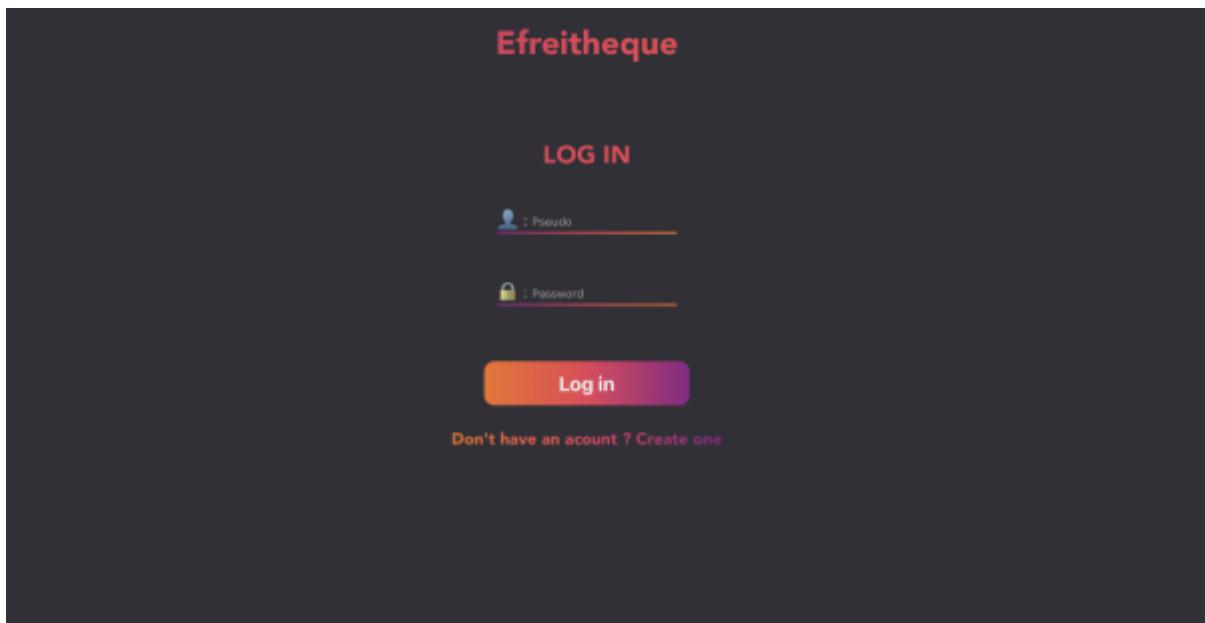


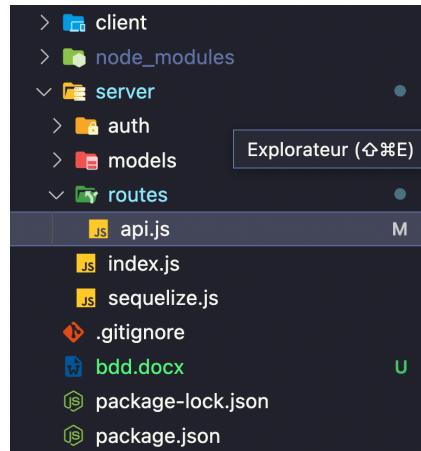
Projet web L3 Efreitheque

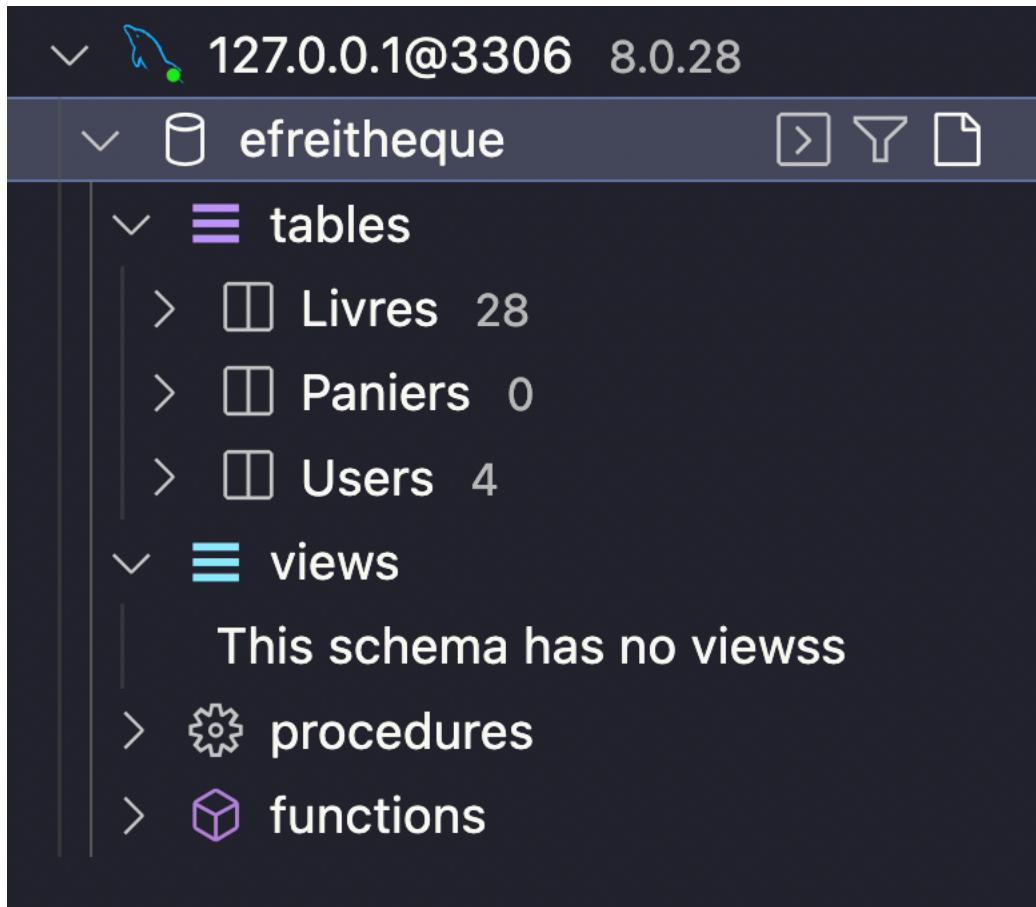


Page d'accueil

Le but de ce projet était de créer un site web avec le framework Vue js pour la partie frontend et node JS avec Express JS pour la partie backend. Pour la gestion de nos données nous avons utilisée une base de donné local en MySQL. Nous avons également utiliser GitHub.

La structure de notre projet s'est donc naturellement divisée en deux dossier principaux à savoir le dossier client et serveur.





The screenshot shows the MySQL Workbench interface with the following details:

- Query: SELECT * FROM `Livres` LIMIT 100;
- Results:

	* id int	* titre varchar(255)	* auteur varchar(255)	* publication varchar(255)	* couverture varchar(255)	* quantite int
1	1	Death of Doctor Strange	Jed Mackay	September 22, 2021	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
2	2	Death of Doctor Strange	Jed Mackay	October 20, 2021	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
3	3	Death of Doctor Strange	Jed Mackay	November 24, 2021	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
4	4	Death of Doctor Strange	Jed Mackay	December 29, 2021	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
5	5	Death of Doctor Strange	Jed Mackay	January 26, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
6	6	Star Wars: Obi-Wan (2022)	Christopher Cantwell	May 04, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
7	7	Star Wars: Obi-Wan (2022)	Christopher Cantwell	June 22, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
8	8	Star Wars: Obi-Wan (2022)	Christopher Cantwell	July 27, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
9	9	Spider-Man 2099: Exodus	Steve Orlando	May 04, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
10	10	Deadpool: Bad Blood (2022)	Rob Liefeld	April 06, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
11	11	Deadpool: Bad Blood (2022)	Chad Bowers, Rob Liefeld	June 01, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
12	12	Deadpool: Bad Blood (2022)	Chad Bowers, Rob Liefeld	July 20, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5
13	13	Deadpool: Bad Blood (2022)	Rob Liefeld	August 31, 2022	https://i.annihil.us/u/prod/000/02/00/0000000000000000	5

Nous avons utiliser VScode pour toute la gestion du projet ainsi que 2 extensions majeur. Tout d'abord Thunder client qui nous a permis de

tester toutes nos requêtes ainsi que l'extension Database qui nous a permis d'interagir avec notre bdd.

Maintenant que les outils ont été présenté, nous aborderons chaque étape de la conception du projet avec les problème que nous avons rencontré et les solutions apportées.

1) Le premier commit, la base du projet

```
● ● ●

1 const express = require('express');
2 const cors = require('cors');
3
4
5 const app = express();
6
7 const PORT = process.env.PORT || 3000
8
9 const apirest = require('../routes/api.js')
10
11
12 app.use(cors());
13 app.use('/api', apirest);
14
15
16 app.get('/', (req, res) => {
17   res.send('This is the backend of the project')
18 });
19
20
21
22 app.listen(PORT, () => { console.log(`Server started on port ${PORT}`)})
```

Nous avons d'abord mis en place le serveur en nous servant des connaissances vu en TP. Nous sommes repartis de zéro afin de bien

comprendre tout les mécanismes. Node JS ainsi que Vue JS et Vue CLI étant déjà installés nous avons donc commencer par la configuration du fichier package.json. Le point d'entrée sera index.js

Toute nos routes seront dans le fichier api

Ici le seul endpoint est un endpoint de test pour vérifier si le serveur fonctionne bien, il sera sur le port 3000.



```
 1 const express = require('express');
 2 const router = express.Router();
 3 const bcrypt = require('bcrypt');
 4 const jwt = require('jsonwebtoken');
 5 const privateKey = require('../auth/private_key')
 6 const auth = require('../auth/auth');
 7
 8 const sequelize = require('../sequelize')
 9 const { Op } = require("sequelize");
10
11 const { Livre } = require('../sequelize')
12 const { User } = require('../sequelize')
13 const { Panier } = require('../sequelize')
```

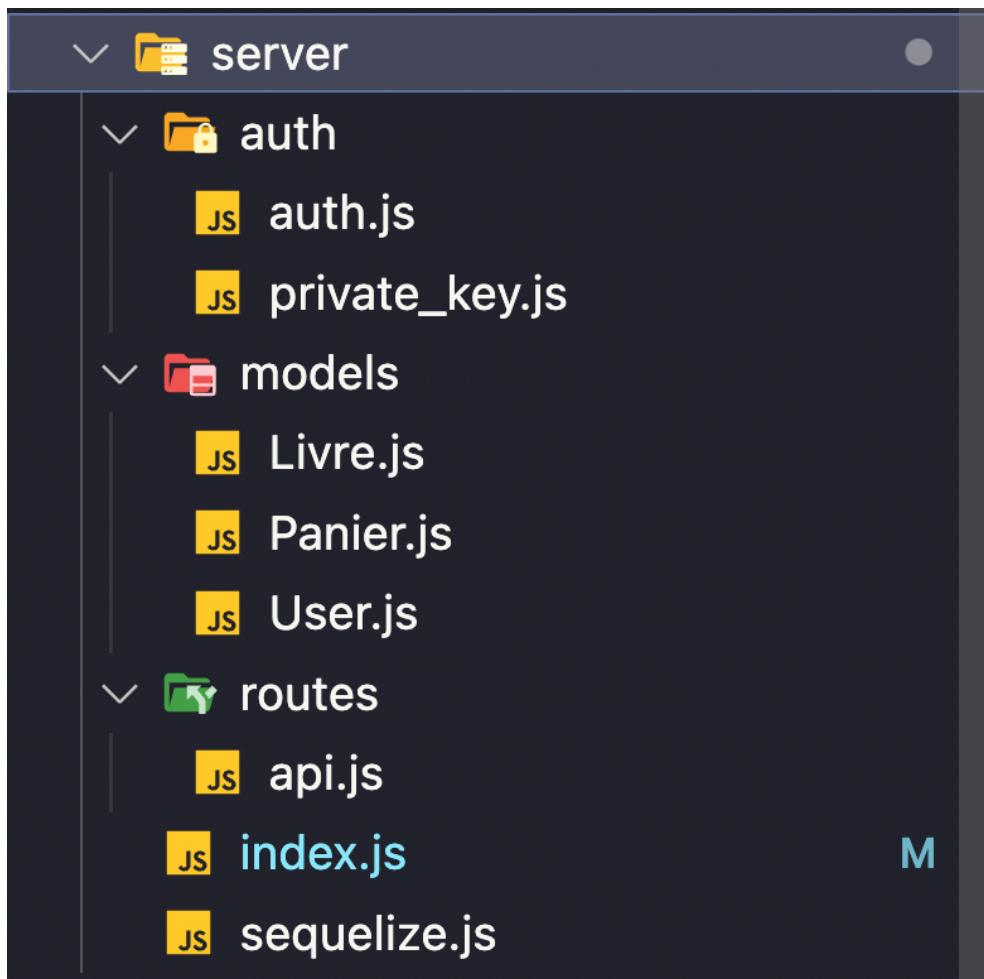
Intéressons nous maintenant au fichier API

Plusieurs choses a dire tout d'abord nous avons utiliser diverses package pour ce projet. Le module router nous vient de Vue Router qui permettra la gestion des page vue js.

Bcrypten jwtn privatekey et auth nous servirons pour la gestion de l'authentification avec le jeton web json.

Nous avons utiliser L'ORM sequelize pour la gestion de la base de données.

Voici comment se présente donc notre serveur, nous avons vraiment essayé de faire au plus simple afin d'en apprendre davantage sur le backend.



Les models représentent nos tables dans la BDD. Nous avons fait le choix d'en utiliser que 3.

Nous nous sommes énormément documenté sur l'outil sequelize afin d'en extraire son potentiel. Chaque models suit la même logique et ont été synchronisé avec le serveur dans le fichier sequelize.js

```

1 module.exports = (sequelize, DataTypes) => {
2   return sequelize.define('Livre', {
3     id: {
4       type: DataTypes.INTEGER,
5       primaryKey: true,
6       autoIncrement: true
7     },
8     titre: {
9       type: DataTypes.STRING,
10      allowNull: false
11    },
12    auteur: {
13      type: DataTypes.STRING,
14      allowNull: false
15    },
16    publication: {
17      type: DataTypes.STRING,
18      allowNull: false
19    },
20    couverture: {
21      type: DataTypes.STRING,
22      allowNull: false
23    },
24    quantite: {
25      type: DataTypes.INTEGER,
26      allowNull: false
27    },
28  },
29  {
30    timestamps: true,
31    createdAt: false,
32    updatedAt: false
33  })
34 }
35 }, {
36   timestamps: true,
37   createdAt: false,
38   updatedAt: false
39 })

```

```

1 module.exports = (sequelize, DataTypes) => {
2   return sequelize.define('Panier', {
3     id: {
4       type: DataTypes.INTEGER,
5       primaryKey: true,
6       autoIncrement: true
7     },
8     id_user: {
9       type: DataTypes.INTEGER,
10      allowNull: false
11    },
12     id_livre: {
13       type: DataTypes.INTEGER,
14      allowNull: false
15    }
16   },
17   {
18     timestamps: true,
19     createdAt: false,
20     updatedAt: false
21   })
22 }
23 }, {
24   timestamps: true,
25   createdAt: false,
26   updatedAt: false
27 })

```

```

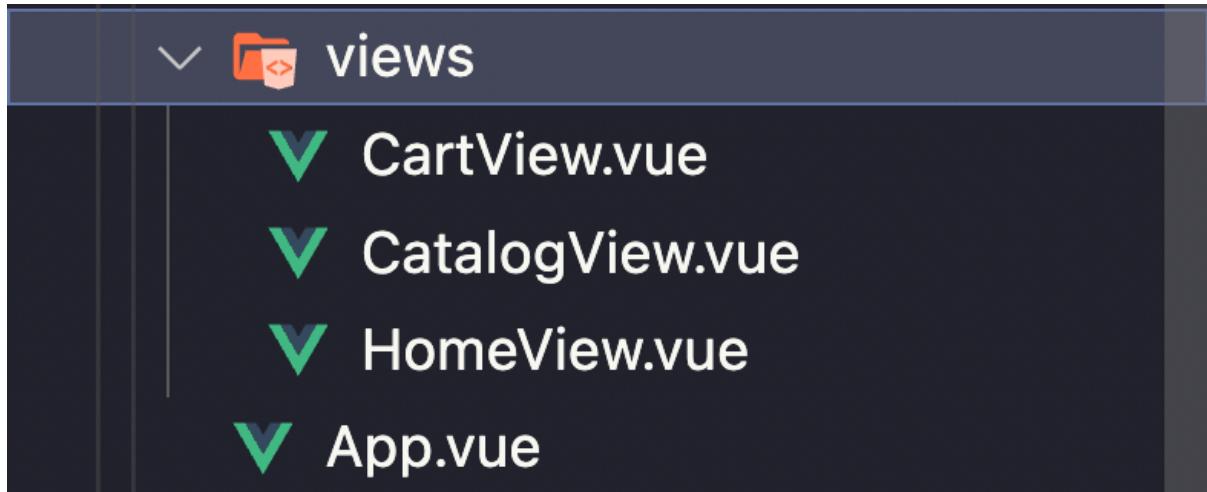
1 module.exports = (sequelize, DataTypes) => {
2   return sequelize.define('User', {
3     id: {
4       type: DataTypes.INTEGER,
5       primaryKey: true,
6       autoIncrement: true
7     },
8     pseudo: {
9       type: DataTypes.STRING,
10      allowNull: false,
11      unique: {
12        msg: 'This pseudo is already used'
13      }
14    },
15     password: {
16       type: DataTypes.STRING,
17       allowNull: false
18     },
19     admin: {
20       type: DataTypes.BOOLEAN,
21       allowNull: false
22     }
23   },
24   {
25     timestamps: false,
26     createdAt: false,
27     updatedAt: false
28   })
29 }
30 }, {
31   timestamps: false,
32   createdAt: false,
33   updatedAt: false
34 })

```

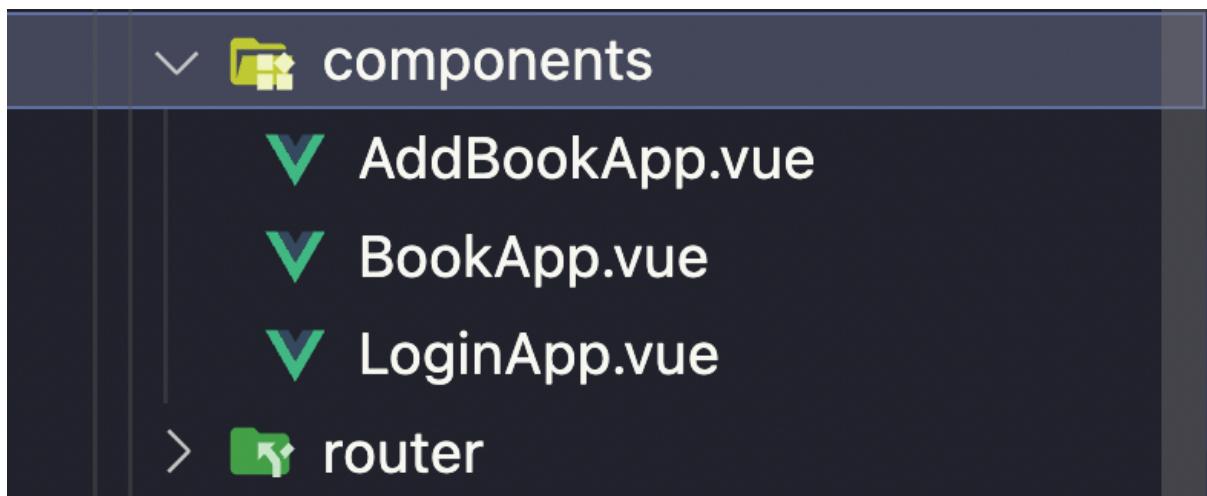
2) Le frontend

Nous avons utiliser vue cli afin de partir d'une structure.

Nous avons 3 viexs qui correspondent aux 3 pages de notre site.

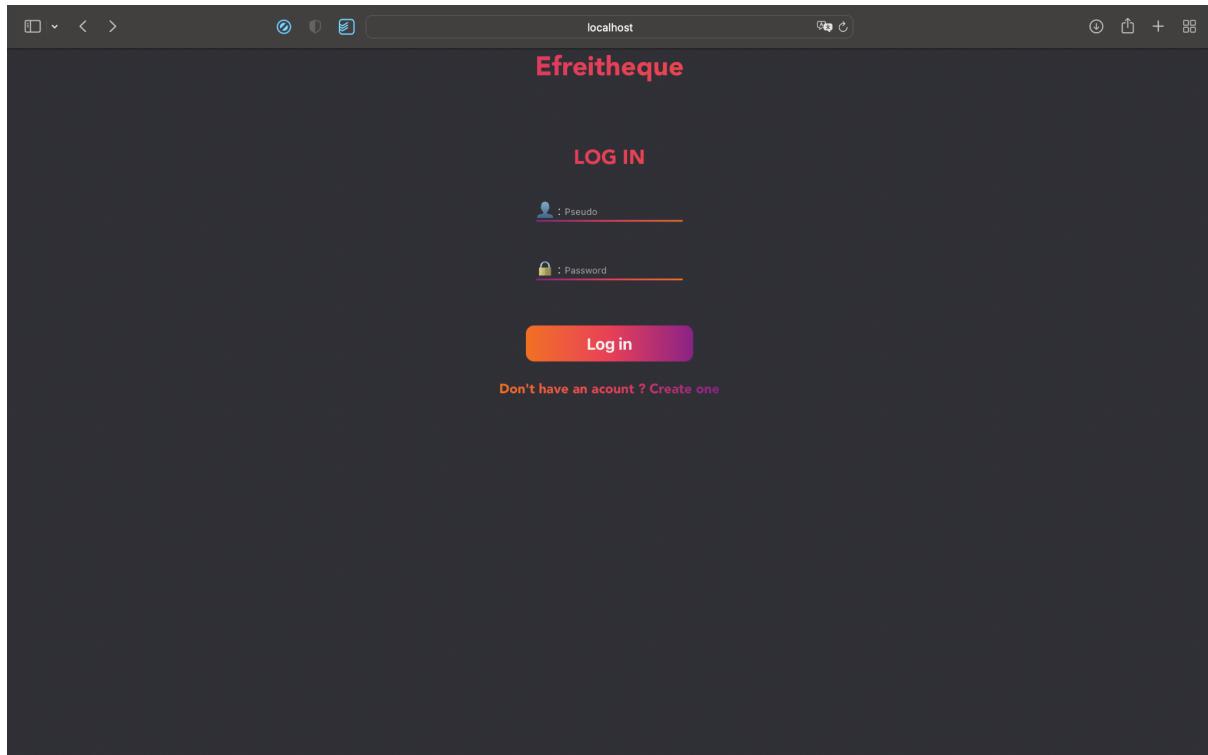


Et nous possédonns également 3 composants :

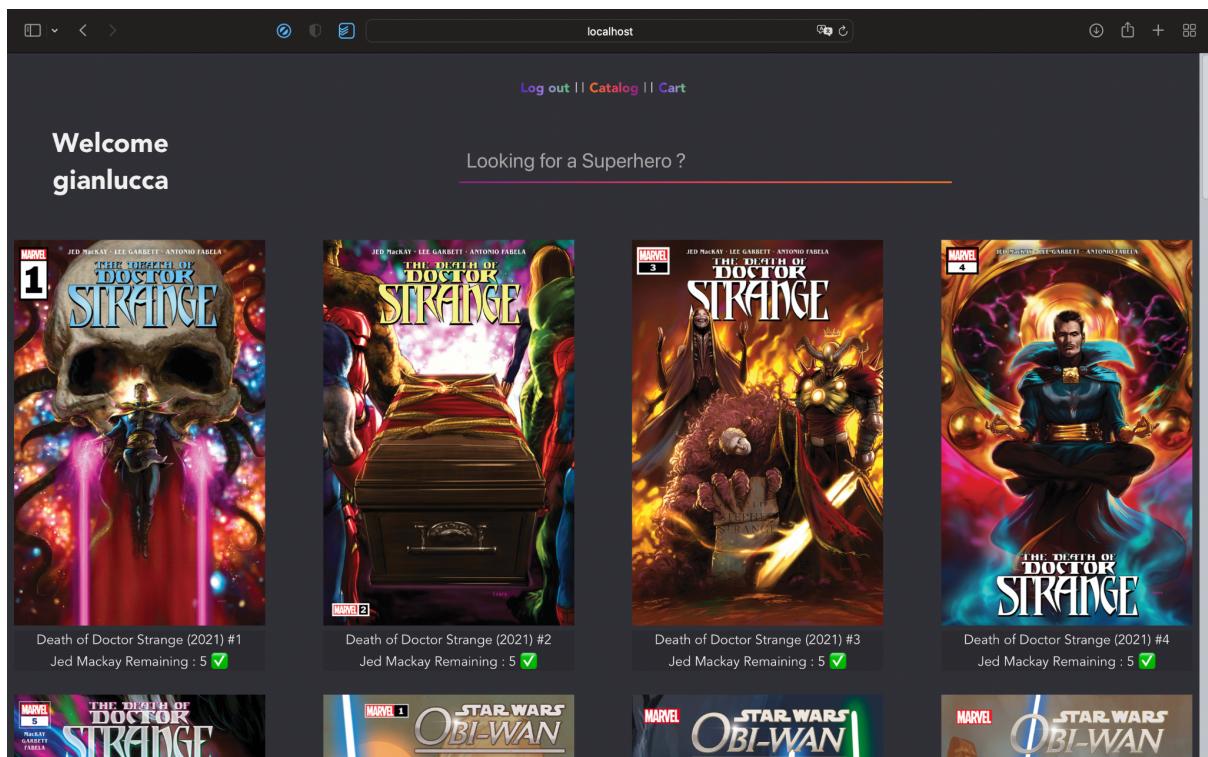


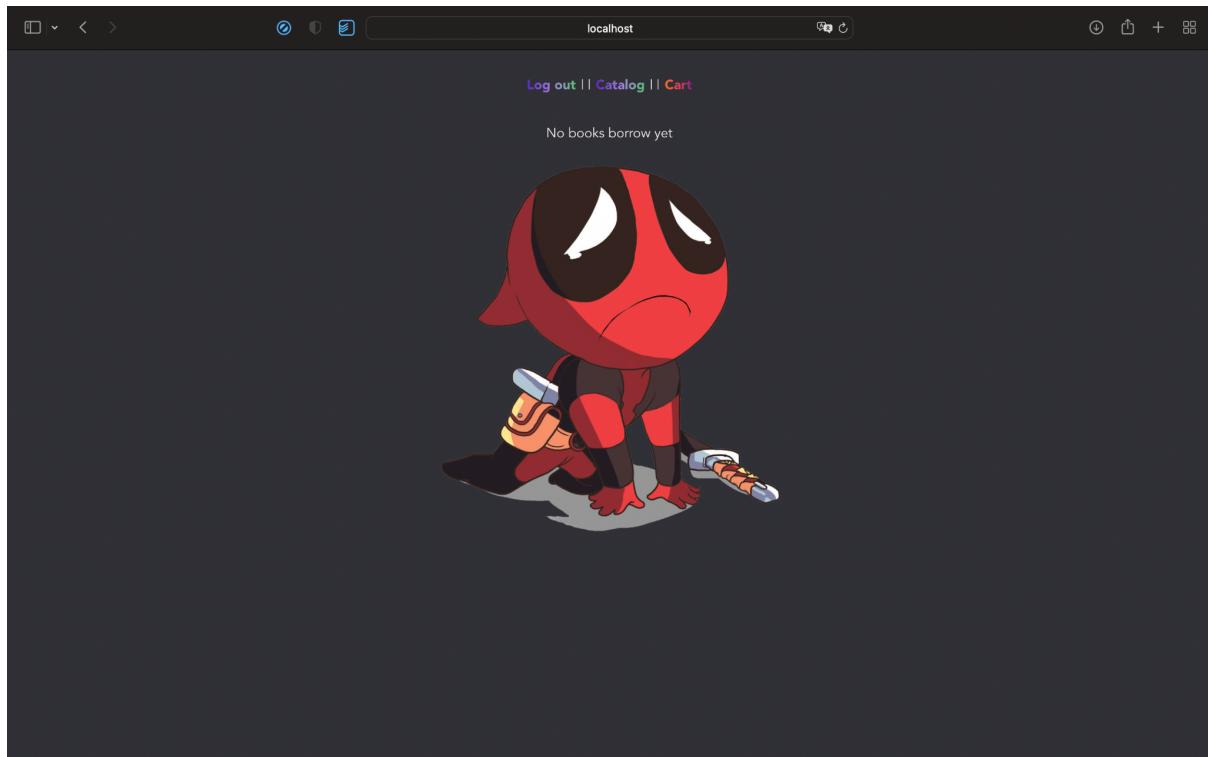
Vue JS est un framework très intéressant ! Il est à la fois puissant et intuitif nous sommes très content de nos connaissances acquises lors du développement du projet.

Voici un aperçu de nos views



Nous avons décidé de partir sur un thème de bibliothèque de comics



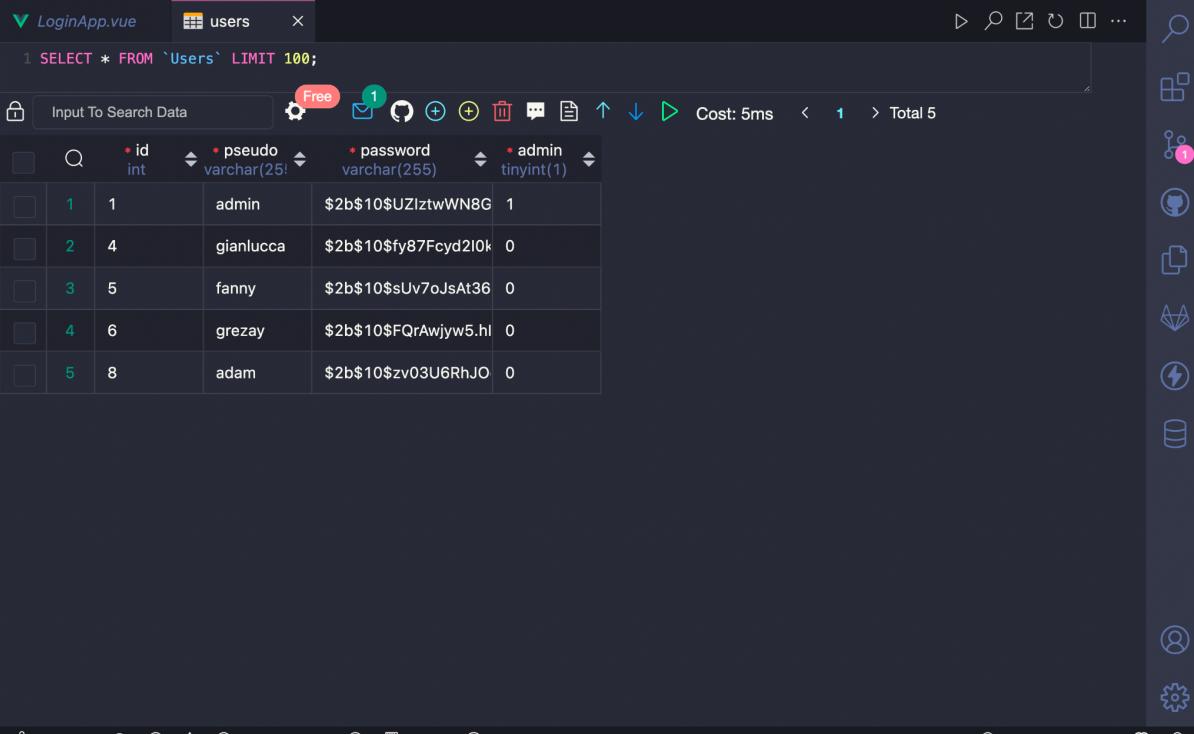


Nous allons voir en détail chaque fonctionnalité du site.

3) Les fonctionnalité

Tout d'abord la création d'un compte

A l'aide du binding et du module axios on récupère la saisie et si l'utilisateur n'est pas présent dans la base de donné USERS alors on peut l'ajouter. Le mot de passe est hasher grace a bcrypt et ensuite stocker dans la bdd, chaque user possede un id unique.

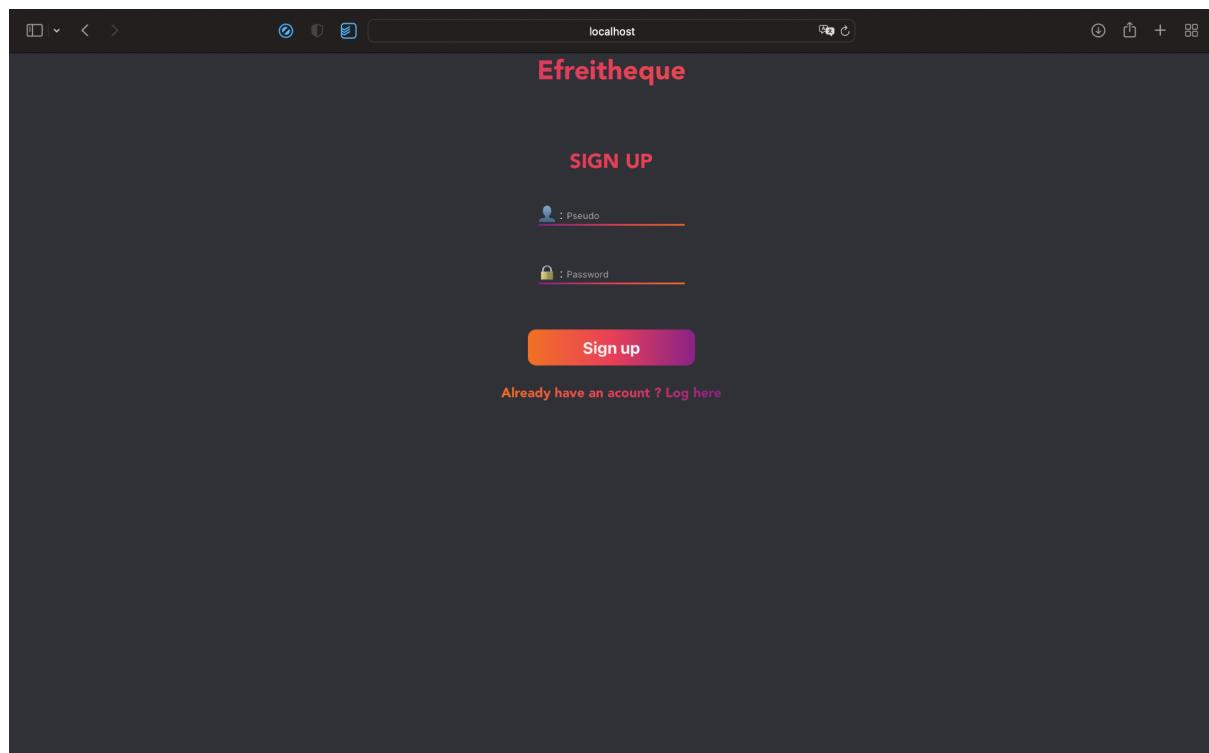


	* id int	* pseudo varchar(255)	* password varchar(255)	* admin tinyint(1)
	1	admin	\$2b\$10\$UZlztwWN8G	1
	2	gianlucca	\$2b\$10\$fy87Fcyd2iok	0
	3	fanny	\$2b\$10\$sUv7oJsAt36	0
	4	grezay	\$2b\$10\$FQrAwjyw5.hl	0
	5	adam	\$2b\$10\$zv03U6RhJO	0

Lors d'une création de compte il lui est automatique attribuer le statut d'étudiant par le admin = false.

```
1 router.post('/signup', (req, res) => {
2
3     const name = req.query.name
4     const mdp = req.query.mdp
5
6     bcrypt.hash(mdp, 10).then(hash => {
7
8         User.create({
9
10             pseudo: name,
11             password: hash,
12             admin: false
13
14         }).then(user => {
15             res.json({ message: 'Signup successful', user: user, sign : true })
16         })
17         .catch(err => {
18             res.json({ message: 'Signup failed', err: err, sign : false })
19         })
20     })
21
22 });


```



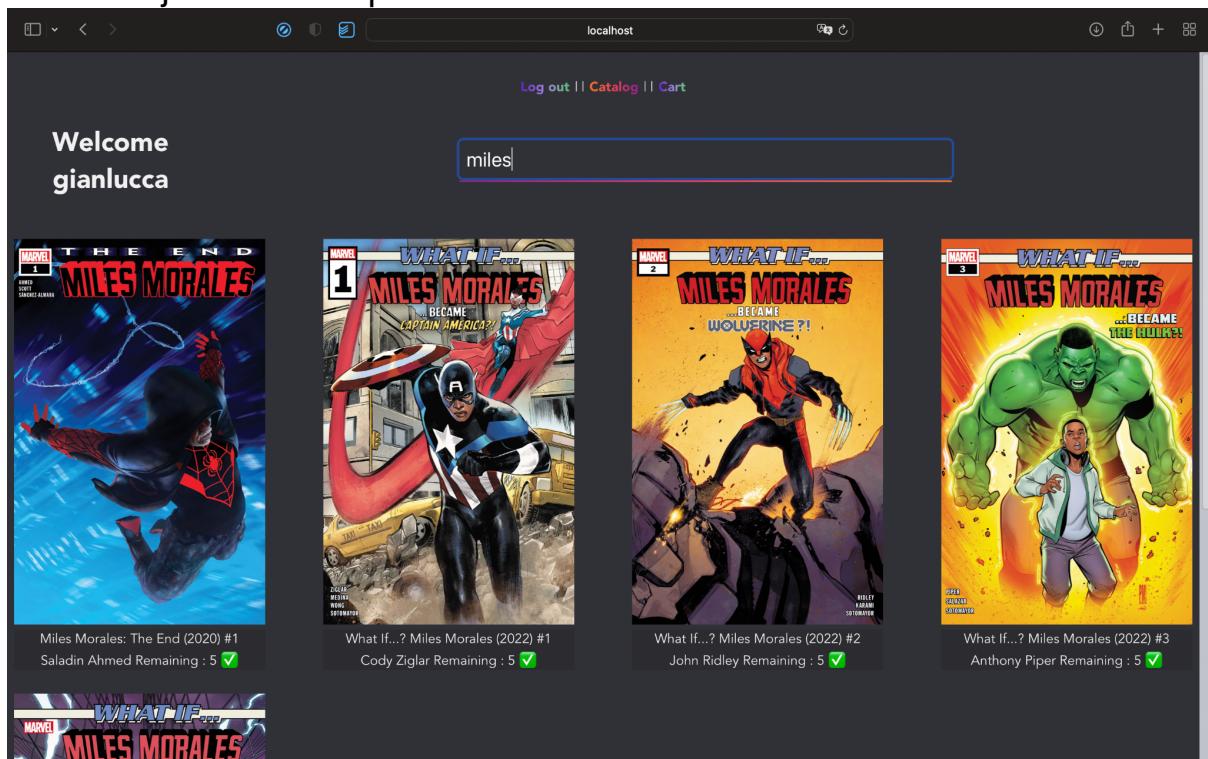
Passons maintenant au login et au JWT

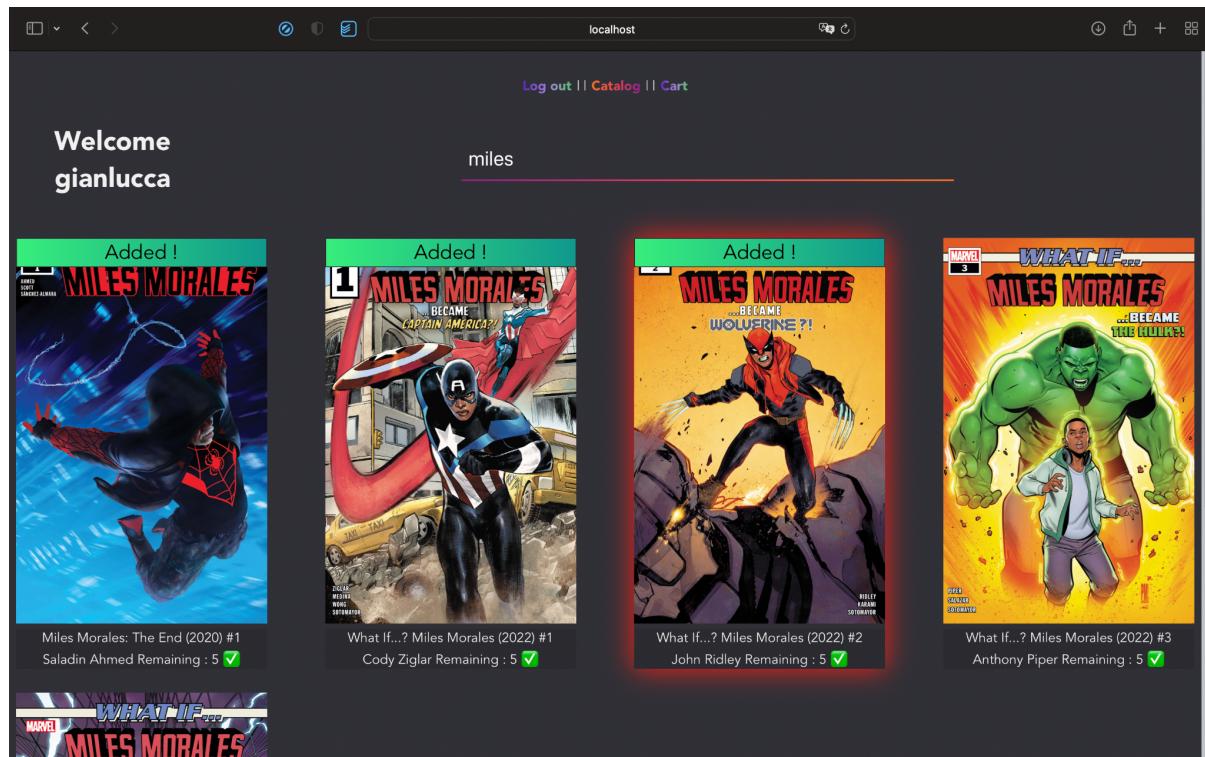
Nous avons fait pas mal de recherches afin d'implémenter le JWT directement. Le jetons sera généré au login de l'utilisateur et toutes les autres routes demanderont ce token stocker de le sessionStorage et récupérer par tous les composants.

```
● ● ●
1 router.post('/login', (req, res) => {
2
3     const name = req.query.name
4     const mdp = req.query.mdp
5
6
7     User.findOne({
8         where: {
9             pseudo: name
10        }
11    }).then(user => {
12
13     if (user) {
14
15         bcrypt.compare(mdp, user.password, (err, result) => {
16
17             if (result) {
18
19                 //JWT
20                 const token = jwt.sign({ id: user.id }, privateKey, { expiresIn: '5min' });
21
22                 res.json({ message: 'Login successful', user: user, auth: true, token: token })
23
24             } else {
25
26                 res.json({ message: 'Login failed -> wrong Password', auth: false })
27             }
28         })
29     }
30     else {
31         res.json({ message: 'Login failed -> Wrong Username', auth: false })
32     }
33 })
34
35 })
```

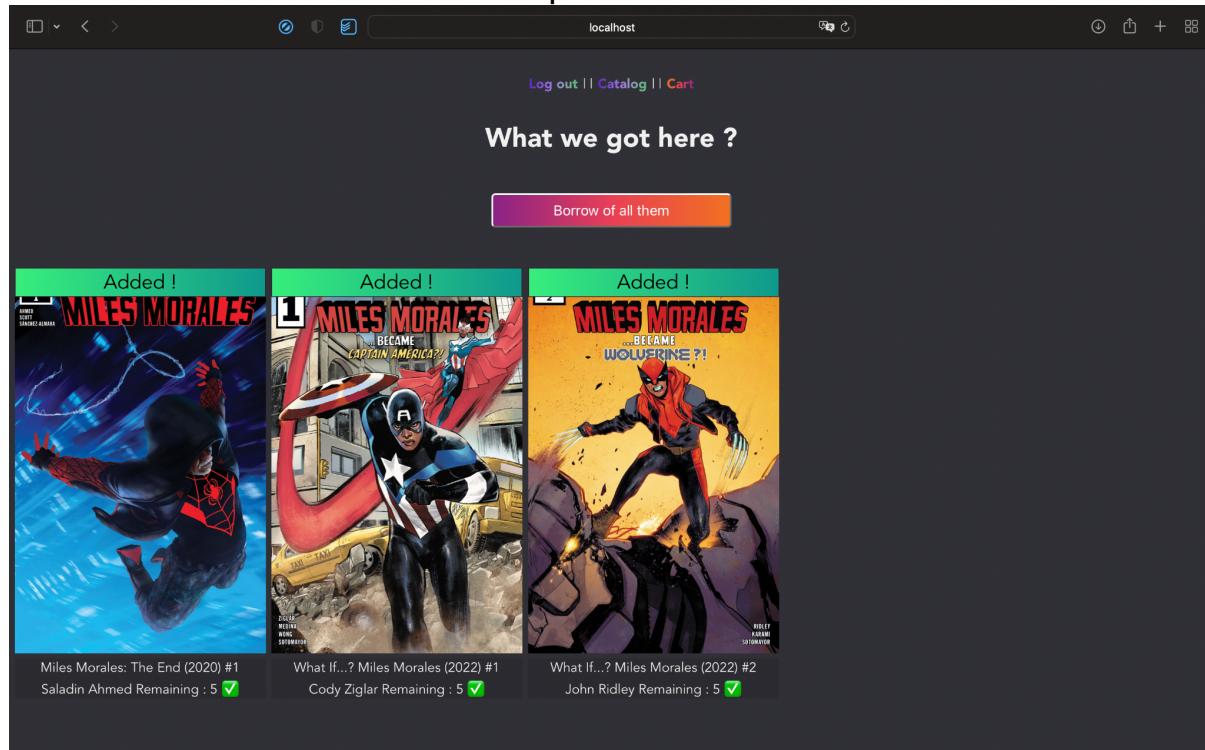
```
1 syncSession(){
2     this.session.userid = sessionStorage.getItem('userid');
3     this.session.username = sessionStorage.getItem('username');
4     this.session.token = sessionStorage.getItem('token');
5     this.session.admin = sessionStorage.getItem('admin');
6
7     if (this.session.admin === 'true') {
8         this.administrator = true;
9     }
10
11 },
```

Une fois que l'utilisateur a accès au catalogue, il peut rechercher un livre et l'ajouter à son panier.



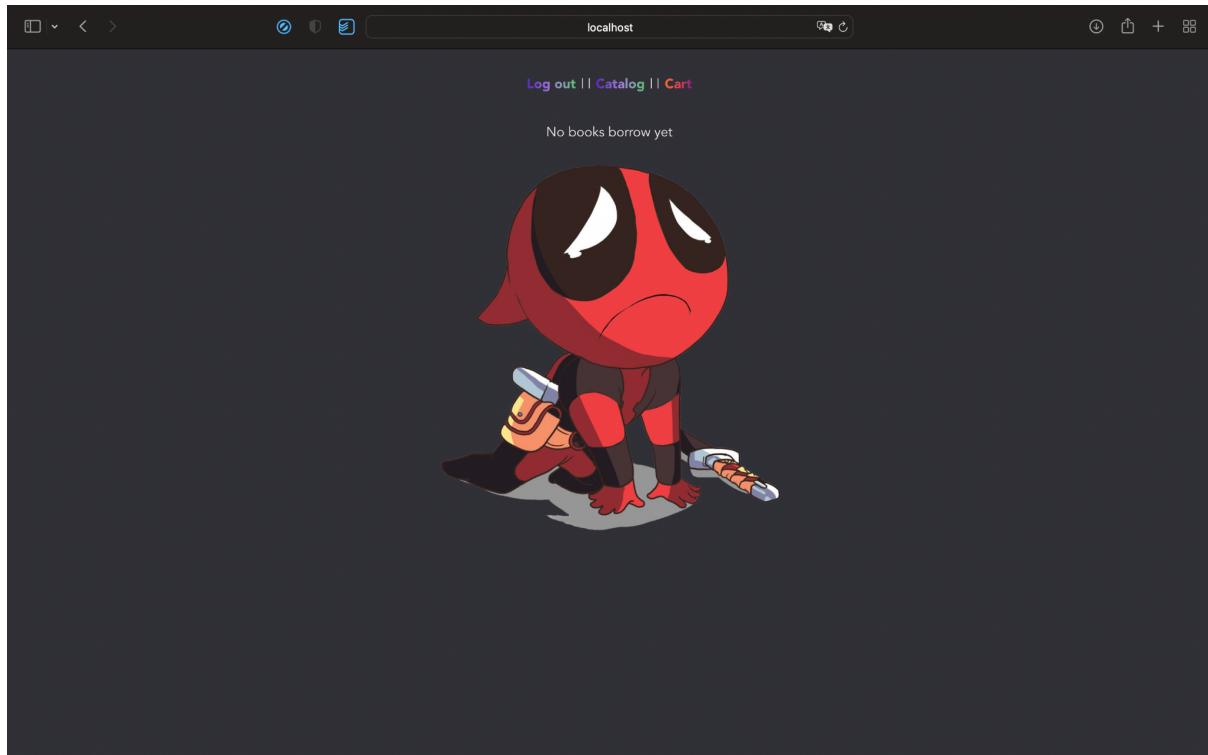


Et ils se retrouvent donc dans le panier



A savoir que la recherche est dynamique et que l'utilisateur ne peut emprunter que si le livre est disponible.

Pour supprimer un livre il lui suffit de cliquer dessus



Le composant livres est central dans le projet et nous avons su exploiter vue JS afin d'optimiser le site.

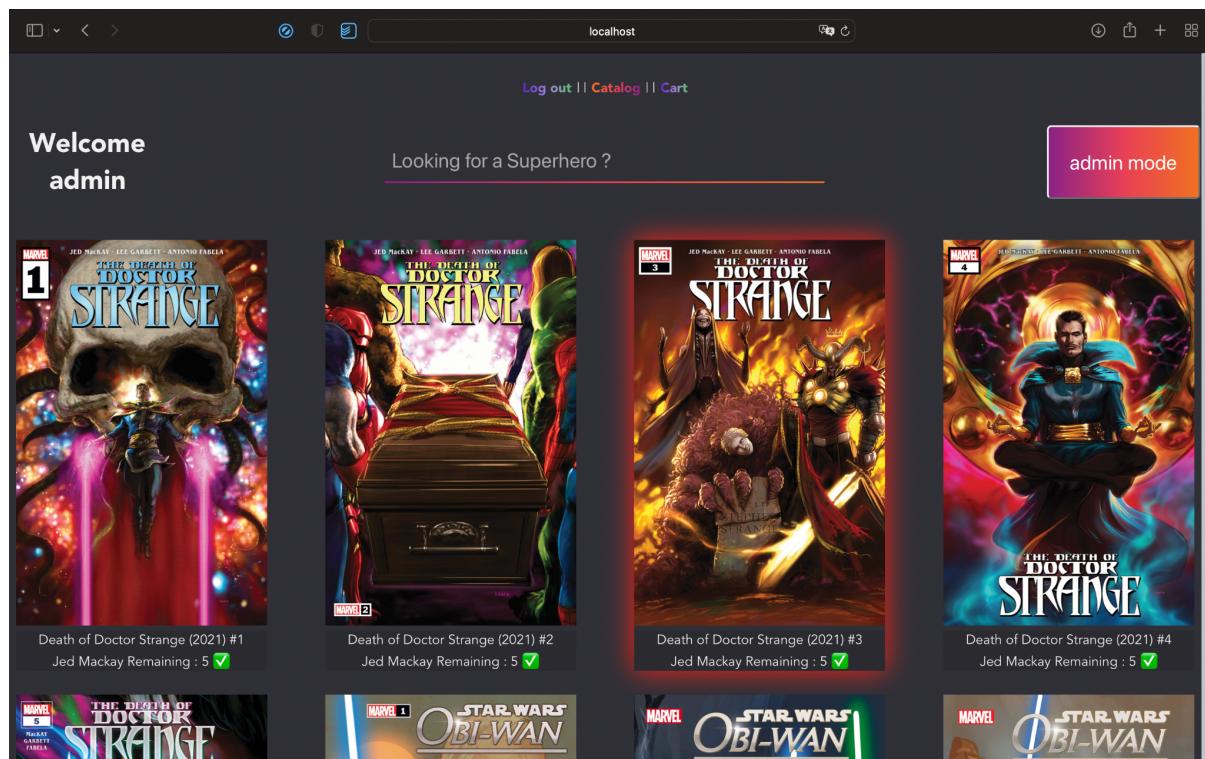
```
1 <div v-for="item in bibli" :key="item.id">
2
3   <div class="book">
4     <BookApp :book="item" @addbook ="addToCart($event)" @removebook ="removeFromCart($event)" @delbook="deleteBook($event)" :admin="admin" />
5   </div>
6
7 </div>
```

Avec V-for on affiche activement tous les livres , il nous as donc suffit de jouer sur la liste des livres à afficher pour nos différents besoins.

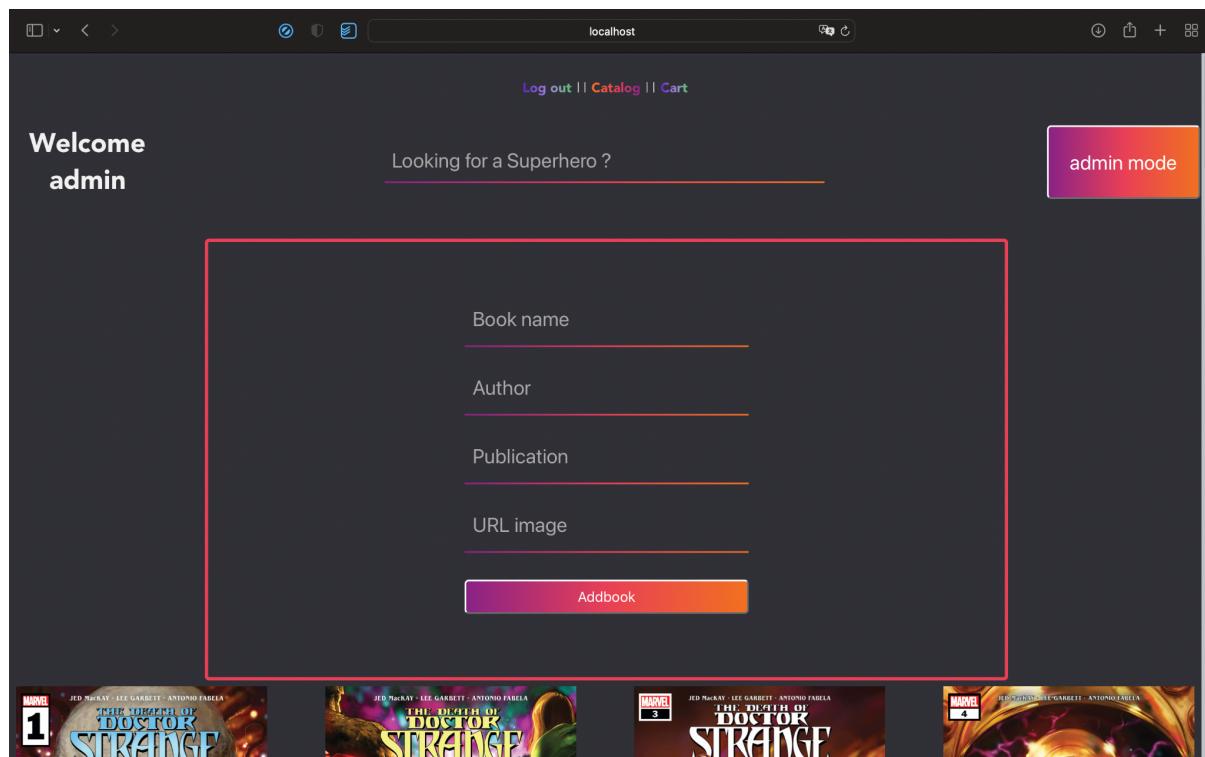
Les différents événements sont gérés dans le composant livre puis émis au different views.

L'utilisateur une fois les livres emprunter peut logout s'il le souhaite.

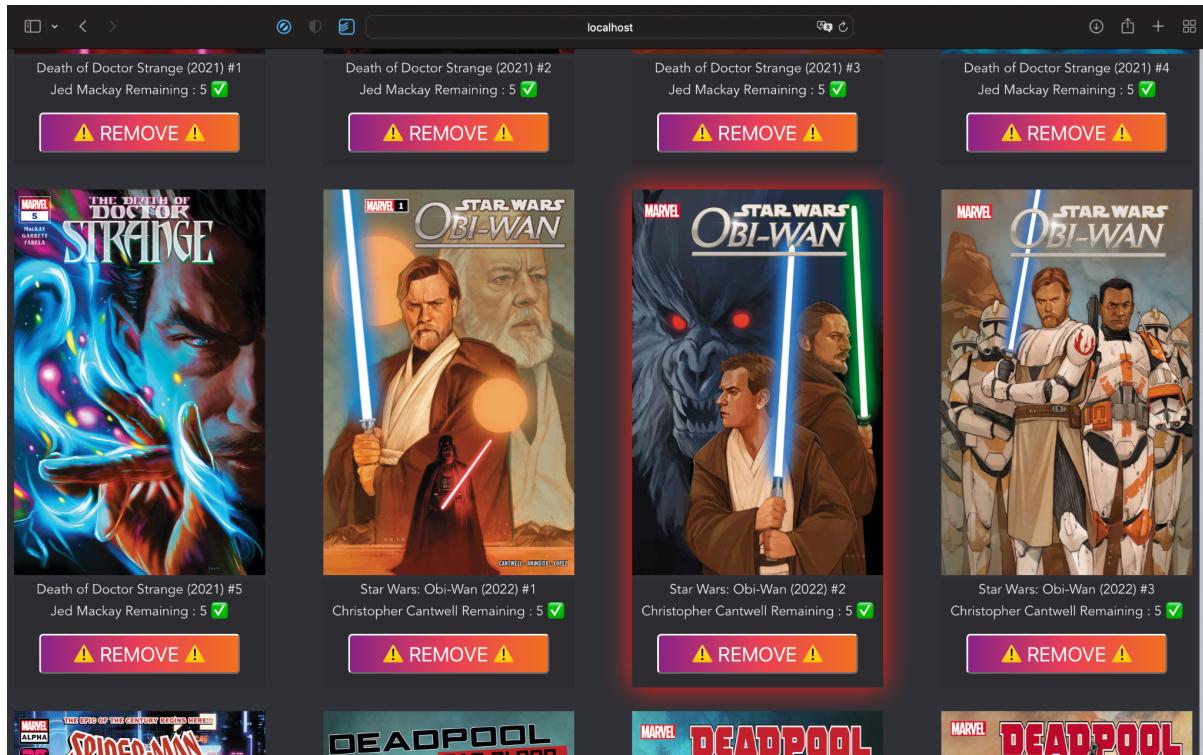
Pour finir intéressons nous au mode ADMIN



En utilisant le compte admin le bouton admin mode apparaît grâce à vshow



On peut ajouter un livre
Ou en supprimer un



Aucun rafraîchissement n'est requis car il est supprimé ou ajouté localement et synchronisé avec la BDD.

Conclusion

Nous sommes très contents de ce que nous avons accompli, nous sommes maintenant capable de créer un site de A-Z avec le backend. Les différents outils que nous avons apprivoisé font maintenant partie de notre bagage. Cela a été un projet très enrichissant ! Pour les axes d'améliorations nous avons pensé à :

- Mettre le site en ligne avec heroku ou github page
- Mettre un système de notation étoile
- Utiliser Vuex

Nous nous sommes beaucoup documentés sur le framework Vuex mais nous ne l'avons pas intégré. Mais aujourd'hui à la fin de ce projet nous nous sommes capable de l'intégrer dans nos future projets