.result.cache
24  /phpunit.xml
25  ###< symfony/phpunit-bridge ###
26

```

Nous avons commencé par mettre en place les entités, en définissant leurs relations complexes, tout en utilisant les commandes pour gérer ces entités de manière cohérente. Les repositories ont joué un rôle central, permettant aux contrôleurs d’accéder efficacement aux données.

Pour créer les entités, propriétés et relations également avec Doctrine j’ utilise la commande suivante : `php bin/console make :entity`

Cette commande nous servira à établir les relations par la suite.

```

○ student@teleporter: /var/www/html/projet-18-o-comicverse-back$ php bin/console make:entity

Class name of the entity to create or update (e.g. GentleKangaroo):
> Comics

Your entity already exists! So let's add some new fields!

New property name (press <return> to stop adding fields):
> comics

Field type (enter ? to see all types) [string]:
> relation

What class should this entity be related to?:
> Characters

What type of relationship is this?
-----
Type           Description
-----
ManyToOne      Each Comics relates to (has) one Characters.
                Each Characters can relate to (can have) many Comics objects

OneToMany      Each Comics can relate to (can have) many Characters objects.
                Each Characters relates to (has) one Comics

ManyToMany     Each Comics can relate to (can have) many Characters objects.
                Each Characters can also relate to (can also have) many Comics objects

OneToOne       Each Comics relates to (has) exactly one Characters.
                Each Characters also relates to (has) exactly one Comics.
-----

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
> 

```

Une fois mes entités créés, je crée la base de données via le terminal de commande avec la commande suivante : **php bin/console doctrine:database:create**

Ensuite nous allons créer une nouvelle migration des données en utilisant la commande **php bin/console make:migration**.

Lorsque j'exécute cette commande, Symfony génère un nouveau fichier de migration dans le répertoire des migrations du projet. Ce fichier contient les instructions pour effectuer les modifications de base de données spécifiques telles que la création de tables, l'ajout de colonnes, etc.

Une fois le fichier de migration créé, et pour finir la migration des données vers la base de données, j'utilise la commande **php bin/console doctrine:migrations:migrate**.

Lorsque j'exécute cette commande, Symfony parcourt les fichiers de migration qui n'ont pas encore été exécutés et les exécute dans l'ordre où ils ont été créés. Chaque fichier de migration contient des instructions pour effectuer des modifications spécifiques de la base de données. En exécutant cette commande, Symfony applique ces modifications à la base de données.

Afin de tester la bonne mise en place de notre base de données, je vais lui injecter des requêtes SQL directement depuis PhpMyAdmin et observer son comportement et les résultats:

```
SELECT author.firstname, author.lastname, comics.title
```

```
FROM author
```

```
INNER JOIN comics ON author.id = comics.author_id;
```

Cette requête combine les tables "author" et "comics" en utilisant un INNER JOIN pour obtenir les noms des auteurs de comics et les titres des comics qu'ils ont créés.

✓ Affichage des lignes 0 - 1 (total de 2, traitement en 0.0008 seconde(s).)

```
SELECT author.firstname, author.lastname, comics.title FROM author INNER JOIN comics ON author.id = comics.author_id
```

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: | Trier sur l'index:

+ Options

firstname	lastname	title
Nicole	Fischer	X-Men: Days of Future Past (Trade Paperback)
Jeanne	Auger	TEST COMICS

```
SELECT comics.*, author.firstname AS firstname_author
```

```
FROM comics
```

```
INNER JOIN author ON comics.author_id = author.id
```

La requête récupère toutes les données des comics et le prénom de l'auteur du comics uniquement sur les comics qui ont un auteur. La clause INNER JOIN est utilisée pour combiner les données de la table "comics" avec les données de la table "author" en fonction de la clé étrangère "author_id." Cela signifie que seuls les comics qui ont un auteur correspondant dans la table "author" seront inclus dans le résultat de la requête. Les comics sans auteur correspondant ne seront pas inclus.

✓ Affichage des lignes 0 - 1 (total de 2, traitement en 0.0001 seconde(s).)

```
SELECT comics.*, author.firstname AS firstname_author FROM comics INNER JOIN author ON comics.author_id = author.id
```

☐ Profilage [Éditer en ligne] [Éditer]

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: | Trier sur l'index: Aucun(e)

+ Options

id	author_id	title	poster	released_at (UTC+0 timestamp, immutable)	synopsis	rarity	firstname_author
5	10	X-Men: Days of Future Past (Trade Paperback)	http://i.annihil.us/u/prod/marvel/i/mg/9/d0/58b5cf...	2004-09-25 00:00:00	"Re-live the legendary first journey into the dyst...	3	Nicole
75	12	TEST COMICS	http://www.test.com	2001-09-03 21:38:00	Tempore facere ipsa	1	Jeanne

CP 7 Développer la partie back-end d'une application web ou web mobile

Dans la phase de développement de la partie back-end de notre application web, nous avons réalisé plusieurs tâches cruciales pour garantir le bon fonctionnement et la sécurité de l'application. Nous avons mis en place un back office complet, offrant un ensemble complet de fonctionnalités CRUD pour gérer les données relatives aux comics.

J'implémente un système CRUD (Create, Read, Update, Delete) pour la gestion de l'entité "Comics". J'utilise les fonctions "persist" pour ajouter de nouvelles instances d'entités Comics et "flush" pour synchroniser les opérations avec la base de données. Les vues sont générées avec Twig et rendues avec la fonction "render" pour afficher les données stockées dans la base de données.

Pour la création et la modification d'entités "Comics", le formulaire est créé à l'aide de la classe ComicsType et est lié à l'entité "Comics". Il est utilisé pour la saisie des données nécessaires à la création ou à la mise à jour d'un comic.

Ensuite, les méthodes "createForm" et "handleRequest" sont employées pour manipuler ce formulaire. La méthode createForm génère un formulaire avec les champs appropriés en utilisant ComicsType, tandis que "handleRequest" gère la soumission du formulaire. Si le formulaire est soumis et valide, les données sont sauvegardées dans la base de données en utilisant "persist" et "flush" comme expliqué précédemment.


```

1  <?php
2
3  namespace App\Controller\Admin;
4
5  use App\Entity\Comics;
6  use App\Form\ComicsType;
7  use App\Repository\ComicsRepository;
8  use Doctrine\ORM\EntityManagerInterface;
9  use Symfony\Component\HttpFoundation\Request;
10 use Symfony\Component\HttpFoundation\Response;
11 use Symfony\Component\Routing\Annotation\Route;
12 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
13
14 class ComicsController extends AbstractController
15 {
16     /**
17      * @Route("/", name="app_admin_comics_list")
18      * Homepage, display the list of all the comicss
19      */
20     public function index(ComicsRepository $comicsRepository, Request $request): Response
21     {
22         $allComics = $comicsRepository->findAllOrderByTitleSearch($request->get("search"));
23         return $this->render('admin/comics/list.html.twig', [
24             'allComics' => $allComics
25         ]);
26     }
27
28     /**
29      * @Route("/admin/comics/{id}", name="app_admin_comics_show", requirements={"id"="\d+"})
30      * Homepage, display the selected comics
31      */
32     public function show(Comics $comics): Response
33     {
34         return $this->render('admin/comics/show.html.twig', [
35             'comics' => $comics,
36         ]);
37     }
38 }


```

```

39  /**
40   * @Route("/admin/comics/add", name="app_admin_comics_add")
41   * Display the form to add a new comics
42   */
43  public function add(Request $request, EntityManagerInterface $entityManager): Response
44  {
45      $comics = new Comics();
46      $form = $this->createForm(ComicsType::class, $comics);
47      $form->handleRequest($request);
48
49      if ($form->isSubmitted() && $form->isValid()) {
50          $entityManager->persist($comics);
51          $entityManager->flush();
52          $this->addFlash("success", "Comics ajouté avec succès!");
53          return $this->redirectToRoute('app_admin_comics_list');
54      } elseif ($form->isSubmitted()) {
55          $this->addFlash("danger", "L'ajout du comics a échoué!");
56      }
57
58      return $this->renderForm("admin/comics/form.html.twig", [
59          "form" => $form
60      ]);
61  }
62
63  /**
64   * @Route("/admin/comics/edit/{id}", name="app_admin_comics_edit", requirements={"id"="\d+"})
65   * Display the form to edit a new comics
66   */
67  public function edit(Comics $comics, Request $request, EntityManagerInterface $entityManager): Response
68  {
69      $form = $this->createForm(ComicsType::class, $comics);
70      $form->handleRequest($request);
71      if ($form->isSubmitted() && $form->isValid()) {
72
73          $entityManager->flush();
74          $entityManager->persist($comics);
75
76          $this->addFlash("success", "Le comics a bien été édité.");
77
78          return $this->redirectToRoute('app_admin_comics_list');
79      }
80
81      return $this->renderForm("admin/comics/form.html.twig", [
82          "form" => $form
83      ]);
84  }
85
86  /**
87   * @Route("/admin/comics/delete/{id}", name="app_admin_comics_delete", requirements={"id"="\d+"})
88   */
89  public function delete(Comics $comics, ComicsRepository $comicsRepository): Response
90  {
91      $comicsRepository->remove($comics, true);
92
93      $this->addFlash("success", "Le comics a bien été supprimé.");
94
95      return $this->redirectToRoute('app_admin_comics_list', [], Response::HTTP_SEE_OTHER);
96  }

```

Ajouter un comics

Titre	Poster	Date de sortie	Auteur	Action
Civil War (Hardcover)	http://i.annihil.us/u/prod/marvel/i/mg/8/c0/51dda501724ed.jpg	03/03/2014		 
Gwen Stacy (2020) #3	http://i.annihil.us/u/prod/marvel/i/mg/a/10/5e9e076819aec.jpg	02/08/2021		 
IMMORTAL X-MEN BY KIERON GILLEN VOL. 3 TPB (Trade Paperback)	http://i.annihil.us/u/prod/marvel/i/mg/c/b0/64e3bed77b17e.jpg	02/08/2017		 
Incredible Hulks: Dark Son (2010)	http://i.annihil.us/u/prod/marvel/i/mg/b/70/4cbda1eb06127.jpg	04/05/2023		 
Kabuki Reflections Vol. 1 (Hardcover)	http://i.annihil.us/u/prod/marvel/i/mg/e/e0/4bac3ad5d17c7.jpg	17/05/2016		 
Marvel Age Spider-Man Vol. 2: Everyday Hero (Digest)	http://i.annihil.us/u/prod/marvel/i/mg/9/20/4bc665483c3aa.jpg	23/02/2019		 
MARVEL MASTERWORKS: THE UNCANNY X-MEN VOL. 3 HC (Trade Paperback)	http://i.annihil.us/u/prod/marvel/i/mg/9/10/4bb3c93c1725d.jpg	27/05/2014		 
Marvel Previews (2017)	http://i.annihil.us/u/prod/marvel/i/mg/c/80/5e3d7536c8ada.jpg	27/09/2018		 
Nebula (2020) #3	http://i.annihil.us/u/prod/marvel/i/mg/6/e0/5e86538d453bc.jpg	29/01/2023		 
Official Handbook of the Marvel Universe (2004) #12 (SPIDER-MAN)	http://i.annihil.us/u/prod/marvel/i/mg/b/40/4bc64020a4ccc.jpg	31/05/2019		 
Penance: Relentless (2008)	http://i.annihil.us/u/prod/marvel/i/mg/9/90/4bb860a46f58d.jpg	13/02/2015		 
PUNISHER VS. THE MARVEL UNIVERSE TPB (Trade Paperback)	http://i.annihil.us/u/prod/marvel/i/mg/c/80/56be3df79d1c0.jpg	20/11/2018		 

Ajouter un nouveau comcis

Titre du comics

Superman n°42

URL de l'image de couverture

l'URL de l'image qui sera affiché

synopsis

Après avoir arrêté le bouffon vert, ...

Année de sortie

jj/mm/aaaa --:--



Rareté du comics

Collector

Auteur du comics

Collet

Personnages

- ☐ A-Bomb (HAS)
- ☐ Abomination (Emil Blonsky)
- ☐ Adam Warlock
- ☐ Alex Wilder
- ☐ Attuma
- ☐ Avengers

Civil War (Hardcover)

Titre	Civil War (Hardcover)
Date de publication	2014-03-03 02:39:25
Synopsis	The landscape of the Marvel Universe is changing, and it's time to choose: Whose side are you on? A conflict has been brewing from more than a year, threatening to pit friend against friend, brother against brother - and all it will take is a single misstep to cost thousands their lives and ignite the fuse! As the war claims its first victims, no one is safe as teams, friendships and families begin to fall apart. The crossover that rewrites the rules, Civil War stars Spider-Man, the New Avengers, the Fantastic Four, the X-Men and the entirety of the Marvel pantheon! Collecting CIVIL WAR #1-7, MARVEL SPOTLIGHT: CIVIL WAR and CIVIL WAR SCRIPT BOOK. Rated T+ ...\$39.99 ISBN: 978-0-7851-2178-7 Trim size: oversized
Image	http://i.annihil.us/u/prod/marvel/i/mg/8/c0/51dda501724ed.jpg
Type	

[Retour à la liste des comics](#)
[Editer le comics](#)
[Supprimer le comics](#)

RÉALISATIONS TECHNIQUES DU PROJET

I. Front-end

A. Main.tsx

Le fichier "main.tsx" est le point d'entrée de l'application "React". Il accomplit diverses tâches essentielles pour l'initialisation de l'application.

Tout d'abord, il commence par importer les dépendances nécessaires, notamment "React" et "ReactDOM" pour créer des éléments "React" et les afficher dans le DOM. Il importe également le composant principal "App" depuis le fichier "App.tsx", les fichiers de style tels que "index.css" pour la mise en forme de l'application, le composant "AuthProvider" utilisé pour la gestion de l'authentification, ainsi que les composants liés à la navigation dans l'application, comme "HashRouter," "Routes," et "Route" depuis la bibliothèque "react-router-dom."

Ensuite, le fichier utilise la fonction "ReactDOM.createRoot" pour définir la racine de rendu de l'application. Cette racine est associée à un élément HTML dans le DOM identifié par l'ID "root." Cela signifie que l'application React sera affichée à l'intérieur de cet élément DOM spécifique.

Le reste du fichier consiste en la définition de la structure de l'application React. L'application est encapsulée dans un composant "React.StrictMode", ce qui permet d'effectuer des vérifications supplémentaires en mode de développement pour détecter d'éventuels problèmes.

Le composant "HashRouter" gère la navigation au sein de l'application en utilisant des ancres (hashes) dans l'URL. Il est utilisé pour configurer les différentes vues de l'application.

Le composant "AuthProvider" gère l'authentification de l'utilisateur.

Le composant "Routes" contient la configuration des différentes routes de l'application.

Enfin, le composant "Route" définit une route spécifique pour l'application. Dans ce cas, il est configuré pour correspondre à toutes les URL en utilisant "/"*"" comme chemin et associe cette route au composant "App".

```

1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import App from './App';
4  import './index.css';
5  import { AuthProvider } from './context/AuthProvider';
6  import { HashRouter, Routes, Route } from 'react-router-dom';
7
8
9  ReactDOM.createRoot(document.getElementById('root')).render(
10     <React.StrictMode>
11       <HashRouter>
12         <AuthProvider>
13           <Routes>
14             <Route path="/"*"" element={<App />} />
15           </Routes>
16         </AuthProvider>
17       </HashRouter>
18     </React.StrictMode>,
19   );
20

```

B. Index.html

L'élément avec l'ID "root" dans le fichier "index.html" sert de point d'ancrage pour le rendu de l'application React. C'est un élément HTML où le contenu généré par React sera inséré.


```

1  <!doctype html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <link rel="icon" type="image/svg+xml" href="/src/assets/Logo.png" />
6      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7      <title>O'Comics</title>
8    </head>
9    <body>
10     <div id="root"></div>
11     <script type="module" src="/src/main.tsx"></script>
12   </body>
13 </html>
14 |

```

C. App.tsx

Le fichier App.tsx sert de colonne vertébrale à notre application React. Ce fichier gère le routage, la gestion de l'authentification et la structure générale de notre application. Au début du fichier, nous importons différents modules de diverses parties de notre projet. Ces modules comprennent des composants pour nos pages, l'authentification, la mise en page et d'autres fonctionnalités clés.

Le point central de ce fichier est le composant fonctionnel App, qui est de type React.FC (Composant Fonctionnel) et ne reçoit pas de props. Dans ce composant App, nous encapsulons l'ensemble de notre application avec le composant AuthProvider. Cela signifie que nous utilisons un contexte d'authentification pour gérer l'état d'authentification dans toute l'application.

À l'intérieur du AuthProvider, nous déclarons nos itinéraires de navigation à l'aide du composant "Routes". Ces itinéraires de navigation sont gérés par React Router, ce qui signifie que différentes URL de notre application correspondent à des composants spécifiques.

Nous avons à la fois des itinéraires publics (accessibles sans authentification) et des itinéraires protégés (accessibles uniquement aux utilisateurs authentifiés). Les itinéraires publics comprennent des pages telles que la page d'accueil, la liste des personnages, la page de connexion, la page d'inscription, et bien d'autres.

Les itinéraires protégés comprennent des pages comme la "WishPage", la "OwnPage", l'administration et le back office.

Nous utilisons des composants "Route" pour définir le rendu des composants associés à chaque URL. Par exemple, lorsque l'URL correspond à "/comics", le composant "ComicsPage" est rendu.

Les itinéraires protégés sont enveloppés dans un composant "Route" parent avec un élément "RequireAuth". Cela signifie que ces itinéraires nécessitent une authentification pour y accéder. Nous avons des pages comme "WishPage", "OwnPage", "Administration", et "BackOffice" qui relèvent de cette catégorie.

```

1  import React from 'react';
2  import { Routes, Route } from 'react-router-dom';
3  import Home from './Pages/Home';
4  import CharactersList from './Pages/CharactersList';
5  import ComicsPage from './Pages/ComicsPage';
6  import Register from './Components/Register';
7  import Login from './Components/Login';
8  import Layout from './Components/Layout';
9  import OwnPage from './Pages/OwnPage';
10 import Unauthorized from './Components/Unauthorized';
11 import RequireAuth from './Components/RequireAuth';
12 import Administration from './Components/Administration';
13 import CharacterComicsPage from './Pages/CharacterComicsPage';
14 import { AuthProvider } from './context/AuthProvider';
15 import BackOffice from './Pages/BackOffice';
16 import WishPage from './Pages/WishPage';
17
18
19 const App: React.FC = () => {
20
21
22   return (
23     <AuthProvider>
24       <Routes>
25
26         <Route path="/" element={<Layout />} >
27           { /*public routes*/ }
28           <Route path="/" element={<Home />} />
29           <Route path="/comics" element={<ComicsPage />} />
30           <Route path="/Personnages" element={<CharactersList />} />
31           <Route path="/register" element={<Register />} />
32           <Route path="/login" element={<Login />} />
33           <Route path="/unauthorized" element={<Unauthorized />} />
34           <Route path="/character-comics/:characterId" element={<CharacterComicsPage />} />
35           { /*protected routes*/ }
36           <Route element={<RequireAuth/>} >
37             <Route path="/wishlist" element={<WishPage />} />
38             <Route path="/ownlist" element={<OwnPage />} />
39             <Route path="/administration" element={<Administration />} />
40             <Route path="/backoffice" element={<BackOffice />} />
41           </Route>
42         </Route>
43
44       </Routes>
45     </AuthProvider>
46   );
47 };
48
49 export default App;

```


D. Home.tsx

Ce code représente un composant React nommé Home, qui sert de page d'accueil dans notre application. Il est utilisé pour afficher le contenu de la page d'accueil de notre application.

Nous commençons par importer différents modules et composants nécessaires. Ces importations incluent des éléments React tels que `"useState"`, `"useEffect"`, ainsi que plusieurs composants personnalisés : `"Carousel"`, `"NavBar"`, `"Footer"`, `"ComicsHome"`, et `"CharactersHome"`. Ces composants sont importés depuis leurs emplacements respectifs dans le projet.

Dans le composant Home, nous déclarons un état local appelé `loading` en utilisant le hook `"useState"`. Initialement, nous le définissons sur `true`, indiquant que la page est en cours de chargement.

Nous récupérons également un jeton d'accès (token) à partir du stockage local en utilisant `localStorage.getItem('accessToken')`.

Nous utilisons le hook `"useEffect"` pour effectuer des actions asynchrones. Dans cet effet, nous effectuons une requête HTTP en utilisant `fetch` vers l'URL `'https://grecoben-server.eddi.cloud/api/home-comics'` pour obtenir des données. Nous incluons le jeton d'accès dans les en-têtes de la requête en utilisant l'authentification Bearer.

Nous gérons la réponse de la requête à l'aide de `.then` et `.catch`. Si la requête réussit, nous mettons à jour un état `"resultAPI"` (qui n'est pas défini dans le code que nous avons partagé) avec les données reçues, puis nous indiquons que le chargement est terminé en définissant l'état `loading` sur `false`. En cas d'erreur, nous affichons l'erreur dans la console et indiquons également que le chargement est terminé.

Enfin, nous retournons le contenu JSX de la page d'accueil. La page est structurée en plusieurs sections, comprenant une barre de navigation `"NavBar"`, un carrousel `"Carousel"`, des sections pour afficher les nouveautés `"ComicsHome"` et les personnages `"CharactersHome"`, ainsi qu'un pied de page `"Footer"`. Nous appliquons des styles CSS en utilisant des classes pour définir l'apparence de la page.

```

1 import React, { useState, useEffect } from 'react';
2 import Carousel from '../Components/Carousel';
3 import NavBar from '../Components/NavBar';
4 import Footer from '../Components/Footer';
5 import ComicsHome from '../Components/ComicsHome';
6 import CharactersHome from '../Components/CharactersHome';
7
8 const Home: React.FC = () => {
9   // State to manage loading status
10   const [loading, setLoading] = useState(true);
11
12   // Retrieve the user's access token from local storage
13   const token = localStorage.getItem('accessToken');
14
15   useEffect(() => {
16     // Use the useEffect hook to perform actions after the component renders
17     // Perform a request to fetch data for the authenticated user
18     fetch('https://grecoben-server.eddi.cloud/api/home-comics', {
19       method: 'GET',
20       headers: {
21         Authorization: `Bearer ${token}`,
22       },
23     })
24       .then((response) => response.json())
25       .then((data) => {
26         // Update resultAPI with received data
27         setResultAPI(data);
28         // Signal that loading is complete
29         setLoading(false);
30       })
31       .catch((err) => {
32         console.error(err);
33         // Signal that loading is complete in case of an error
34         setLoading(false);
35       });
36   }, [token]);
37
38   // You can now use resultAPI in your JSX
39   return (
40     <div>
41       <div className="flex flex-col bg-gray-800 min-h-screen">
42         <div className="">
43           {/* Render the navigation bar with authentication status */}
44           <NavBar isAuthenticated={false} />
45         </div>
46         <div><Carousel /></div>
47         <section className="mx-40 lg:mx-56">
48           <h1 className="text-white font-bold mb-1 text-xl inline-block p-2 bg-red-700 my-5">
49             {/* Display a title for new content */}
50             WHAT'S NEW
51           </h1>
52           {/* Render the component for displaying comics */}
53           <ComicsHome />
54           <h1 className="text-white font-bold mb-5 text-xl inline-block p-2 bg-red-700 my-5">
55             {/* Display a title for characters */}
56             CHARACTERS
57           </h1>
58           {/* Render the component for displaying characters */}
59           <CharactersHome />
60           <div className="flex justify-center text-center m-4 md:my-16 text-white"></div>
61         </section>
62         <section className="">
63           {/* Render the footer component */}
64           <Footer />
65         </section>
66       </div>
67     </div>
68   );
69 };
70
71 export default Home;

```

II. Backend

A. Entity

Dans le cadre du développement du back-end, nous avons d'abord dû créer les entités qui seront par la suite les tables de la base de données. Dans l'exemple qui suit on trouve l'entité "Comics".

Les propriétés de la classe "Comics" correspondent aux colonnes de la table "Comics" dans la base de données, définissant des champs tels que l'identifiant, le titre, l'affiche, la date de publication, le synopsis, et la rareté des comics. Les annotations dans la classe sont utilisées pour configurer Doctrine, précisant les contraintes et les relations, notamment la relation avec l'entité "Author" qui relie chaque comic à un auteur.

```
src > Entity > Comics.php > ...
1  <?php
2
3  namespace App\Entity;
4
5  use App\Entity\UserCollection;
6  use App\Entity\WishCollection;
7  use Doctrine\ORM\Mapping as ORM;
8  use App\Repository\ComicsRepository;
9  use Doctrine\Common\Collections\Collection;
10 use Doctrine\Common\Collections\ArrayCollection;
11 use Symfony\Component\Serializer\Annotation\Groups;
12 use Symfony\Component\Validator\Constraints as Assert;
13
14 /**
15  * @ORM\Entity(repositoryClass=ComicsRepository::class)
16  */
17 class Comics
18 {
19     /**
20      * @ORM\Id
21      * @ORM\GeneratedValue
22      * @ORM\Column(type="integer")
23      * @Groups({"comics", "charactersWithRelation", "comicsWithRelation", "wishlist", "ownedli"})
24      */
25     private $id;
26
27     /**
28      * @ORM\Column(type="string", length=128)
29      * @Groups({"comics", "charactersWithRelation", "comicsWithRelation", "wishlist", "ownedli"})
30      * @Assert\NotBlank
31      * @Assert\Length(max=128)
32      */
33     private $title;
34
35     /**
36      * @ORM\Column(type="string", length=255)
37      * @Groups({"comics", "charactersWithRelation", "comicsWithRelation", "wishlist", "ownedli"})
38      * @Assert\NotBlank
39      * @Assert\Url
40      */
41     private $poster;
42
43     /**
44      * @ORM\Column(type="datetime_immutable", nullable=true)
45      * @Groups({"comics", "charactersWithRelation", "comicsWithRelation"})
46      * @Assert\NotBlank
47      */
48     private $released_at;
49
50     /**
51      * @ORM\Column(type="text", nullable=true)
52      * @Groups({"comics", "charactersWithRelation", "comicsWithRelation"})
53      */
54     private $synopsis;
55
56     /**
57      * @ORM\Column(type="smallint", nullable=true)
58      * @Groups({"comics", "charactersWithRelation", "comicsWithRelation"})
59      */
60     private $rarity;
61
62     /**
63      * @ORM\ManyToOne(targetEntity=Author::class, inversedBy="comics")
64      * @ORM\JoinColumn(nullable=true)
65      * @Groups({"comicsWithRelation"})
66      */
67     private $author;
68
69     /**
70      * @ORM\ManyToMany(targetEntity=Characters::class, inversedBy="comics")
```

B. Controller

Nous avons ensuite codé les controllers. Les contrôleurs sont responsables de l'interaction entre l'application web et la base de données. Ils récupèrent des données depuis la base de données, les transforment en JSON et renvoient ces données au front-end. En d'autres termes, les contrôleurs agissent comme un intermédiaire entre la base de données et la vue (le front-end). Ils permettent de récupérer, traiter et fournir des données à l'interface utilisateur (UI).

Dans l'exemple qui va suivre, les contrôleurs utilisent les repositories pour interroger la base de données et extraire les données. Ensuite, ils utilisent un sérialiseur pour convertir ces données en format JSON, qui peut être facilement consommé par le front-end, que ce soit un site web, une application mobile, ou tout autre client. Cela permet au front-end de recevoir et d'afficher les données de manière cohérente et structurée.

src > Controller >  ApiComicsController.php >  ApiComicsController

```

1  <?php
2
3  namespace App\Controller;
4
5  use App\Entity\Comics;
6  use App\Repository\ComicsRepository;
7  use App\Repository\CharactersRepository;
8  use Doctrine\ORM\EntityManagerInterface;
9  use Symfony\Component\HttpFoundation\Request;
10 use Symfony\Component\HttpFoundation\Response;
11 use Symfony\Component\Routing\Annotation\Route;
12 use Symfony\Component\HttpFoundation\JsonResponse;
13 use Symfony\Component\Serializer\SerializerInterface;
14 use Symfony\Component\Validator\Validator\ValidatorInterface;
15 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
16 use Symfony\Component\Serializer\Exception\NotEncodableValueException;
17
18
19
20 class ApiComicsController extends AbstractController
21 {
22     /**
23      * @Route("/api/comics", name="app_api_comics_list", methods={"GET"})
24      */
25     public function listComics(ComicsRepository $comicsRepository): JsonResponse
26     {
27         $comics = $comicsRepository->findAll();
28
29         return $this->json($comics, Response::HTTP_OK, [], ['groups' => 'comicsWithRelation']);
30     }
31
32
33     /**
34      * @Route("/api/comics/{id}", name="app_api_comics_show", methods={"GET"})
35      */
36     public function showComics($id, ComicsRepository $comicsRepository): JsonResponse
37     {
38         $comics = $comicsRepository->find($id);
39
40         if (!$comics) {
41             return $this->json(['message' => 'Le comics n\'existe pas'], Response::HTTP_NOT_FOUND);
42         }
43
44         return $this->json($comics, JsonResponse::HTTP_OK, [], ['groups' => 'comicsWithRelation']);
45     }
46
47     /**
48      * @Route("/api/home-comics", name="app_api_comics_home", methods={"GET"})
49      */
50     public function homeComics(ComicsRepository $comicsRepository): JsonResponse
51     {
52         $comics = $comicsRepository->findNineComics();
53
54         return $this->json($comics, Response::HTTP_OK, [], ['groups' => 'comicsWithRelation']);
55     }
56
57     /**
58      * @Route("/api/search-comics", name="api_search_comics", methods={"GET"})
59      */
60     public function searchComics(Request $request, ComicsRepository $comicsRepository): JsonResponse
61     {
62         // Retrieve the title of the comics
63         $title = $request->query->get('title');
64
65         // Use the custom query in order to do the search by title
66         $comics = $comicsRepository->findAllOrderByTitleSearch($title);
67
68         // Convert the array as a json
69         $response = [];

```

RÉALISATIONS PERSONNELLES

A. MarvelApiUrlGenerator :

Afin de peupler notre base de données avec des datas officielles, je me suis rendu sur le site <https://developer.marvel.com/> afin de récupérer une clé API pour avoir accès à leurs données.

The screenshot shows the Marvel Developer Portal website. At the top, there's a navigation bar with the Marvel logo and links for NEWS, COMICS, CHARACTERS, MOVIES, TV SHOWS, GAMES, VIDEOS, and MORE. Below this is a sub-navigation bar for the DEVELOPER PORTAL, including links for How-Tos, Interactive Documentation, My Developer Account, Help, and News and Updates. The main content area features a large banner with the text "CREATE AWESOME STUFF WITH THE WORLD'S GREATEST COMIC API" and a "GET STARTED" button. To the right of the text is an image of Iron Man. Below the banner, there are three sections: "API Documentation" with a document icon, "Test Calls" with a double-headed arrow icon, and "My Developer Account" with a person icon. Each section has a brief description of its purpose. At the bottom, there's a footer with links for ABOUT MARVEL, HELP/FAQS, ADVERTISING, DISNEY+, MARVEL INSIDER, and FOLLOW MARVEL.

CREATE AWESOME STUFF WITH THE WORLD'S GREATEST COMIC API

The Marvel Comics API allows developers everywhere to access information about Marvel's vast library of comics—from what's coming up, to 70 years ago.

GET STARTED

API Documentation »
Read the documentation and rules of the road for the Marvel Comics API.

Test Calls »
Use the interactive test page to explore and test API calls.



My Developer Account »
Check your account status and manage your API keys and referrers.

ABOUT MARVEL
HELP/FAQS

ADVERTISING
DISNEY+

MARVEL INSIDER
Get Rewarded for Being a Marvel Fan

FOLLOW MARVEL



NEWSCOMICSCHARACTERSMOVIESTV SHOWSG

DEVELOPER PORTAL | How-TosInteractive DocumentationMy Developer Ac

MY DEVELOPER ACCOUNT

Hi **grecob420804665!**
Here's your personal Marvel Comics API information:

Your public key

98ba40964d53c077c5bbf572e59d62ab

Your private key

d5933e5770e5cc5b2156240ee8adfaed002e3dbe

Read more about how to use your keys to sign requests. »

Your rate limit: **3000** calls/day

Number of calls your application can make per day.

Your authorized referrers

List any domains that can make calls to the Marvel Comics API using your API key here:

[delete](#)

[add a new referrer](#)

Note: List the domain and path only - don't include "http" or other scheme designations. Only use the characters `a - z`, `0 - 9`, `.`, `_`, `-`, and `*`.

Read more about how to authorize referring domains in browser-based apps and web sites. »

Tell us about your projects

Authentication for Server-Side Applications

Server-side applications must pass two parameters in addition to the `apikey` parameter:

ts - a timestamp (or other long string which can change on a request-by-request basis)

hash - a md5 digest of the `ts` parameter, your private key and your public key (e.g. `md5(ts+privateKey+publicKey)`)

For example, a user with a public key of "1234" and a private key of "abcd" could construct a valid call as follows:

`http://gateway.marvel.com/v1/public/comics?ts=1&apikey=1234&hash=ffd275c5130566a2916217b101f26150` (the hash value is the md5 digest of 1abcd1234)

Authorization Errors

The following errors are returned by the Marvel Comics API when issues with authorization occur. These errors are returned by all endpoints.

Error Code	Error Message	Reason for occurring
409	Missing API Key	Occurs when the <code>apikey</code> parameter is not included with a request.
409	Missing Hash	Occurs when an <code>apikey</code> parameter is included with a request, a <code>ts</code> parameter is present, but no hash parameter is sent. Occurs on server-side applications only.
409	Missing Timestamp	Occurs when an <code>apikey</code> parameter is included with a request, a hash parameter is present, but no <code>ts</code> parameter is sent. Occurs on server-side applications only.
401	Invalid Referrer	Occurs when a referrer which is not valid for the passed <code>apikey</code> parameter is sent.
401	Invalid Hash	Occurs when a <code>ts</code> , hash and <code>apikey</code> parameter are sent but the hash is not valid per the above hash generation rule.
405	Method Not Allowed	Occurs when an API endpoint is accessed using an HTTP verb which is not allowed for that endpoint.
403	Forbidden	Occurs when a user with an otherwise authenticated request attempts to access an endpoint to which they do not have access.

Une fois mon compte créé, je vais dans un premier temps stocker les deux clés publique et privée dans mon fichier `.env` pour des raisons de sécurité. Le fichier `.env` se trouvant dans le `.gitignore`, il n'est pas présent lors des sauvegardes sur Github.


```

1  # In all environments, the following files are loaded if they exist,
2  # the latter taking precedence over the former:
3  #
4  # * .env                contains default values for the environment variables needed by
5  # * .env.local          uncommitted file with local overrides
6  # * .env.$APP_ENV       committed environment-specific defaults
7  # * .env.$APP_ENV.local uncommitted environment-specific overrides
8  #
9  # Real environment variables win over .env files.
10 #
11 # DO NOT DEFINE PRODUCTION SECRETS IN THIS FILE NOR IN ANY OTHER COMMITTED FILES.
12 # https://symfony.com/doc/current/configuration/secrets.html
13 #
14 # Run "composer dump-env prod" to compile .env files for production use (requires symfony)
15 # https://symfony.com/doc/current/best\_practices.html#use-environment-variables-for-info
16
17 ###> symfony/framework-bundle ###
18 APP_ENV=dev
19 APP_SECRET=1efbbfe19151c593af94fde9ac2678b4
20 ###< symfony/framework-bundle ###
21
22 ###> doctrine/doctrine-bundle ###
23 # Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/
24 # IMPORTANT: You MUST configure your server version, either here or in config/packages/
25 #
26 # DATABASE_URL="sqlite:///kernel.project_dir%/var/data.db"
27 # DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=8&charset=utf8mb4"
28 # DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8mb4"
29 ###< doctrine/doctrine-bundle ###
30
31 ###> lexik/jwt-authentication-bundle ###
32 JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
33 JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
34
35 JWT_PASSPHRASE=8ae04946cfef32188e7c8f1ff2192a77ae9a651de8a2e66101c9d1a6059f8c0f
36 ###< lexik/jwt-authentication-bundle ###
37
38 ###> nelmio/cors-bundle ###
39 CORS_ALLOW_ORIGIN='^https?:/(localhost|127\.\0\.\0\.\1)(:[0-9]+)?$'
40 ###< nelmio/cors-bundle ###
41
42 ###> symfony/mailer ###
43 # MAILER_DSN=null://null
44 ###< symfony/mailer ###
45
46 ###> symfony/mailjet-mailer ###
47 # MAILER_DSN=mailjet+api://PUBLIC_KEY:PRIVATE_KEY@api.mailjet.com
48 # #MAILER_DSN=mailjet+smtp://PUBLIC_KEY:PRIVATE_KEY@in-v3.mailjet.com
49 ###< symfony/mailjet-mailer ###
50
51 ### Marvel API
52 MARVEL_PUBLIC_KEY=98ba40964d53c077c5bbf572e59d62ab
53 MARVEL_PRIVATE_KEY=d5933e5770e5cc5b2156240ee8adfaed002e3dbef
54
55

```

Je renseigne ensuite le fichier “service.yaml” dans lequel je vais définir une section “App/Service/MarvelApiUrlGenerator”. Il s’agit d’un exemple de configuration de service explicite. Le service “MarvelApiUrlGenerator” dépend de deux paramètres : “\$publicKey” et “\$privateKey”. Ces paramètres sont récupérés des paramètres définis dans la section “parameters” grâce à la notation %env(NOM_DU_PARAMETRE)%. Cela permet d’injecter les clés d’API Marvel dans le service.

```

1  # This file is the entry point to configure your own services.
2  # Files in the packages/ subdirectory configure your dependencies.
3
4  # Put parameters here that don't need to change on each machine where the app is deployed
5  # https://symfony.com/doc/current/best_practices.html#use-parameters-for-application-configuration
6  parameters:
7
8  services:
9      # default configuration for services in *this* file
10     defaults:
11         autowire: true      # Automatically injects dependencies in your services.
12         autoconfigure: true # Automatically registers your services as commands, event subscribers, etc.
13
14     # makes classes in src/ available to be used as services
15     # this creates a service per class whose id is the fully-qualified class name
16     App\:
17         resource: '../src/'
18         exclude:
19             - '../src/DependencyInjection/'
20             - '../src/Entity/'
21             - '../src/Kernel.php'
22
23     # add more service definitions when explicit configuration is needed
24     # please note that last definitions always *replace* previous ones
25     App\Service\MarvelApiUrlGenerator:
26         arguments:
27             $publicKey: '%env(MARVEL_PUBLIC_KEY)%'
28             $privateKey: '%env(MARVEL_PRIVATE_KEY)%'

```

Je vais ensuite suivre les instructions de la documentation de Marvel qui stipule qu’il manque un hash et un timestamp pour générer les URL pour accéder aux données de l’API.

Je vais donc créer un service dans Symfony qui va s’appeler “MarvelApiUrlGenerator” qui va générer des URL en utilisant la clé API que j’ai récupérée. Nous avons fait une commande similaire pendant notre formation avec “OmdbApi” pour récupérer des affiches des films en fonction de leur titre.

Dans ce service, j’ai créé la classe “MarvelApiUrlGenerator”. Son rôle est de générer des URLs pour interagir avec l’API Marvel. Pour cela, elle a besoin de deux clés d’authentification, une clé publique “publicKey” et une clé privée “privateKey”, qui sont fournies par Marvel.

La méthode “generateComicsUrl” génère une URL pour accéder à la liste des comics de Marvel. On peut spécifier deux paramètres optionnels, le nombre limite de comics par page et le décalage (offset) pour paginer les résultats. Je commence par construire l’URL de base pour l’endpoint des comics de l’API Marvel.

Le constructeur dans la classe `MarvelApiUrlGenerator` est utilisé pour initialiser cette classe avec les clés d'API (public et privée) de Marvel. Dans le contexte de cette classe, ces clés d'API sont des dépendances nécessaires pour générer les URL pour accéder aux données Marvel via leur API.

Ces clés sont stockées en tant qu'attributs de la classe pour pouvoir être utilisées plus tard lors de la génération des URL vers l'API Marvel.

Ensuite, je crée un timestamp qui représente l'heure actuelle. Je génère un hash en utilisant le timestamp, la clé privée et la clé publique. Cela assure la sécurité de la requête.

Enfin, j'assemble tous ces éléments pour former l'URL complète qui peut être utilisée pour récupérer la liste des comics de Marvel.

À l'intérieur de ces méthodes, les clés d'API (publique et privée) sont utilisées pour générer un "hash" sécurisé, qui est une partie essentielle de la requête d'authentification envers l'API Marvel.

```

1  <?php
2
3  namespace App\Service;
4
5  class MarvelApiUrlGenerator
6  {
7      private $publicKey;
8      private $privateKey;
9
10     public function __construct(string $publicKey, string $privateKey)
11     {
12         // Store the provided public and private keys for authentication
13         $this->publicKey = $publicKey;
14         $this->privateKey = $privateKey;
15     }
16
17     public function generateComicsUrl($limit = 100, $offset = 0)
18     {
19         // Base URL for comics endpoint
20         $baseUrl = 'https://gateway.marvel.com:443/v1/public/comics';
21
22         // Create a timestamp for the current time
23         $timestamp = time();
24
25         // Generate a hash using the timestamp, private key, and public key
26         $hash = md5($timestamp . $this->privateKey . $this->publicKey);
27
28         // Construct the comics URL with timestamp, public key, hash, limit, and offset
29         $url = "$baseUrl?ts=$timestamp&apikey=$this->publicKey&hash=$hash&limit=$limit&offset=$offset";
30
31         return $url;
32     }
33
34     public function generateCharactersUrl($limit = 100, $offset = 0)
35     {
36         // Base URL for characters endpoint
37         $baseUrl = 'https://gateway.marvel.com:443/v1/public/characters';
38
39         // Create a timestamp for the current time
40         $timestamp = time();
41
42         // Generate a hash using the timestamp, private key, and public key
43         $hash = md5($timestamp . $this->privateKey . $this->publicKey);
44
45         // Construct the characters URL with timestamp, public key, hash, limit, and offset
46         $url = "$baseUrl?ts=$timestamp&apikey=$this->publicKey&hash=$hash&limit=$limit&offset=$offset";
47
48         return $url;
49     }
50 }
51
52

```

Une fois ces 3 étapes effectuées, je peux créer la commande personnalisée "ImportComicscommand".

Cette commande est utilisée pour importer des données de comics Marvel depuis l'API Marvel et les stocker dans la base de données.

Tout d'abord, cette commande hérite de la classe `Command`, qui est fournie par Symfony et utilisée pour créer des commandes personnalisées.

Je configure cette commande en lui attribuant une description et une aide pour expliquer son objectif. La description indique que la commande importe des données de comics Marvel depuis l'API, et l'aide fournit des informations supplémentaires.

Dans le constructeur de la commande, j'injecte deux dépendances : `"MarvelApiUrlGenerator"` et `"EntityManagerInterface"`. Ces dépendances sont nécessaires pour générer l'URL de l'API Marvel et accéder à la base de données.

Lorsque j'exécute la commande, je commence par créer une instance de `"SymfonyStyle"` pour gérer l'interaction avec la console.

Ensuite, je génère l'URL pour récupérer les informations des comics Marvel à partir de `"MarvelApiUrlGenerator"`. Cette URL est utilisée pour envoyer une requête GET à l'API Marvel.

J'utilise la classe `"HttpClient"` de Symfony pour envoyer la requête HTTP et obtenir la réponse JSON. Je convertis ensuite cette réponse en un tableau PHP.

Les données récupérées de l'API Marvel se trouvent dans `"$marvelComics"`. Je parcoure ces données et pour chaque comic, je crée une instance de l'entité `Comics`. J'attribue le titre et, s'il existe, le synopsis du comic à cette instance.

Je vérifie également s'il y a une miniature (thumbnail) pour le comic. Si c'est le cas, je construis l'URL de la miniature. Si l'URL de la miniature ne se termine pas par `"image_not_available.jpg"`, je l'attribue à l'entité `Comics`.

Ensuite, je génère une date de sortie aléatoire pour chaque comic, et je persiste l'entité `Comics` en utilisant `EntityManager`.

Après avoir bouclé sur tous les comics Marvel, j'exécute finalement les opérations de base de données en appelant `flush()` sur l'`EntityManager`. Cela enregistre les comics dans la base de données.

Finalement, j'affiche un message de réussite dans la console pour indiquer que les données des comics Marvel ont été importées avec succès.

Il ne me reste plus qu'à exécuter la commande dans mon terminal et observer le résultat dans la base de données.

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

```
student@teleporter: /var/www/html/projet-10-o-comicverse-back$ php bin/console app:import-comics
```

```
[OK] Marvel comics data imported successfully!
```

```
student@teleporter: /var/www/html/projet-10-o-comicverse-back$
```

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes : | Trier sur l'index : Aucun(e)

Options

	id	author_id	title	poster	released_at	synopsis	rarity
<input type="checkbox"/>	5	10	X-Men: Days of Future Past (Trade Paperback)	http://i.annihil.us/u/prod/marvel/iimg/9/d0/58b5cf...	2004-09-25 00:00:00	"Re-live the legendary first journey into the dyst...	3
<input type="checkbox"/>	24	NULL	Ultimate Spider-Man Ultimate Collection Book 1 (Tr...	http://i.annihil.us/u/prod/marvel/iimg/6/c0/590799...	2016-03-21 00:24:35	Collecting the groundbreaking first year of Ultima...	NULL
<input type="checkbox"/>	26	NULL	Official Handbook of the Marvel Universe (2004) #1...	http://i.annihil.us/u/prod/marvel/iimg/b/40/4bc640...	2019-05-31 23:17:23	The spectacular sequel to last year's OFFICIAL HAN...	NULL
<input type="checkbox"/>	28	NULL	X-Men: Fall Of The Mutants (2010)	http://i.annihil.us/u/prod/marvel/iimg/e/c0/4cbcd3...	2016-09-21 21:09:22	The body count rises higher than ever as the X-Men...	NULL
<input type="checkbox"/>	30	NULL	Penance: Relentless (2008)	http://i.annihil.us/u/prod/marvel/iimg/9/90/4bb860...	2015-02-13 19:54:42	From the pages of CIVIL WAR: FRONT LINE and THUNDE...	NULL
<input type="checkbox"/>	32	NULL	Punishermax: Kingpin (2010)	http://i.annihil.us/u/prod/marvel/iimg/5/90/4c4e01...	2013-11-11 18:40:41	The Punisher's one-man war on crime has been going...	NULL
<input type="checkbox"/>	34	NULL	MARVEL MASTERWORKS: THE UNCANNY X-MEN VOL. 3 HC (T...	http://i.annihil.us/u/prod/marvel/iimg/9/10/4bb3c9...	2014-05-27 16:54:53	Setting a new standard for Marvel super heroes was...	NULL
<input type="checkbox"/>	36	NULL	Kabuki Reflections Vol. 1 (Hardcover)	http://i.annihil.us/u/prod/marvel/iimg/e/0/4bac3a...	2016-05-17 11:05:49	Collecting the first six art books of Marvel's REF...	NULL
<input type="checkbox"/>	40	NULL	Civil War (Hardcover)	http://i.annihil.us/u/prod/marvel/iimg/8/c0/51dda5...	2014-03-03 02:39:25	The landscape of the Marvel Universe is changing. ...	NULL
<input type="checkbox"/>	42	NULL	Incredible Hulks: Dark Son (2010)	http://i.annihil.us/u/prod/marvel/iimg/b/70/4cbda1...	2023-05-04 19:33:10	Bruce Banner is the Hulk once again, and far from ...	NULL
<input type="checkbox"/>	46	NULL	The Stand: American Nightmares HC (Hardcover)	http://i.annihil.us/u/prod/marvel/iimg/a/10/4bb598...	2020-12-12 06:04:56	The deadly super flu Captain Trips has devastated ...	NULL
<input type="checkbox"/>	48	NULL	Gwen Stacy (2020) #3	http://i.annihil.us/u/prod/marvel/iimg/a/10/5e9e07...	2021-08-02 14:24:26	Captain George Stacy (A.K.A. Gwen's dad) has been ...	NULL
<input type="checkbox"/>	50	NULL	Nebula (2020) #3	http://i.annihil.us/u/prod/marvel/iimg/6/e0/5e8653...	2023-01-29 17:19:57	Note: The on-sale date listed here is subject to c...	NULL
<input type="checkbox"/>	52	NULL	PUNISHER VS. THE MARVEL UNIVERSE TPB (Trade Paperb...	http://i.annihil.us/u/prod/marvel/iimg/c/80/56be3d...	2018-11-20 13:27:21	The Punisher quite literally kills the Marvel Univ...	NULL
<input type="checkbox"/>	54	NULL	SPIDER-MAN BY TOM TAYLOR TPB (Trade Paperback)	http://i.annihil.us/u/prod/marvel/iimg/c/b0/64e3bf...	2020-11-13 19:55:18	Collects Friendly Neighborhood Spider-Man (2019) #...	NULL
<input type="checkbox"/>	56	NULL	Marvel Age Spider-Man Vol. 2: Everyday Hero (Diges...	http://i.annihil.us/u/prod/marvel/iimg/9/20/4bc665...	2019-02-23 09:08:25	"The Marvel Age of Comics continues! This time aro...	NULL
<input type="checkbox"/>	60	NULL	Marvel Previews (2017)	http://i.annihil.us/u/prod/marvel/iimg/c/80/5e3d75...	2018-09-27 06:18:49	Shathra's army wants to destroy all of Spider-Man'...	NULL
<input type="checkbox"/>	62	NULL	X-MEN EPIC COLLECTION: LEGACIES TPB (Trade Paperb...	http://i.annihil.us/u/prod/marvel/iimg/3/40/643713...	2015-12-18 06:11:06	In this new event spanning two issues, the formula...	NULL
<input type="checkbox"/>	64	NULL	ULTIMATE X-MEN VOL. 5: ULTIMATE WAR TPB (Trade Pap...	http://i.annihil.us/u/prod/marvel/iimg/2/0/4bc667...	2022-05-30 09:56:02	In an apocalyptic future, mutants are hunted by hu...	NULL
<input type="checkbox"/>	67	NULL	IMMORTAL X-MEN BY KIERON GILLEN VOL. 3 TPB (Trade ...	http://i.annihil.us/u/prod/marvel/iimg/c/b0/64e3be...	2017-08-02 04:35:28	Professor Xavier is the target of an assassination...	NULL

Options

☐ Tout cocher | Avec la sélection : Éditer Copier Supprimer Exporter

B. ApiRegisterController :

Dans le contrôleur "ApiRegisterController", je m'occupe de la création d'un nouvel utilisateur dans l'application Symfony lorsque des données JSON sont soumises à l'endpoint /api/register via une requête HTTP POST. Voici comment ce contrôleur fonctionne :

Je commence par recevoir une demande POST avec des données JSON. Ces données JSON contiennent les informations d'un nouvel utilisateur que nous souhaitons créer.

Ensuite, j'utilise le service de sérialisation "SerializerInterface" pour désérialiser les données JSON en un objet de la classe User. Cela signifie que je prends les données JSON et les transforme en une instance d'un utilisateur.

Je crée également des instances des classes "UserCollection" et "WishCollection". Ces classes servent à créer une nouvelle "UserCollection" et une nouvelle "WishCollection" dès la création d'un nouveau "User".

Je valide ensuite l'objet User nouvellement créé en utilisant un validateur "ValidatorInterface". Cette validation permet de s'assurer que les données de l'utilisateur sont correctes et répondent aux contraintes spécifiées. Si des

erreurs de validation sont détectées, je renvoie une réponse JSON avec un code d'état 422 (HTTP_UNPROCESSABLE_ENTITY) contenant les erreurs de validation.

Pour des raisons de sécurité, je hache le mot de passe de l'utilisateur à l'aide de "UserPasswordHasherInterface". Le mot de passe est initialement en clair dans les données JSON, mais je le sécurise en le hachant avant de le stocker dans la base de données.

Je configure également les relations entre l'utilisateur nouvellement créé, la collection d'utilisateurs et la collection de souhaits.

Enfin, j'utilise l'EntityManager pour persister l'utilisateur nouvellement créé en base de données. Une fois que l'utilisateur est persisté avec succès, je renvoie une réponse JSON avec un code d'état 201 (HTTP_CREATED) pour indiquer que le nouvel utilisateur a été créé avec succès.

```

1 <?php
2
3 namespace App\Controller;
4
5 use App\Entity\User;
6 use App\Entity\UserCollection;
7 use App\Entity\WishCollection;
8 use Doctrine\ORM\EntityManagerInterface;
9 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
10 use Symfony\Component\HttpFoundation\JsonResponse;
11 use Symfony\Component\HttpFoundation\Request;
12 use Symfony\Component\HttpFoundation\Response;
13 use Symfony\Component\Routing\Annotation\Route;
14 use Symfony\Component\Serializer\SerializerInterface;
15 use Symfony\Component\Validator\Validator\ValidatorInterface;
16 use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
17
18 class ApiRegisterController extends AbstractController
19 {
20     private $serializer;
21
22     public function __construct(SerializerInterface $serializer)
23     {
24         $this->serializer = $serializer;
25     }
26
27     /**
28      * @Route("/api/register", name="app_api_register", methods={"POST"})
29      */
30     public function register(Request $request, ValidatorInterface $validator, EntityManagerInterface $entityManager, UserPasswordHasherInterface $passwordHasher): JsonResponse
31     {
32         // deserialize and validate the JSON data
33         $newUser = $this->serializer->deserialize($request->getContent(), User::class, 'json');
34
35         // ...
36         $userCollection = new UserCollection();
37         $wishCollection = new WishCollection();
38         // ...
39
40         $errors = $validator->validate($newUser, null, ["registration"]);
41
42         if (count($errors) > 0) {
43             return $this->json($errors, Response::HTTP_UNPROCESSABLE_ENTITY);
44         }
45
46         // hashed password
47         $plainPassword = $newUser->getPassword();
48         $hashedPassword = $passwordHasher->hashPassword($newUser, $plainPassword);
49         $newUser->setPassword($hashedPassword);
50         // ...
51         $newUser->setUserCollection($userCollection);
52         $newUser->setWishCollection($wishCollection);
53
54         $wishCollection->setUser($newUser);
55         $userCollection->setUser($newUser);
56         // ...
57         $entityManager->persist($newUser);
58         $entityManager->flush();
59
60         return $this->json(['message' => 'Nouvel utilisateur créé.'], Response::HTTP_CREATED);
61     }
62 }

```

C. Access Control

Dans le fichier “security.yaml”, j’ai configuré un système de contrôle d’accès pour gérer qui peut accéder à quelles parties de mon site web. Dans ce cadre, j’ai défini deux rôles principaux, “ROLE_ADMIN” et “ROLE_USER”, ainsi qu’un rôle spécial nommé “PUBLIC_ACCESS”. Les utilisateurs ayant le rôle “ROLE_ADMIN” disposent de privilèges étendus, tandis que ceux ayant le rôle “ROLE_USER” ont des privilèges plus limités. Les ressources marquées avec “PUBLIC_ACCESS” sont accessibles au public sans restriction.

Pour définir quel rôle peut accéder à quelles parties du site, j’ai établi différentes règles. Par exemple, les opérations de “CRUD” (Create, Read, Update, Delete) ne sont accessibles qu’aux utilisateurs ayant le rôle “ROLE_ADMIN”. Les routes commençant par “/api/wish-list/remove/” permettent aux utilisateurs ayant le rôle “ROLE_USER” de supprimer des éléments de leur liste de souhaits. Les routes commençant par “/api/character/” sont accessibles au public sans restriction, tout comme les informations sur les personnages et les bandes dessinées (routes commençant par “/api/character/”).

J’ai également mis en place une hiérarchie des rôles, où “ROLE_ADMIN” hérite des droits de “ROLE_USER”. Ainsi, les utilisateurs ayant le rôle “ROLE_ADMIN” ont tous les privilèges des utilisateurs ayant le rôle “ROLE_USER”.

```

47 # Easy way to control access for large sections of your site
48 # Note: Only the *first* access control that matches will be used
49 access_control:
50
51     # * CRUD operation are accessible only for the admin, you can only s
52     - { path: ^/admin, roles: ROLE_ADMIN }
53     - { path: ^/api/wish-list/remove/\d+, roles: ROLE_USER }
54     - { path: ^/api/own-list/remove/\d+, roles: ROLE_USER }
55     - { path: ^/api/wish-list/add/\d+, roles: ROLE_USER }
56     - { path: ^/api/own-list/add/\d+, roles: ROLE_USER }
57     - { path: ^/api/wish-list, roles: ROLE_USER }
58     - { path: ^/api/own-list, roles: ROLE_USER }
59     - { path: ^/api/character/\d+, roles: PUBLIC_ACCESS }
60     - { path: ^/api/home-character, roles: PUBLIC_ACCESS }
61     - { path: ^/api/character/\d+, roles: PUBLIC_ACCESS }
62     - { path: ^/api/character, roles: PUBLIC_ACCESS }
63     - { path: ^/api/comics/\d+, roles: PUBLIC_ACCESS }
64     - { path: ^/api/home-comics, roles: PUBLIC_ACCESS }
65     - { path: ^/api/search-comics, roles: PUBLIC_ACCESS }
66     - { path: ^/api/comics, roles: PUBLIC_ACCESS }
67     - { path: ^/api/login, roles: PUBLIC_ACCESS }
68     # ! ROUTES API
69     - { path: ^/api/admin, roles: ROLE_ADMIN, methods: POST }
70     - { path: ^/api/admin, roles: ROLE_ADMIN, methods: DELETE }
71     - { path: ^/api/admin, roles: ROLE_ADMIN, methods: PUT }
72
73
74 role hierarchy:
75     ROLE_ADMIN: ROLE_USER
76
77

```


Dans ma configuration de contrôle d'accès, j'utilise des expressions régulières (regex) pour définir des modèles de chaînes de caractères qui correspondent aux URL demandées par les utilisateurs. Cela me permet de spécifier quelles règles d'accès s'appliquent à des URL spécifiques.

Par exemple, en utilisant l'expression régulière `"/api/wish-list/remove/\d+"`, je définis un modèle pour les URL qui commencent par `/api/wish-list/remove/` suivi de un ou plusieurs chiffres. Cela signifie que seules les URL correspondant à ce modèle seront soumises à la règle d'accès spécifiée, ce qui permet aux utilisateurs ayant le rôle `"ROLE_USER"` de supprimer des éléments de leur liste de souhaits.

De la même manière, avec l'expression régulière `"/api/character/\d+"`, je crée un modèle pour les URL qui commencent par `/api/character/` suivi de un ou plusieurs chiffres. Ces URL sont accessibles au public sans restriction, car elles correspondent au rôle `"PUBLIC_ACCESS"`.

En utilisant ces expressions régulières, je peux définir des modèles d'URL flexibles pour le contrôle d'accès, ce qui me permet de précisément déterminer quelles règles s'appliquent à chaque partie de mon site en fonction de leur structure d'URL.

JEU D'ESSAI

Pour réaliser le jeu d'essai, je vais créer un comics via le back office en tant qu'admin, ensuite nous ferons les vérifications sur le front et dans la base de données.

Et dans un deuxième temps, je déconnecterai l'admin pour créer un nouvel utilisateur en front via le formulaire d'enregistrement et je vais insérer un comics dans chacune des de ses deux listes de favoris.

Ici on se connecte au CRUD du back office et on rentre les informations de création de mon comics test.

Espace utilisateur Liste des comics Déconnexion Bienvenue, ben@oclock.io

Ajouter un nouveau comics

Titre du comics

TEST COMICS


URL de l'image de couverture

www.test.com

synopsis

Tempore facere ipsa

Année de sortie

05/03/1975 09:21 

Rareté du comics

Rare


Auteur du comics

Meyer

Personnages

☒ A-Bomb (HAS)
☐ Abomination (Emil Blonsky)
☐ Adam Warlock
☐ Alex Wilder
☐ Attuma
☒ Avengers

Ici je reçois un message flash de confirmation de création



[Espace utilisateur](#)
[Liste des comics](#)
[Déconnexion](#)
[Bienvenue, ben@oclock.io](#)

Comics ajouté avec succès!

Ajouter un comics

Titre	Poster	Date de sortie	A
Civil War (Hardcover)	http://i.annihil.us/u/prod/marvel/i/mg/8/c0/51dda501724ed.jpg	03/03/2014	
Gwen Stacy (2020) #3	http://i.annihil.us/u/prod/marvel/i/mg/a/10/5e9e076819aec.jpg	02/08/2021	
IMMORTAL X-MEN BY KIERON GILLEN VOL. 3 TPB (Trade Paperback)	http://i.annihil.us/u/prod/marvel/i/mg/c/b0/64e3bed77b17e.jpg	02/08/2017	
Incredible Hulks: Dark Son (2010)	http://i.annihil.us/u/prod/marvel/i/mg/b/70/4cbda1eb06127.jpg	04/05/2023	
Kabuki Reflections Vol. 1 (Hardcover)	http://i.annihil.us/u/prod/marvel/i/mg/e/e0/4bac3ad5d17c7.jpg	17/05/2016	

Ici je vérifie que le comics apparaît bien dans la liste des comics.



IMMORTAL X-MEN BY KIERON GILLEN VOL. 3 TPB (Trade Paperback)

Professor Xavier is the target of an assassination attempt. The primary suspect is Cable, the controversial leader of the X-Force team. Other mutant teams, including the X-Men and Factor-X, are seeking the truth. One wonders what Apocalypse and Sinister have to do with this case! Enormous secrets are about to be revealed about key members of the mutant family!

Je possède
Je le veux

TEST COMICS

Tempore facere ipsa

Je possède
Je le veux

Ici je vérifie que le comics a bien été ajouté dans la base de données

<input type="checkbox"/>				62	NULL	X-MEN EPIC COLLECTION: LEGACIES TPB (Trade Paperba...	http://i.annihil.us/u/prod/marvel/i/mg/3/40/643713...	2015-12-18 06:11:06	In this new event spanning two issues, the fo
<input type="checkbox"/>				64	NULL	ULTIMATE X-MEN VOL. 5: ULTIMATE WAR TPB (Trade Pap...	http://i.annihil.us/u/prod/marvel/i/mg/2/f0/4bc667...	2022-05-30 09:56:02	In an apocalyptic future, mutants are hunted
<input type="checkbox"/>				67	NULL	IMMORTAL X-MEN BY KIERON GILLEN VOL. 3 TPB (Trade ...	http://i.annihil.us/u/prod/marvel/i/mg/c/b0/64e3be...	2017-08-02 04:35:28	Professor Xavier is the target of an assassin
<input type="checkbox"/>				75	12	TEST COMICS	http://www.test.com	2001-09-03 21:38:00	Tempore facere ipsa

☐ Tout cocher
 Avec la sélection : Éditer Copier Supprimer Exporter

☐ Tout effacer | Nombre de lignes : 400 | Filtrer les lignes : | Trier par l'index :

Je vais maintenant me déconnecter afin de créer un nouvel utilisateur, que nous appellerons “test”, via le formulaire Register en front sur la page d'accueil.

Dans un deuxième temps, une fois connecté en temps qu'utilisateur “test”, j'insérerais un comics dans chacune des deux listes.

Pour terminer nous irons vérifier la création de l'utilisateur ainsi que ses deux tables associées “UserCollection” et “WishCollection”, ainsi que la présence des comics insérés dans les tables “Collection” directement dans la base de données.

Ici je m'enregistre en tant que nouvel utilisateur “test”.

Inscription

Prénom:

Nom:

Email:

Nom d'utilisateur:

4 à 24 caractères.
 Doit commencer par une lettre.
 Les lettres, les chiffres, les traits de soulignement et les traits d'union sont autorisés.

Mot de passe:

8 à 24 caractères.
 Doit inclure une minuscule, une majuscule, un chiffre et un caractère spécial.
 Caractères spéciaux acceptés: ! @ # \$ %

Confirmer le mot de passe:

Doit correspondre au premier mot de passe saisi.

Déjà inscrit ?
 Connectez-vous !
 Accueil

Et je me connecte en tant qu'utilisateur et j'ajoute un comics dans chacune de mes tables "Collection"



Ensuite je vérifie la création de l'utilisateur dans la base de données. Ici on constate bien la présence de l'utilisateur "test" et son "ID" est 38.

	id	email	roles (JSON)	password	firstname	lastname	username
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	thanh@oclock.io	["ROLE_ADMIN"]	\$2y\$13\$lyCfW46By2fWjd1MKZuYFDWfuWV3Vv8nNT081...	Marthe	Goncalves	Thanh
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	nicolas@oclock.io	["ROLE_ADMIN"]	\$2y\$13\$JnFQUXBGXeV7HM1JZsu0u3SMJScBj2rF0galBf3eV...	Alex	Renard	Nicolas
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	hocine@oclock.io	["ROLE_ADMIN"]	\$2y\$13\$W3TAP60SEVTMhd8LrYY3rDMjfeZtwbrCyB6SuSjiqz...	Margaud	Rey	Hocine
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	4	admin@oclock.io	["ROLE_ADMIN"]	\$2y\$13\$3dwHPZLdBWegeTL3wLjJM.cuEu5rEvelRkhkVSSNlg0...	Aimé	Lefebvre	Admin
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	5	ben@oclock.io	["ROLE_ADMIN"]	\$2y\$13\$GBXp2du0EmhMau2fR8A6ekkFz72PV8LspF8ls8ZL8...	Émile	Parent	Ben
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	6	user@oclock.io	[]	\$2y\$13\$JNGmM1qI7PidhDxQXmCUZ.hYx87/W1U1HdpgwLY2yrk...	Anastasie	Lebrun	User
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	34	jeudessai@test.fr	["ROLE_USER"]	\$2y\$13\$J6UyOJQsk6FKf06kjqENO0mqBdsCmBwILCV0.1k4On...	Oclock	Ecole	jeudessai
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	38	test@oclock.io	["ROLE_USER"]	\$2y\$13\$U5Ndr8j30oPRAaeYsMjAae29i9Y00eLkbQVAYgEVp5y...	Test	Test	test

Je vérifie maintenant la création de la table "UserCollection" associée. On constate que le "User" ayant l' "ID" 38 possède bien une "UserCollection" ayant l' "ID" 15

	id	user_id
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	1
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	2
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	3
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	4	4
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	5	5
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	6	6
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	13	34
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	15	38

Pour terminer, je vérifie la présence du comics ajouté dans la table pivot “UserCollectionComics”. Dans la dernière ligne, on constate que la table “UserCollection” avec l’ “ID” 15 possède bien le comics avec l’ “ID” 5.

Récentes Préférées

Nouvelle base de données

information_schema

mysql

ocomics

Nouvelle table

author

character

comics

comics_characters

doctrine_migration_versions

user

user_collection

user_collection_comics

wish_collection

wish_collection_comics

performance_schema

vega

		user_collection_id	comics_id
<input type="checkbox"/>	Éditer Copier Supprimer	2	30
<input type="checkbox"/>	Éditer Copier Supprimer	2	32
<input type="checkbox"/>	Éditer Copier Supprimer	2	34
<input type="checkbox"/>	Éditer Copier Supprimer	2	36
<input type="checkbox"/>	Éditer Copier Supprimer	3	5
<input type="checkbox"/>	Éditer Copier Supprimer	4	5
<input type="checkbox"/>	Éditer Copier Supprimer	4	26
<input type="checkbox"/>	Éditer Copier Supprimer	4	30
<input type="checkbox"/>	Éditer Copier Supprimer	4	36
<input type="checkbox"/>	Éditer Copier Supprimer	4	42
<input type="checkbox"/>	Éditer Copier Supprimer	4	48
<input type="checkbox"/>	Éditer Copier Supprimer	4	52
<input type="checkbox"/>	Éditer Copier Supprimer	5	28
<input type="checkbox"/>	Éditer Copier Supprimer	5	30
<input type="checkbox"/>	Éditer Copier Supprimer	5	32
<input type="checkbox"/>	Éditer Copier Supprimer	6	5
<input type="checkbox"/>	Éditer Copier Supprimer	6	26
<input type="checkbox"/>	Éditer Copier Supprimer	6	28
<input type="checkbox"/>	Éditer Copier Supprimer	6	34
<input type="checkbox"/>	Éditer Copier Supprimer	6	36
<input type="checkbox"/>	Éditer Copier Supprimer	13	5
<input type="checkbox"/>	Éditer Copier Supprimer	15	5

VEILLE TECHNIQUE & SÉCURITÉ

I. CORS (Cross-Origin Resource Sharing)

La politique des CORS est une mesure de sécurité mise en place dans les navigateurs web pour contrôler les requêtes HTTP entre des domaines différents. Elle vise à empêcher les attaques potentielles et à protéger la confidentialité et la sécurité des utilisateurs.

Une requête est considérée comme "cross-origin" (depuis un autre domaine) lorsque l'origine de la page web émettrice de la requête diffère de l'origine de la ressource cible de la requête.

Nous avons utilisé le bundle "NelmioCorsBundle" qui nous a permis de définir les règles CORS. Ce bundle permet de définir les domaines qui auront accès à notre API REST. Ceux-ci étaient désactivés lors de l'utilisation du serveur en mode développement, mais activés en production avec uniquement acceptation des requêtes provenant du serveur front. Cette opération interdit le chargement à partir d'autres serveurs

```
config > packages > ! nelmio_cors.yaml
1  nelmio_cors:
2    defaults:
3      origin_regex: true
4      allow_origin: ['%env(CORS_ALLOW_ORIGIN)%']
5      allow_methods: ['GET', 'OPTIONS', 'POST', 'PUT', 'PATCH', 'DELETE']
6      allow_headers: ['Content-Type', 'Authorization']
7      expose_headers: ['Link']
8      max_age: 3600
9
10   paths:
11     '^/api/login_check':
12       allow_credentials: true
13
```

II. Token JWT

JWT, ou JSON Web Token, est un format de token utilisé pour sécuriser l'échange de données entre deux parties.

Nous avons choisi d'utiliser des JWT pour gérer l'authentification et l'autorisation entre le front-end React et le back-end. Les JWT offrent plusieurs avantages essentiels pour notre architecture. Tout d'abord, ils sont auto-suffisants, ce qui signifie que toutes les informations nécessaires pour vérifier l'identité de l'utilisateur

et ses droits d'accès sont incluses dans le token lui-même. Cela réduit la dépendance sur le serveur pour stocker des sessions.

Les JWT sont sécurisés. Grâce à la signature cryptographique, nous pouvons garantir l'intégrité des données et empêcher toute altération ou falsification des tokens.

Le bundle "JWTAuthenticationBundle" nous a offert la possibilité d'utiliser les JSON Web Tokens afin de protéger les ressources de notre API REST. De fait, lorsque l'utilisateur s'authentifie correctement, un token lui est retourné. Ce token est ensuite transmis côté front au serveur afin de permettre à l'utilisateur de lire ou d'écrire les ressources protégées à chaque requête.

```

config > packages > ! security.yaml
1  security:
2      enable_authenticator_manager: true
3      # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
4      password_hashers:
5          Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
6      # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
7      providers:
8          # used to reload user from session & other features (e.g. switch_user)
9          app_user_provider:
10             entity:
11                 class: App\Entity\User
12                 property: email
13
14      firewalls:
15          dev:
16             pattern: ^/((profiler|wdt)|css|images|js)/
17             security: false
18          login:
19             pattern: ^/api/login
20             stateless: true
21             json_login:
22                 check_path: /api/login_check
23                 username_path: email
24                 success_handler: lexik_jwt_authentication.handler.authentication_success
25                 failure_handler: lexik_jwt_authentication.handler.authentication_failure
26
27          api:
28             pattern: ^/api
29             stateless: true
30             jwt: ~
31      main:
32          lazy: true
33          provider: app_user_provider
34          form_login:
35             login_path: app_security_login
36             check_path: app_security_login
37          logout:
38             path: app_security_logout
39
40
41      # activate different ways to authenticate
42      # https://symfony.com/doc/current/security.html#the-firewall
43
44      # https://symfony.com/doc/current/security/impersonating-user.html
45      # switch_user: true
46
47      # Easy way to control access for large sections of your site
48      # Note: Only the *first* access control that matches will be used
49      access_control:

```

III. Security.yaml

Dans le fichier `config/packages/security.yaml`, nous configurons les règles d'authentification et d'autorisation pour notre application. Cela signifie que nous spécifions qui peut accéder à quelles parties de l'application. Par exemple, nous exigeons une authentification pour les routes situées dans le dossier 'api' qui contiennent les points d'accès de notre application.

Nous avons également configuré l'algorithme de hachage des mots de passe utilisé pour notre entité utilisateur (User). Par exemple, nous avons choisi d'utiliser un algorithme de hachage spécifique pour sécuriser les mots de passe.

En ce qui concerne les rôles, nous avons défini un rôle utilisateur appelé "ROLE_USER" qui est attribué à tous les utilisateurs inscrits. Ce rôle permet aux utilisateurs d'accéder à certaines parties de l'application.

Nous avons défini un rôle administrateur appelé "ROLE_ADMIN" qui est attribué aux administrateurs. Ce rôle permet d'accéder au back office et à toutes les fonctions qui en découlent.

IV. CSRF Token

Le CSRF Token, abréviation de "Cross-Site Request Forgery Token" (jeton anti-CSRF), est une mesure de sécurité utilisée dans les applications web pour protéger contre les attaques de type Cross-Site Request Forgery (CSRF). Les attaques CSRF surviennent lorsque des acteurs malveillants tentent de faire effectuer des actions non autorisées à un utilisateur connecté sans son consentement.

Dans la fonction "Delete" du CRUD des comics sur notre back office, nous avons ajouté une protection CSRF Token afin de renforcer la sécurité de cette fonction.

```
/**
 * @Route("/admin/comics/delete/{id}", name="app_admin_comics_delete", methods={"POST"}, requirements={"id"="\d+"})
 */
public function delete(Request $request, Comics $comics, ComicsRepository $comicsRepository): Response
{
    if ($this->isCsrfTokenValid('delete.'.$comics->getId(), $request->request->get('_token'))) {
        $comicsRepository->remove($comics, true);
        $this->addFlash("success", "Le comics a bien été supprimé.");
    }
    else {
        $this->addFlash("danger", "Erreur, le comics n'a pas été supprimé!");
    }
    return $this->redirectToRoute('app_admin_comics_list', [], Response::HTTP_SEE_OTHER);
}
```

Lorsque l'admin soumet le formulaire pour supprimer le comics, le jeton CSRF est inclus en tant que champ caché dans le formulaire. Il n'est pas visible pour l'utilisateur. Pour cela on se sert du champ "hidden" dans le formulaire.

```
templates > admin > comics > _delete_form.html.twig
1 <form class="my-4" method="post" action="{{ path('app_admin_comics_delete', {'id': comics.id}) }}"
2   onsubmit="return confirm('Voulez-vous vraiment supprimer ce Comics? Cette action est définitive.');"
3   <input type="hidden" name="_token" value="{{ csrf_token('delete' ~ comics.id) }}"
4   <button class="badge bg-danger fs-6 my-4 mx-2 text-decoration-none">Supprimer le comics</button>
5 </form>
6
```

DIFFICULTÉS RENCONTRÉES

Lorsque j'ai déployé la partie front de mon site sur Vercel, j'ai choisi une plateforme qui utilise le protocole HTTPS par défaut. HTTPS est une version sécurisée du protocole HTTP, qui garantit que les données échangées entre le navigateur des utilisateurs et le serveur sont chiffrées, ce qui protège la confidentialité des informations. Cela est particulièrement important lors de la transmission de données sensibles, telles que les identifiants de connexion ou les informations personnelles.

Cependant, le back de mon site était déployé sur le serveur distant de mon école, qui utilisait initialement le protocole HTTP. Lorsque le front, qui est en HTTPS, essaie de communiquer avec le back en HTTP, cela peut poser des problèmes de sécurité. Les navigateurs modernes sont de plus en plus stricts en ce qui concerne la sécurité, et ils peuvent bloquer les requêtes provenant de sites en HTTPS vers des sites en HTTP, car cela peut présenter un risque potentiel d'interception de données.

Pour résoudre ce problème, j'ai dû migrer mon serveur distant vers HTTPS. Cela signifie que le serveur est désormais capable de chiffrer les données qu'il envoie au front, de la même manière que Vercel le fait. Ainsi, lorsque le front communique avec le back, les données sont sécurisées tout au long du processus, ce qui garantit la confidentialité et l'intégrité des informations échangées.

Pour migrer le serveur en HTTPS, je vais devoir obtenir un certificat SSL. SSL (Secure Sockets Layer) est un protocole de sécurité qui permet de chiffrer les données échangées entre un navigateur web et un serveur. Il est essentiel pour garantir la confidentialité, l'intégrité et la sécurité des informations transmises sur Internet. Voici pourquoi SSL est nécessaire pour passer à HTTPS, le protocole sécurisé.

Pour activer SSL sur un site web, un certificat SSL est nécessaire. C'est là que Certbot entre en jeu. Certbot est un outil qui simplifie considérablement le processus d'obtention et de déploiement de certificats SSL gratuits fournis par Let's Encrypt. Sans un outil comme Certbot, la configuration d'un certificat SSL serait complexe et fastidieuse.

Certbot automatise le processus de demande et de déploiement de certificats SSL, ce qui facilite grandement la sécurisation d'un site web. Il prend en charge des tâches telles que la génération de clés, la création de demandes de signature de certificat (CSR), l'interaction avec l'autorité de certification Let's Encrypt, et la configuration du serveur web (dans ce cas, Apache) pour utiliser le certificat SSL. Cela signifie que les propriétaires de sites web peuvent passer à HTTPS en quelques étapes simples, renforçant ainsi la sécurité de leur site et la confiance de leurs utilisateurs.

Je me suis servi de la ressource suivante pour faire cette opération:

<https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-20-04>

UTILISATION DE SITES ANGLOPHONES

Pour illustrer la veille technique réalisée au cours de ce projet, je vais utiliser la documentation en anglais de Symfony qui donne les instructions pour l'installation du bundle "lexik/jwt-authentication". Je vais donc traduire les informations de cette page en français.

<https://symfony.com/bundles/LexikJWTAuthenticationBundle/current/index.html>

Getting started

[Edit this page](#)

Prerequisites

This bundle requires Symfony 4.4+ and the openssl extension.

Protip: Though the bundle doesn't enforce you to do so, it is highly recommended to use HTTPS.

Installation

Add `lexik/jwt-authentication-bundle` to your `composer.json` file:

```
$ php composer.phar require "lexik/jwt-authentication-bundle"
```

Register the bundle

Register bundle into `config/bundles.php` (Flex did it automatically):

```
return [
    //...
    Lexik\Bundle\JWTAuthenticationBundle\LexikJWTAuthenticationBundle::class => ['all'
];
```

Generate the SSL keys

```
$ php bin/console lexik:jwt:generate-keypair
```

Your keys will land in `config/jwt/private.pem` and `config/jwt/public.pem` (unless you configured a different path).

Available options:

Pour Commencer

Prérequis

Ce bundle nécessite Symfony 4.4+ et l'extension openssl.

Conseil professionnel : Bien que le bundle ne vous oblige pas à le faire, il est fortement recommandé d'utiliser HTTPS.

Installation

Ajoutez lexik/jwt-authentication-bundle à votre fichier composer.json :

Enregistrez le bundle

Enregistrez le bundle dans le fichier config/bundles.php (Flex l'a fait automatiquement) :

Générez les clés SSL

Vos clés seront stockées dans les fichiers config/jwt/private.pem et config/jwt/public.pem (sauf si vous avez configuré un chemin différent).

Options disponibles :

2. Use the token

Simply pass the JWT on each request to the protected firewall, either as an authorization header or as a query parameter.

By default only the authorization header mode is enabled: `Authorization: Bearer {token}`

See the [configuration reference document](#) to enable query string parameter mode or change the header value prefix.

Examples

See [Functionally testing a JWT protected api document](#) or the sandbox application [\(Symfony4\)](#) for a fully working example.

Notes

About token expiration

Each request after token expiration will result in a 401 response. Redo the authentication process to obtain a new token.

Maybe you want to use a [refresh token](#) to renew your JWT. In this case you can check [JWTRefreshTokenBundle](#).

Working with CORS requests

This is more of a Symfony2 related topic, but see [Working with CORS requests document](#) to get a quick explanation on handling CORS requests.

Impersonation

For impersonating users using JWT, see https://symfony.com/doc/current/security/impersonating_user.html

Important note for Apache users

Utilisation du jeton

Il suffit de transmettre le JWT à chaque requête vers le pare-feu protégé, soit en tant qu'en-tête d'autorisation, soit en tant que paramètre de requête.

Par défaut, seul le mode d'en-tête d'autorisation est activé : `Authorization: Bearer {token}`

Consultez le document de référence de configuration pour activer le mode de paramètre de chaîne de requête ou modifier le préfixe de la valeur de l'en-tête.

Exemples Consultez le document [Functionally testing a JWT protected api](#) ou l'application [sandbox](#) (Symfony4) pour un exemple entièrement fonctionnel.

Notes À propos de l'expiration du jeton Chaque demande effectuée après l'expiration du jeton entraînera une réponse 401. Refaites le processus d'authentification pour obtenir un nouveau jeton.

Peut-être souhaitez-vous utiliser un jeton de rafraîchissement pour renouveler votre JWT. Dans ce cas, vous pouvez consulter [JWTRefreshTokenBundle](#).

[travailler](#) avec des demandes CORS Il s'agit davantage d'un sujet lié à Symfony2, mais consultez le document [Working with CORS requests](#) pour obtenir une explication rapide sur la gestion des demandes CORS.

[Impersonation](#) Pour l'impersonation d'utilisateurs à l'aide de JWT, consultez https://symfony.com/doc/current/security/impersonating_user.html.

Note importante pour les utilisateurs d'Apache

CONCLUSION

Pour conclure, je voudrais dire que ce premier projet collaboratif m'a fait prendre conscience de l'apport extraordinaire que pouvait apporter le travail en équipe, mais aussi la rigueur et la nécessité de la communication.

Ce projet m'a aussi permis de me rassurer sur la qualité de ma formation chez O'Clock qui à été tellement intense que, au bout des 5 mois de formation, j'étais persuadé d'avoir déjà oublié le début alors que tout ce que nous avons appris est ressorti dès que nous avons commencé le projet.

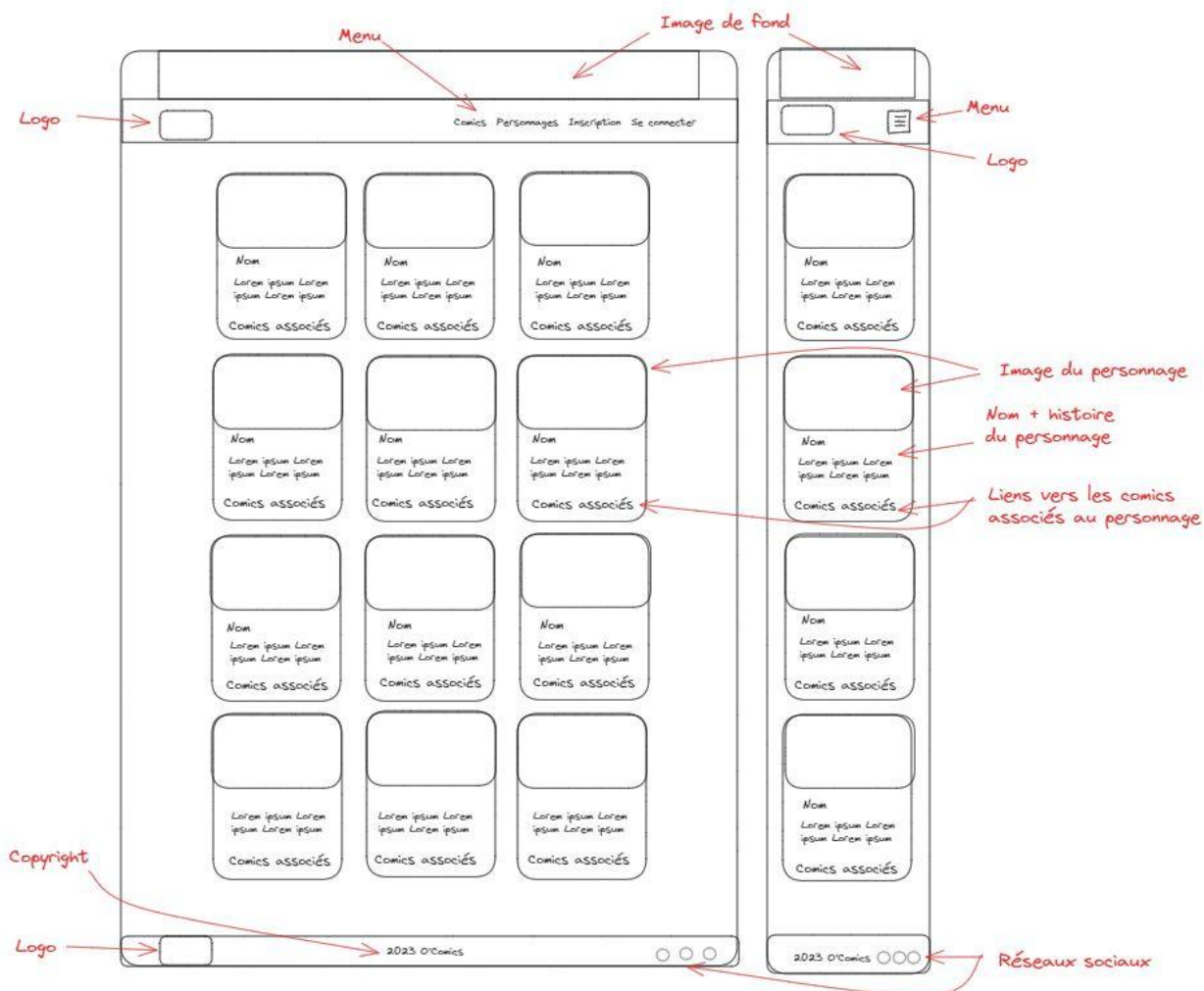
Je tiens d'ailleurs à féliciter et remercier tous les intervenants, formateurs et tuteurs qui nous ont encadrés pendant ces 6 mois pour leur bienveillance.

ANNEXES

Endpoints API

Contrôleur	Endpoint	Méthodes HTTP	Description
ApiCharacterController	/api/character	GET	List all the characters
ApiCharacterController	/api/home-character	GET	List the five first characters
ApiCharacterController	/api/character/{id}	GET	Show a specific character
ApiComicsController	/api/comics	GET	List all the comics
ApiComicsController	/api/comics/{id}	GET	Show a specific comics
ApiComicsController	/api/home-comics	GET	List of the nine first comics
ApiComicsController	/api/search-comics	GET	Search comics by title
ApiComicsController	/api/admin/comics/add	POST	Add a comics
ApiComicsController	/api/admin/comics/update/{id}	PUT	Update a comics
ApiComicsController	/api/admin/comics/delete/{id}	DELETE	Delete a comics
ApiOwnListController	/api/ownedlist	GET	List all the comics owned by a user
ApiOwnListController	/api/ownedlist/add/{comicsId}	POST	Add a comics in the list of the comics owned by a user
ApiOwnListController	/api/ownedlist/remove/{comicsId}	DELETE	Delete a comics
ApiWishListController	/api/wishlist	GET	List all the comics that a user wish to have
ApiWishListController	/api/wishlist/add/{comicsId}	POST	Add a comics in the list of the comics a user wish to have
ApiWishListController	/api/wishlist/remove/{comicsId}	DELETE	Remove a comics in the list of the comics a user wish to have
ApiRegisterController	/api/register	POST	Create a new User

Personnages en tant que visiteur



Personnages en tant qu'utilisateur

