

React

1) Général.

Utilisation de vite, qui est un builder de React beaucoup moins volumineux que create-react-app. Ils installent le strict minimum pour faire tourner l'ensemble, autant en Vanilla JS qu'en TypeScript.

2) Material UI.

2.1) Thème.

Le thème utilisé ici est dans le fichier `appTheme.js` que l'on retrouve dans le dossier `utils` de la racine.

Il nous permet de définir le thème pour l'ensemble du projet. Tous les composants de type material ui auront la même couleur, cela va donner une uniformité à l'application.

On peut le pousser davantage, quasiment toutes les propriétés CSS sont paramétrables par défaut.

Cela facilite aussi le passage du mode clair au mode sombre.

2.2) Composants natifs.

Material Ui est fourni avec des composants génériques que l'on peut réutiliser selon notre convenance comme les boutons, les grille de données, les icônes, etc ...

Permet un développement rapide et uniforme de l'application mais on devient vite dépendant de celui-ci et de la manière dont il a été développé.

Même si les paramètres qu'on peut leur passer sont assez conséquents et qu'ils sont le plus génériques possibles, il peut y avoir des cas particuliers où l'on se retrouve bloqués.

2.3) Composants personnalisés.

Parfois un composant devient limité quant à l'utilisation que l'on a de celui-ci.

Pour cela il est possible de récupérer un composant natif et de l'englober dans un composant plus gros que l'on va personnaliser avec les paramètres qui nous intéressent et qui ne sont pas disponibles à l'origine. Le but est juste de gérer les paramètres en amont.

On peut prendre par exemple le composant Table
path: /components/Table/index.jsx

```
export default function Table({ rows, columns, headerButton }) {  
  const navigate = useNavigate();  
  const ref = useRef(null);  
  const { width } = useContainerDimension(ref);  
  let updatedCol = columns;  
  
  if (columns.length > 0 && width > 800) {  
    updatedCol = columns.map((col) => ({  
      ...col,  
      width: width / columns.length,  
    }));  
  }  
  
  return (  
    <Card style={{ height: '100%' }} ref={ref}>  
      <Box  
        sx={{  
          height: '10%',  
          display: 'flex',  
          alignItems: 'center',  
          padding: '0 10px',  
        }}>  
        <Button onClick={() =>  
          navigate(headerButton?.link)}  
          variant="contained"  
        >
```

```

        size="small"
        endIcon={<AddIcon />}
      >
      {headerButton?.title ?
headerButton.title : 'Ajouter'}
      </Button>
    </Box>
    <Box sx={{ height: '90%' }}>
      <DataGrid
        rows={rows}
        columns={updatedCol}
        sortModel={[{ field: 'date',
sort: 'desc' }]}

        rowSelection={false}
        hideFooterSelectedRowCount
        headerClassName="custom-
header"
        initialState={{
          pagination: {

paginationModel: { page: 0, pageSize: 10 },
          },
        }}
        pageSizeOptions={[10, 50,
100]}
      />
    </Box>
  </Card>
);
}

```

Le but ici était de créer un composant Table générique utilisable dans toute l'application, il prend les props :

- **rows** : data
- **columns** : liste d'objets contenant un field (nom de la données en reçu lors de l'appel API), headerName (nom de la donnée tel

qu'on veut l'afficher)

- **headerButton** : objet contenant title (Titre à afficher dans le boutton) link (lien vers la page de création).

Exemple d'utilisation:

```
<Table
  rows={rows}

  columns={columns}

  headerButton={{

    title: 'Ajouter un événement',

    link: '/events/create',


  }}
/>
```

L'utilisation ici du hook `useContainerDimension` permet de récupérer la dimension du container spécifié grâce au `ref` et de mettre à jour la taille des colonnes en fonction du nombre de celle-ci et de la largeur du container contenant le tableau.

3) React-router-dom.

Module populaire dans la gestion du router et des pages dans l'application.

On gère grâce à celui-ci la protection des routes et des différentes pages et donc l'authentification.

Le fichier `index.js` du dossier `router` est le gestionnaire principal, c'est ici que l'on répertorie les routes de l'application, lorsque l'url est reconnu, il rend le composant correspondant.

Si l'on essaie d'aller sur une route qui n'est pas répertoriée ici, on se retrouve sur la page 404.

Le composant récupère le composant rendu par le router et vérifie que l'utilisateur est bien connecté pour rejoindre la route, sinon il redirige vers la page `/login`.

Schema:

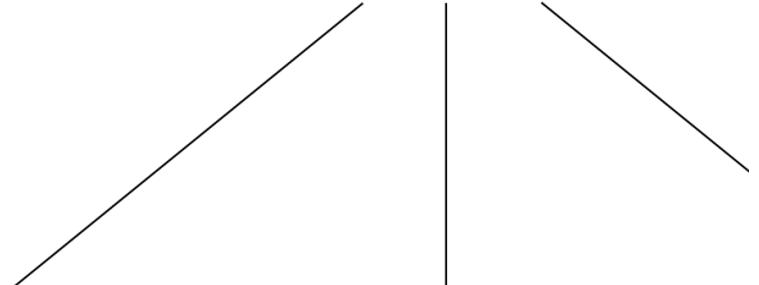
Router



Back office

Settings

Events



4) Redux.

Redux est un gestionnaire de state globalisé. Ils sont enregistré dans un store qui est distribué là où on le souhaite et de les récupérer dans n'importe quel composant qui sont dans le scope du store en question. Cela permet d'éviter ce qu'on appelle le 'props drilling'.

Ici on définit le store à la racine du projet dans le main. Il nous permet de gérer l'authentification, l'utilisateur, les notifications (les confirmations de créations, afficher les erreurs, etc) et d'enregistrer les infos de l'utilisateurs.

Back office

I - Installation de react.

L'installation se fait en même temps que les autres packages `npm run install:all` ou `yarn install:all`

Rentrer l'url de votre serveur web dans le fichier `backendUrl.js`

II - Lancement du backoffice.

Afin de lancer le backoffice, il suffit de lancer dans le terminal la commande `npm run start:front` tout en étant sur que le serveur web tourne correctement.

III - Connexion au backoffice.

Rentrer les identifiants de connexion en tant que gérant de société aussi associé au compte admin.



Login

Email Address*

Password*



Remember me

LOGIN

[Forgot password?](#)

[Do not have an account? Register](#)

Copyright © Dedicate 2023.

IV - Création d'un événement.

DEDICATE

Création d'un événement
mardi 19 septembre 2023

JA Julien Armand
Gérant

ÉVÉNEMENTS

PARAMÈTRES

Nom de l'événement*

Lieu*

Date de l'événement

Genre

Description*

Prix*

Nombre de places*

CRÉER UN ÉVÉNEMENT

Certains champs sont obligatoires.

V - Modifications d'un événement.



mardi 19 septembre 2023


Julien Armand
Gérant

✉ **ÉVÉNEMENTS**
⚙️ **PARAMÈTRES**

Nom de l'événement*
Désintégration

Lieu*
Toulouse

Date de l'événement
12/09/2023 22:30 CALENDAR

Genre
rock ▼

Description*
Soirée de désintégration epitech msc-pro

Prix*
2

Nombre de places*
10

MODIFIER UN ÉVÉNEMENT

VI - Informations sur l'événement.



mardi 19 septembre 2023


Julien Armand
Gérant

✉ **ÉVÉNEMENTS**
⚙️ **PARAMÈTRES**

Désintégration

Date : mardi 12 septembre 2023

Type de musique : rock

Description : Soirée de désintégration epitech msc-pro

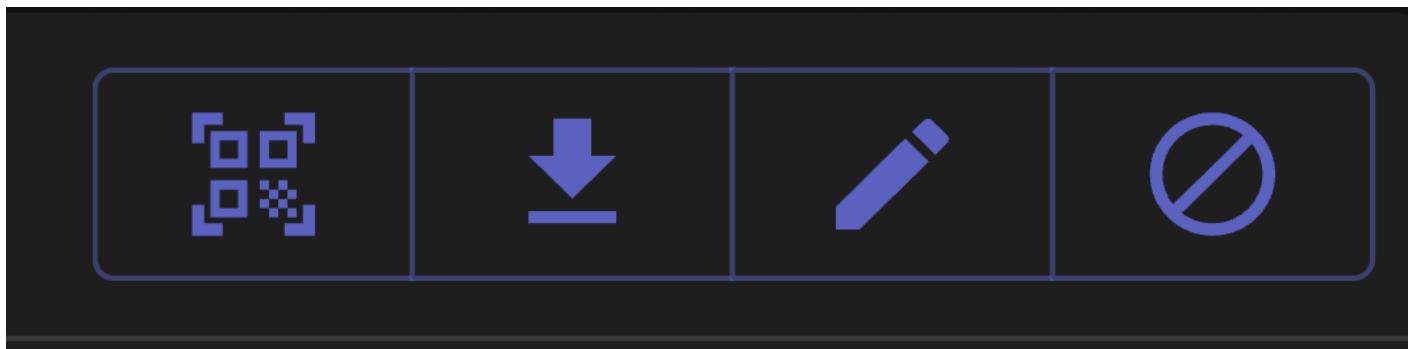
Prix minimum : 2

Nombre de musiques : 10

Lieu de l'événement : Toulouse

Playlist
Enchères

VII - Barre d'action sur un événement.



De droite à gauche :

- 1 - Permet d'afficher le Qr code de l'événement, c'est celui-ci qui permet d'y participer.
- 2 - Télécharger le Qr code.
- 3 - Modification de l'événement.
- 4 - Mettre fin à l'événement.