

P11 - Développez une application Web avec React et React Router



par Thomas RANQUE



Le projet

Kasa est dans la location d'appartements entre particuliers depuis près de 10 ans. Son site a été codé en ASP.NET et commence à dater.

Demande : Refonte du site avec React en front et NodeJS en back. La première étape concerne le front-end.

La structure du projet

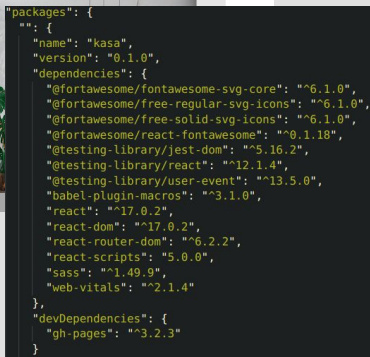
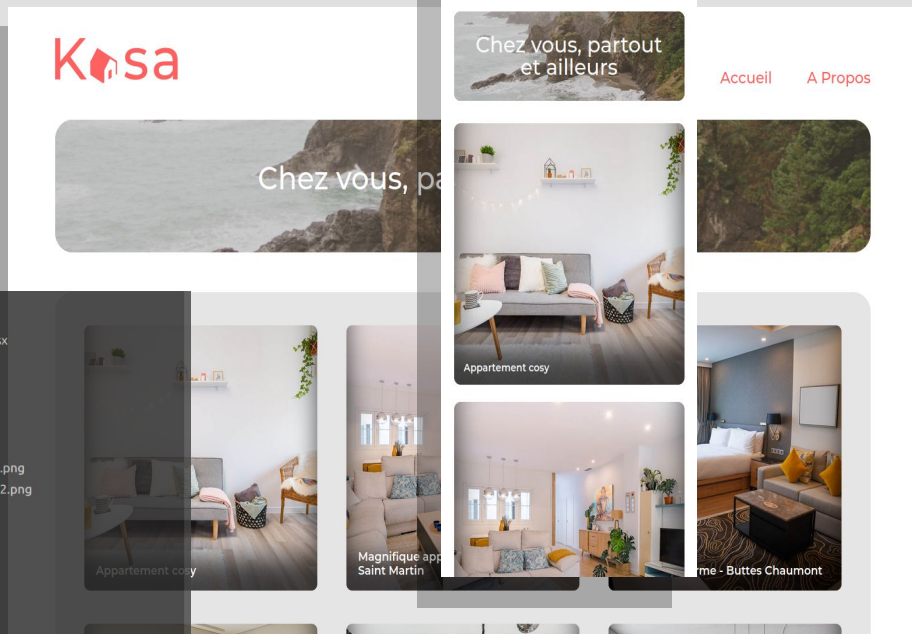
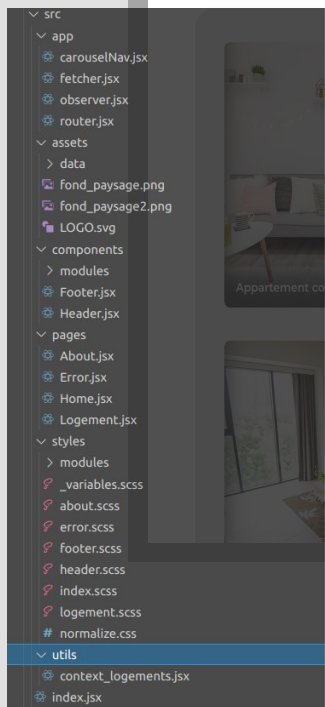
Le projet est développé en tant qu'application React en faisant usage de l'environnement fourni par la commande *create-react-app*.

Seuls **react** et **react-router-dom** sont employés, aucune autre librairie **react** n'est ajoutée. A noter l'emploi de **fontAwesome** et **Sass**.

Les versions **Desktop** et **Mobil** ont été développées.

La structure des fichiers de travail suit une logique claire:

- **app** – fonctions
- **assets** – médias
- **components** – composants
- **pages** – pages
- **styles** – scss & css
- **utils** – context



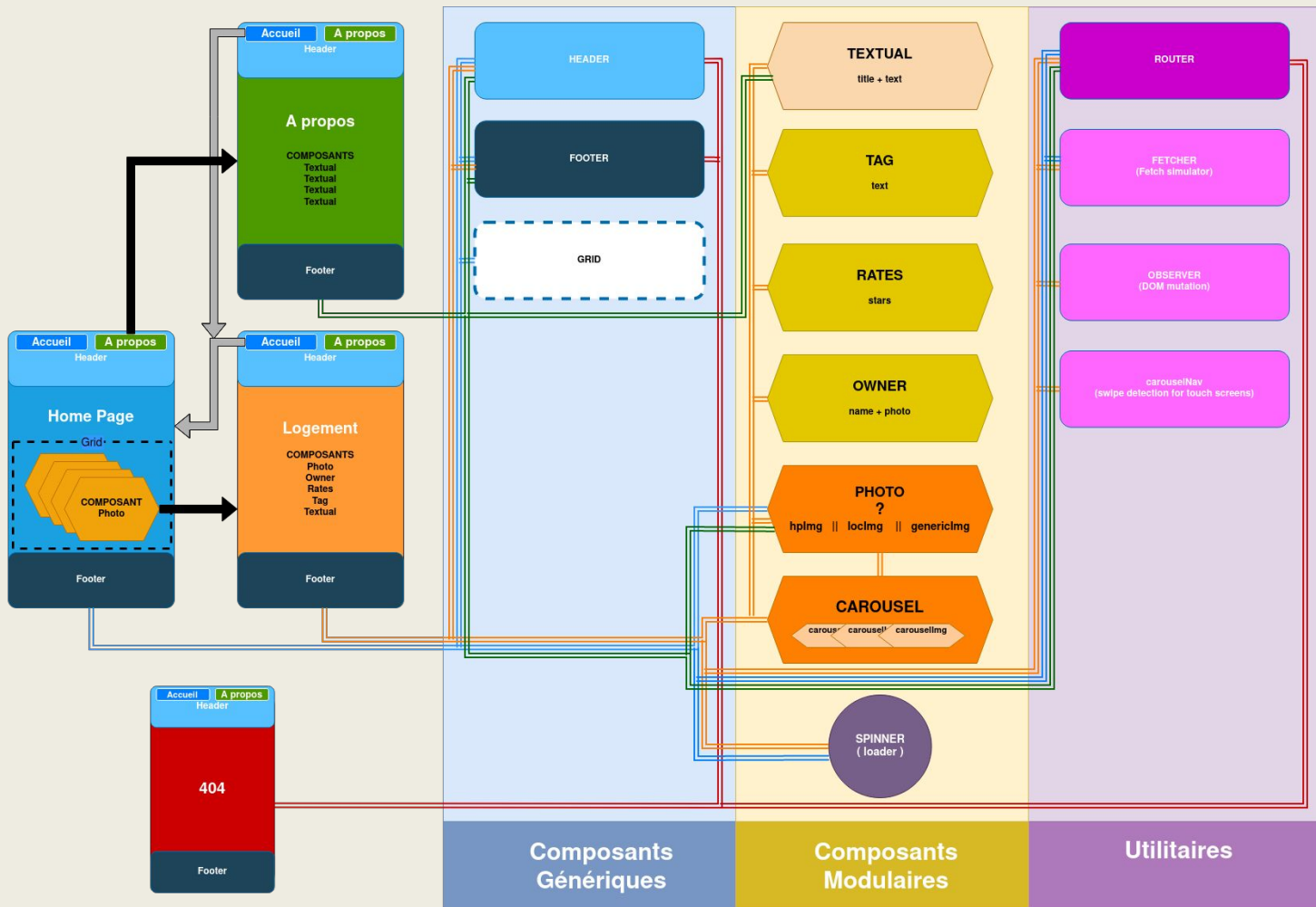
Algorithme

Relations entre composants

Les pages, puis chaque fonctionnalité de chaque page, sont décomposés en Composants simples.

Cet algorithme présente les relations qui les lient.

PAGES



Le Router

centralisation de la navigation

Afin de simplifier la gestion des **Routes** qui orchestrent la navigation de l'application, **react-router-dom** a été utilisé.

Un fichier unique (*router.jsx*) regroupe l'ensemble des routes existantes.

Plusieurs points sont à noter:

Un **baseName** est utilisé pour gérer la mise en ligne de l'application.

Un **Provider** englobe les routes concernées par le **Context**.

Routes (v6) remplace **Switch** (v5) pour englober les éléments **Route**.

Un **paramètre** :idPage est utilisé pour gérer les différentes locations dans un même composant **Logement**.

La **Route** "*" capture les URLs erronées.

```
2 import { BrowserRouter, Route, Routes } from "react-router-dom";
3
4 import Header from "../components/Header";
5 import Footer from "../components/Footer";
6 import Home from "../pages/Home";
7 import Logement from "../pages/Logement";
8 import About from "../pages/About";
9 import Error404 from "../pages/Error";
10 import { ProviderLogements } from "../utils/context_logements";
11 // import fetcher from "../app/fetcher"
12
13 function Router() {
14   // const allIDs = fetcher.get().map(loc => loc.id)
15
16   return (
17     <React.StrictMode>
18       <BrowserRouter className="container" basename="/P11_Kasa">
19         <ProviderLogements>
20           <div className="App">
21             <Header />
22             <Routes>
23               <Route path="/" element={<Home />} />
24               <Route path="/logement/:idPage" element={<Logement />} />
25               <Route path="/apropos" element={<About />} />
26               <Route path="*" element={<Error404 />} />
27             </Routes>
28           </div>
29         </ProviderLogements>
30         <Footer className="footer" />
31       </BrowserRouter>
32     </React.StrictMode>
33   );
34 }
35
36 export default Router;
```

Fonctionnalités

Home Page

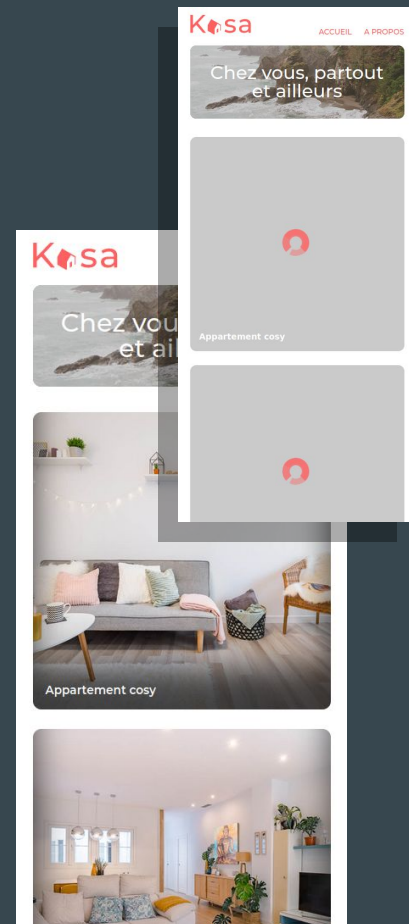
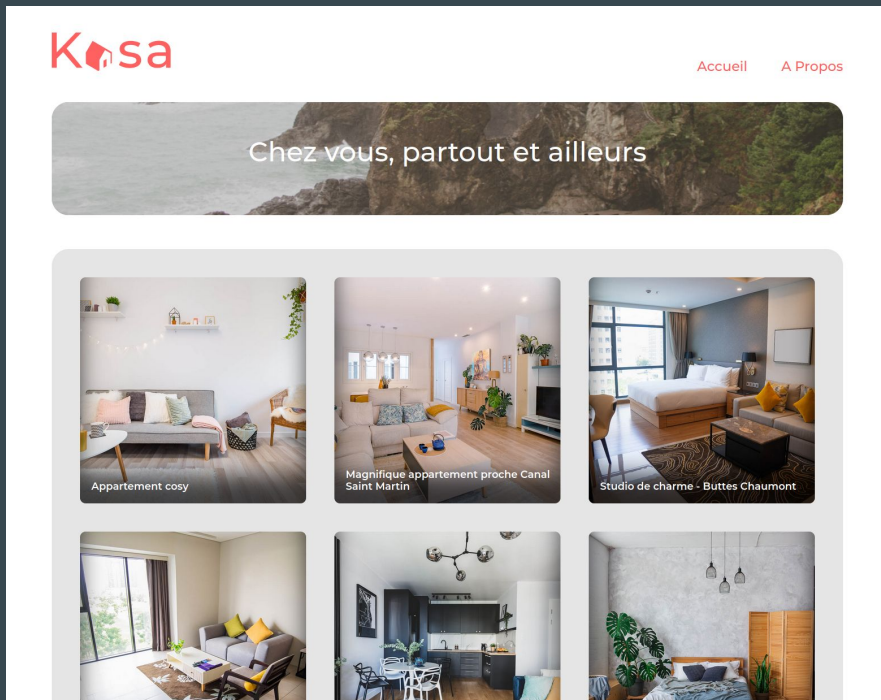
Les composants développés sur ce projet sont des **composants fonction**.

Le chargement des images est géré par l'affichage d'un **loader Spinner**.

Les images et les liens sont **générés à la volée** par la lecture du fichier JSON simulant la base de données.

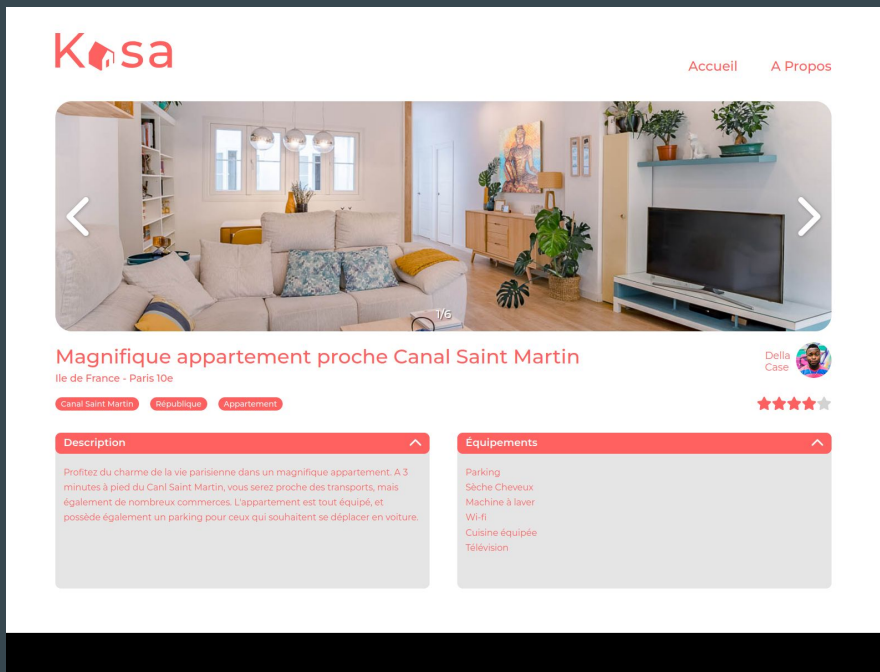
La maquette Figma est respectée et le site est **responsive**.

Des break-points sont définis à 964px pour les **tablettes** et 768 px pour les **mobiles**.



Fonctionnalités

Logement



Le **Carousel d'images** boucle de la dernière image à la première, et vice-versa.

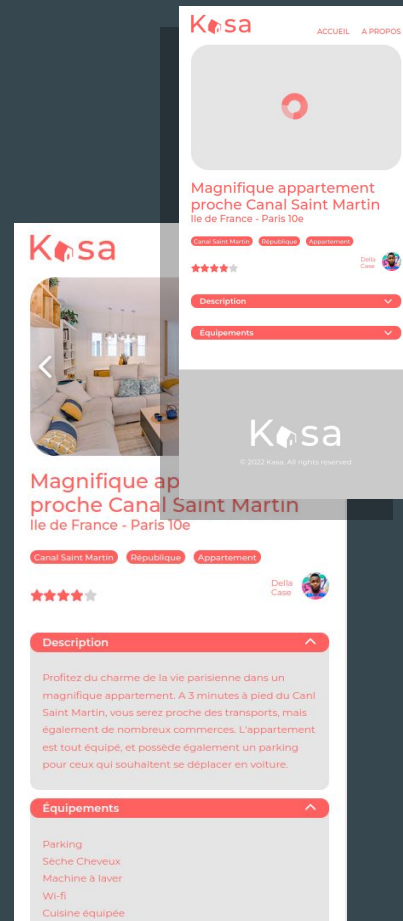
La gestion des écrans tactile est également implémenté.

Le chargement des images est géré par l'affichage du loader.

Le composant **Textual** (Collapse) est fermé au chargement de la page. Un click l'ouvre, le suivant le ferme.

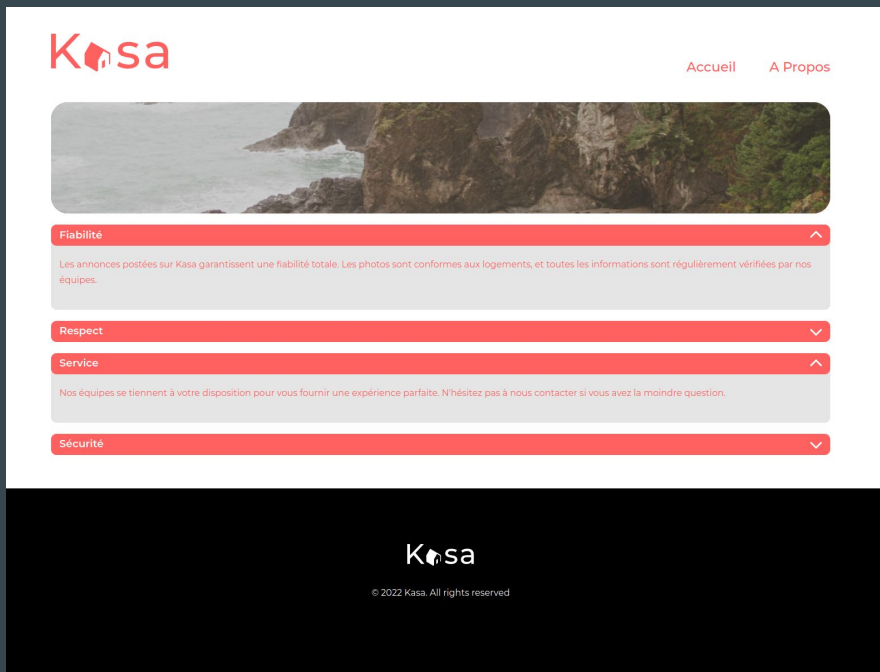
Les "id" erronés entrées par l'utilisateur dans l'URL déclenchent une **redirection** sur la page d'erreur.

L'ensemble des éléments du Figma sont en place.



Fonctionnalités

A propos



Le composant **Textual** est employé de nouveau quatre fois.

Les props **type** et **text** permettent d'en modifier le contenu.

Le fonctionnement reste le même.



Fonctionnalités

Erreur 404



Cette route est renvoyée pour **toute route inexistante**, qu'elle soit due à une erreur de code ou à une modification intentionnelle de l'utilisateur dans la barre d'adresse.

Le **Router** gère les routes erronées à l'exception des routes `/logement/:idPage` gérées par le composant **Logement**. Ce dernier renvoie à la page d'erreur en cas d'erreur de réception des données JSON.



Sass

La mise en forme est gérée par Sass.

_variables.scss (module)

Il gère l'importation de la police
GoogleFont Montserrat.

Il permet également la définition de
variables pour les couleurs
principales du projet.

box-sizing: border-box.

break-points à 964px & 768px.

```
index.scss
1 @import "../styles/normalize.css";
2 @import "../styles/variables";
3
4 *,
5 *::before,
6 *::after {
7   box-sizing: border-box;
8 }
9
10 body {
11   margin: 0;
12   font-family: $Montserrat, "Segoe UI", "Roboto", "Oxygen", "Ubuntu",
13     "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue", sans-serif;
14   font-weight: 400;
15   -webkit-font-smoothing: antialiased;
16   -moz-osx-font-smoothing: grayscale;
17 }
18
19 .Link {
20   color: $red;
21   text-decoration: none;
22   &:hover {
23     text-decoration: underline;
24   }
25 }
```

```
_variables.scss
1 @import url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,300;0,400;0,
2   500;0,600;1,300;1,400;1,500;1,600&display=swap');
3
4 // Font variables
5 $Montserrat: 'Montserrat', sans-serif;
6
7 // Colors variables
8 $red: #FF6060;
9 $greyl: #E5E5E5;
10 $greym: #cacaca;
```

```
37
38 @media (max-width: 964px) {
39   .App {
40     padding: 0 40px;
41   }
42 }
43
44 @media (max-width: 768px) {
```

GitHub Pages

Le projet est hébergé sur Github, sur deux branches :

- **main** – les fichiers de développement.
- **gh_pages** – Les fichiers de production.

The screenshot displays the GitHub repository interface for 'Peanuts-83/P11_Kasa'. The repository is public and has 2 branches and 0 tags. The file structure on the left includes a 'readme' folder with subfolders 'build', 'public', and 'src', and several files including '.gitignore', 'README.md', 'babel-plugin-macros.config.js', 'package-lock.json', 'package.json', 'reportWebVitals.js', and 'setupTests.js'. The 'README.md' file is selected, showing the title 'P11 - Kasa' and a description: 'Ce projet comporte 2 branches:'. It lists two branches: 'main' (Les fichiers de développement) and 'gh-pages' (La version de production). The 'gh-pages' branch is selected, showing a list of files and their update times. The files include 'static', '404.html', 'asset-manifest.json', 'favicon.ico', 'index.html', 'logo192.png', 'logo512.png', 'manifest.json', and 'robots.txt'. The 'About' section on the right provides information about the repository, including stars, watching, and forks. The 'Releases' and 'Packages' sections are also visible, showing no releases or packages published. The 'Environments' section shows the 'github-pages' environment as active. The 'Languages' section shows the file types: JavaScript (56.0%), SCSS (35.6%), and HTML (8.4%).

Getting Started with Create React App

This project was bootstrapped with [Create React App](#).