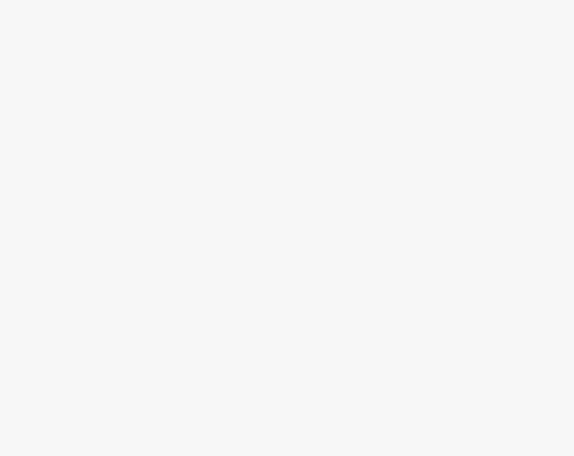


# Projet 11 OpenClassrooms

## Développez une application web avec React et React Router

### Le projet



Kasa est dans le métier de la location d'appartements entre particuliers depuis près de 10 ans maintenant. Avec plus de 500 annonces postées chaque jour Kasa fait partie des leaders de la location d'appartements entre particuliers en France.

Le site de Kasa a été codé il y a maintenant plus de 10 ans en ASP.NET avec un code legacy important.

Refonte total du site en React côté Front-end.

#### Outils standards

- Create React App
- React Router

● Styling: L'utilisation de Sass est optionnelle. CSS est à utiliser par défaut.

#### Contraintes techniques

##### React :

Il est impératif d'utiliser ces éléments de React pour un code de qualité :

- Découpage en composants modulaires et réutilisables ;
- Un composant par fichier ;
- Structure logique des différents fichiers ;
- Utilisation des props entre les composants ;
- Utilisation du state dans les composants quand c'est nécessaire ;
- Gestion des événements ;

● Listes : React permet de faire de choses vraiment intéressantes avec les listes, en itérant dessus, par exemple avec map. Il faut les utiliser autant que possible.

Il est également recommandé, mais pas imposé, d'utiliser des composants fonctionnels plutôt que des composants classes.

##### React Router :

- Les paramètres des routes sont gérés par React Router dans l'URL pour récupérer les informations de chaque logement.
- Il existe une page par route.

● La page 404 est renvoyée pour chaque route inexistante, ou si une valeur présente dans l'URL fait pas partie des données renseignées.

● La logique du routeur est réunie dans un seul fichier.

##### Général :

- Le code ne doit pas produire d'erreur ou de warning dans la console.

### COMPOSANTS GENERIQUES

HEADER  
(logo + nav)

FOOTER  
(logo + text)

GRID  
(Home→photos)

### COMPOSANTS MODULAIRES

COLLAPSE  
(titles + textes)

TAGS  
(text)

RATES  
(stars)

LESSOR  
(name + portrait)

PHOTO  
(banner + pictures)

GALLERY  
(include GALLERYIMG)

LOADER  
(spinner)

### UTILITAIRES

ROUTER

FETCHER  
(Simulator)

OBSERVER  
(DOM mutation)

GALLERYNAV  
(sensor touch screen for swipe img)

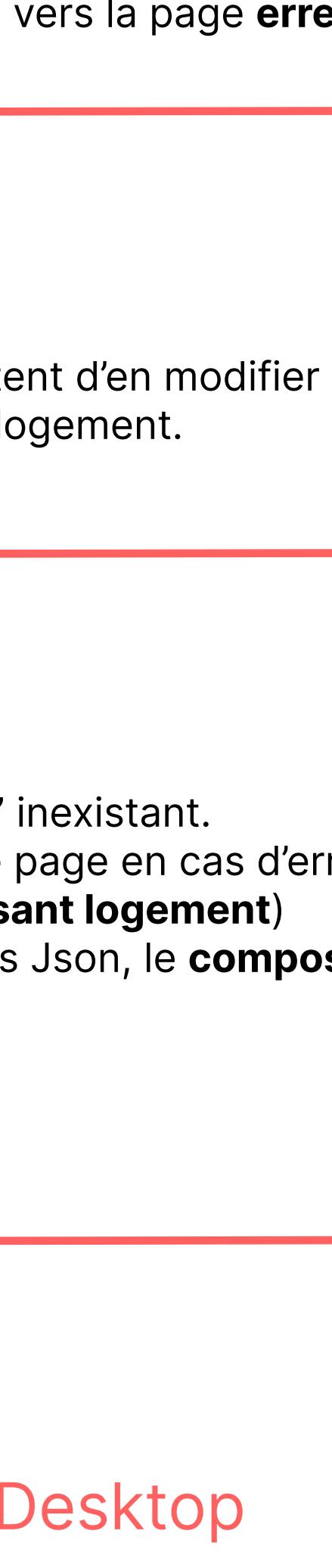
### La Structure

Initialisation de l'environnement par `create-react-app`, afin de développer le projet en tant qu'application React.  
Seul React et React-router-dom sont ici employés, comme stipulé, aucun autre librairie externe.  
Utilisation du Sass pour le visuel.

Une version desktop et mobile ont été créées.

Découpage des dossiers:

- **app** - fonctions
- **assets** - médias / logo
- **components** - composants
  - **pages** - pages
  - **styles** - SCSS + CSS
  - **utils** - context



### Le Router

Permet la centralisation de la navigation de l'application, `react-router-dom`.  
Les routes sont regroupées dans un fichier `router.jsx`

-Un `baseName` gère la mise en ligne de l'application (nomage vide d'arriver directement sur localhost:3000)  
-Les routes concernées par le Context sont englobées dans un Provider (ProviderLogements)

-`:idPage` est utilisé pour gérer les différents logements dans le composant Housing

-La route "\*" permet d'afficher la page erreur en cas d'url erronée

```
function Router() {
```

```
    return (
        <React.StrictMode>
            <BrowserRouter basename="">
                <ProviderLogements>
                    <div className="App">
                        <div className="content">
                            <Header />
                            <Routes>
                                <Route path="/" element={<Home />} />
                                <Route path="/logement/:idPage" element={<Logement />} />
                                <Route path="/about" element={<About />} />
                                <Route path="*" element={<Error />} />
                            </Routes>
                        </div>
                    </div>
                </ProviderLogements>
            </BrowserRouter>
        </React.StrictMode>
    );
}
```

```
export default Router;
```

---

### Fonctionnalités

Comme recommandé, les **composants fonctionnels** ont été utilisés sur ce projet.

De plus en plus utilisé

Les images apparaissent après une animation de type **loader**.

Ces images ainsi que les liens sont générées par la lecture du fichier **Json**, simulant une base de données.

#### page logement (housing)

La galerie d'image présente sous le header est bouclée, la dernière image ramène à la première, et vice-versa.

Implémentation d'une gestion des écrans tactiles, avec des paramètres de switch incorporés.

A l'ouverture d'une page logement, le **composant Collapse** (description + équipements) est fermé, un clic permet de l'ouvrir, un autre clic permet de le fermer. Ajout d'une animation de type **shaker** sur les petites flèches afin d'attirer l'oeil

En cas d'URL erroné, l'utilisateur sera dirigé vers la page **erreur**.

---

#### page à propos (about)

Présence du composant Collpase 4 fois.

Les propriétés (**props**) type et text permettent d'en modifier directement le contenu.

Fonctionnement identique que sur la page logement.

---

#### page erreur (error)

Cette page s'affiche en cas de route ou "id" inexistant.

Géré par le Router, qui redirigera vers cette page en cas d'erreur (mise à part des routes / logement/:idPage ⇒ gérées par le **composant logement**)

En cas d'erreur de chargement des données Json, le **composant logement** redirigera vers la page erreur.

---

### Rendus

#### Desktop



Accueil A Propos



#### Mobile



ACCUEIL A PROPOS



