**Outils standards**

**Create React App**

Vous avez utilisé Create React App pour initialiser votre projet, ce qui est visible par la structure de votre projet et les fichiers générés automatiquement.

**React Router**

Vous utilisez React Router dans votre fichier App.js pour gérer les routes :

javascript

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

// ...

<Router>

<div className="App">

<Header />

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/flat/:id" element={<Flat />} />

<Route path="/about" element={<About />} />

<Route path="\*" element={<NotFound />} /> // Page 404

</Routes>

<Footer />

</div>

</Router>

**Contraintes techniques**

**Découpage en composants modulaires et réutilisables**

Voici comment vous avez structuré votre composant FlatCard dans un fichier séparé :

// src/components/FlatCard.js

import React from 'react';

import { Link } from 'react-router-dom';

import '../App.css';

const FlatCard = ({ id, title, cover }) => {

return (

<div className="flat-card">

<Link to={`/flat/${id}`}>

<img src={cover} alt={title} className="flat-cover" />

<div className="flat-title">{title}</div>

</Link>

</div>

);

}

export default FlatCard;

**Un composant par fichier**

Chaque composant est dans son propre fichier, par exemple :

* src/components/Header.js
* src/components/Footer.js
* src/components/FlatCard.js

**Structure logique des différents fichiers**

Votre projet a une structure logique :

lua

Copier le code

src/

|-- components/

| |-- Header.js

| |-- Footer.js

| |-- FlatCard.js

| |-- Flat.js

| |-- About.js

|-- App.js

|-- index.js

|-- App.scss

|-- index.scss

**Utilisation des props entre les composants**

Vous passez des props à vos composants, par exemple FlatCard reçoit id, title, et cover comme props :

javascript

Copier le code

// src/components/FlatCard.js

const FlatCard = ({ id, title, cover }) => {

return (

<div className="flat-card">

<Link to={`/flat/${id}`}>

<img src={cover} alt={title} className="flat-cover" />

<div className="flat-title">{title}</div>

</Link>

</div>

);

}

**Utilisation du state dans les composants quand c'est nécessaire**

Vous utilisez le state pour gérer des données dynamiques, par exemple dans Flat.js :

javascript

Copier le code

// src/components/Flat.js

import React, { useState, useEffect } from 'react';

import { useParams } from 'react-router-dom';

import axios from 'axios';

const Flat = () => {

const { id } = useParams();

const [flat, setFlat] = useState(null);

const [currentPhotoIndex, setCurrentPhotoIndex] = useState(0);

useEffect(() => {

axios.get('/flats.json')

.then(response => {

const flat = response.data.find(flat => flat.id === id);

setFlat(flat);

})

.catch(error => {

console.error("There was an error fetching the flat data!", error);

});

}, [id]);

**Gestion des événements**

Vous gérez des événements pour le carrousel des photos, par exemple :

javascript

Copier le code

// src/components/Flat.js

const handlePrevClick = () => {

setCurrentPhotoIndex((prevIndex) => (prevIndex - 1 + flat.pictures.length) % flat.pictures.length);

};

const handleNextClick = () => {

setCurrentPhotoIndex((prevIndex) => (prevIndex + 1) % flat.pictures.length);

};

**Utilisation de map pour itérer**

Vous utilisez map pour afficher les composants FlatCard :

javascript

Copier le code

// src/App.js

<Routes>

<Route path="/" element={

<div>

<div className="gallery">

{flats.map(flat => (

<FlatCard key={flat.id} id={flat.id} title={flat.title} cover={flat.cover} />

))}

</div>

</div>

} />

</Routes>

**Gestion des routes avec React Router**

Vous gérez les routes dans un fichier unique (App.js), et utilisez useParams pour récupérer les paramètres de l'URL :

javascript

Copier le code

// src/App.js

<Router>

<div className="App">

<Header />

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/flat/:id" element={<Flat />} />

<Route path="/about" element={<About />} />

<Route path="\*" element={<NotFound />} /> // Page 404

</Routes>

<Footer />

</div>

</Router>

**Conversion en Sass**

Voici comment convertir un fichier CSS en Sass. Par exemple, pour App.css, renommez-le en App.scss :

scss

Copier le code

// App.scss

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 20px;

background-color: white;

margin: 0 20px;

}

.logo img {

height: 50px;

}

nav ul {

list-style: none;

display: flex;

margin: 0;

padding: 0;

}

nav li {

margin-left: 20px;

position: relative;

}

nav a {

text-decoration: none;

color: black;

font-weight: bold;

}

nav a.active {

color: black;

}

nav a.active::after {

content: '';

position: absolute;

bottom: -2px;

left: 0;

width: 100%;

height: 2px;

background-color: rgb(4, 4, 4);

}

// Other styles...

N'oubliez pas d'installer node-sass et de mettre à jour les imports dans vos fichiers React :

bash

Copier le code

npm install node-sass

javascript

Copier le code

// src/App.js

import './App.scss';

**Vérification des Erreurs et Warnings**

Pour vérifier que votre code ne produit pas d'erreurs ou de warnings, exécutez votre application en mode développement et consultez la console du navigateur :