React.js with PHP API REST

6. Composants Génériques et components as props

Le composant ListScreen.jsx

Ce composant va reprendre le code déjà écrit dans les 3 screens Home, Series et Techs
Pour la partie logique, seul l'url de la requête vers l'api pour récupérer les données va changer

src > components > screens > ListScreen.jsx > ListScreen

```jsx
import { useState, useEffect } from 'react';
import PropTypes from 'prop-types';
function ListScreen({apiUrl, title, Card}) {
    const [data, setData] = useState([]);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        const fetchData = async () => {
            try {
                const response = await fetch(apiUrl);
                if (!response.ok) {
                    throw new Error('Erreur de réseau');
                }
                const result = await response.json();
                console.log(result);
                setData(result || []);
            } catch (error) {
                console.log(error);
            } finally {
                setLoading(false);
            }
        };
        fetchData();
    }, []);

    return (
```

Pour la partie render, le titre va changer d'un screen à l'autre, ainsi que la Card utilisée pour afficher les données.

src > components > screens > ListScreen.jsx > ListScreen

```jsx
function ListScreen({apiUrl, title, Card}) {
    return (
        <main className="mt-5 pt-3 row">
            <h2>{title}</h2>
            {loading &&
                <p className='col-12 text-center'>
                    Chargement des données ...
                </p>}
            {(data.length > 0 )
                ? (data.map((item, i) => <Card key={i} data={item} />))
                : (!loading &&  (<p className='col-12 text-center'>
                                    Aucune donnée trouvée ...
                                </p>)
                )}
        </main>
    );
}
```

N’oubliez pas les propTypes

```
41  }
42  ListScreen.propTypes = {
43      apiUrl: PropTypes.string.isRequired,
44      title: PropTypes.string.isRequired,
45      Card: PropTypes.func.isRequired,
46  }
47  export default ListScreen;
```

Nous devons adapter les composants de type Card pour que la props data se retrouvent dans tous les composants.
Pour ArticleCard nous renommons la props article en data

```
src > components > cards > ArticleCard.jsx > ArticleCard
1   import PropTypes from "prop-types";
2   import Card from "./Card";
3   function ArticleCard(props) {
4       const { data } = props;
5       return (
6           <Card
7               title={data.title}
8               imgSrc={data.img_src}
9               link={"articles/details/" + data.id_article}
10              btnText="Lire"
11          >
12              <p className="card-text" style={{ textAlign: "justify" }}>
13                  {data.summary}
14              </p>
15          </Card>
16      );
17  }
18  ArticleCard.propTypes = {
19      data: PropTypes.shape({
20          id_article: PropTypes.number.isRequired,
```

Même chose pour SerieCard (serie -> data) et pour TechCard (tech -> data)

Nous allons maintenant modifier HomeScreen pour utiliser ListScreen et lui passer ArticleCard en props (ligne 10)

```
src > screens > home > HomeScreen.jsx > HomeScreen
1   import "./HomeScreen.css";
2   import ListScreen from "../../components/screens/ListScreen";
3   import ArticleCard from "../../components/cards/ArticleCard";
4
5   function HomeScreen() {
6       return (
7           <ListScreen
8               apiUrl="http://api.php-blog-project.loc/article?orderby=published_at&sort=DESC&limit=12"
9               title="Accueil"
10              Card={ArticleCard}
11          />
12      );
13  }
14  export default HomeScreen;
15
```

Nous modifierons l'API Rest ensuite pour gérer les params que nous avons ajouté dans l'url d'appel à l'API

Résultat

Même principe pour SerieScreen

src > screens > series > SeriesScreen.jsx > SeriesScreen

```jsx
import SerieCard from '../../components/cards/SerieCard';
import ListScreen from '../../components/screens/ListScreen';

function SeriesScreen() {
    return (
        <ListScreen
            apiUrl="http://api.php-blog-project.loc/serie?orderby=title"
            title="Les séries"
            Card={SerieCard}
        />
    );
}
export default SeriesScreen;
```

Et pour TechsScreen

src > screens > techs > TechsScreen.jsx > TechsScreen

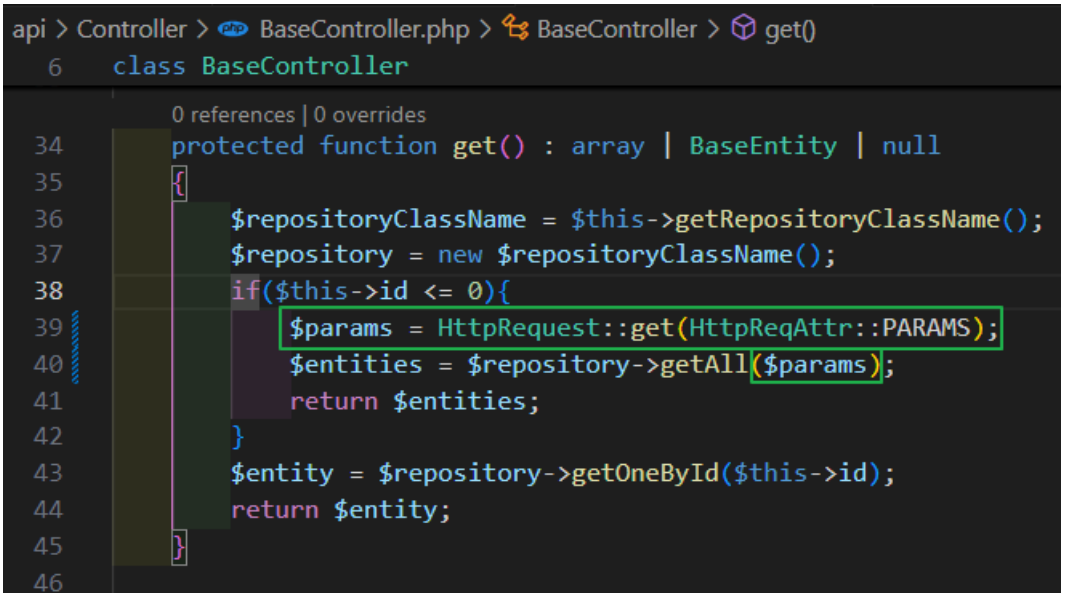
```jsx
import TechCard from '../../components/cards/TechCard';
import ListScreen from '../../components/screens/ListScreen';

function TechsScreen() {
    return (
        <ListScreen
            apiUrl="http://api.php-blog-project.loc/tech?orderby=label"
            title="Les Techs"
            Card={TechCard}
        />
    );
}
export default TechsScreen;
```

Ressources :
https://www.dhiwise.com/post/react-pass-component-as-prop-guide-to-component-composition#role-of-parent-and-child-components

Modification de l'API Rest

Pour prendre en compte les paramètres GET de l'url que nous avons ajoutés dans nos 3 screens (orderby, sort, limit) nous devons modifier légèrement notre API Rest en php.
Dans BaseController, méthode get(), nous récupérons les paramètres grâce à la classe HttpRequest et les passons à la méthode getAll du repository.

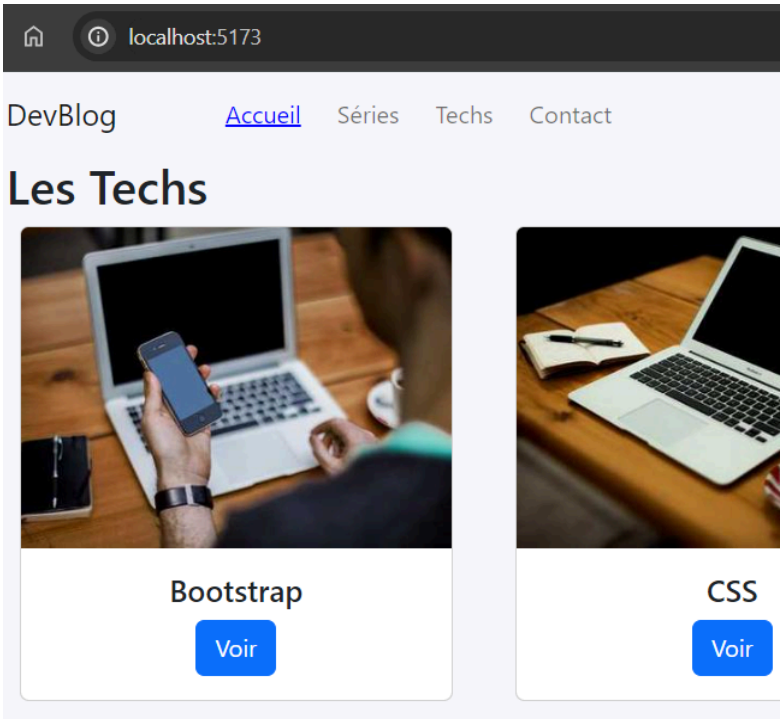


Puis nous modifions la requête sql dans la méthode getAll() de BaseRepository pour tenir compte des éventuels paramètres GET de l'url.

Résultat pour HomeScreen (Les 12 derniers articles publiés)

Pour SeriesScreen (par ordre alphabétique sur le titre)

Pour TechsScreen (par ordre alphabétique sur le label)



git : https://github.com/DWWM-23526/REACT_BLOG_PROJECT/tree/Step06
https://github.com/DWWM-23526/PHP_BLOG_PROJECT/tree/Step16-6