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

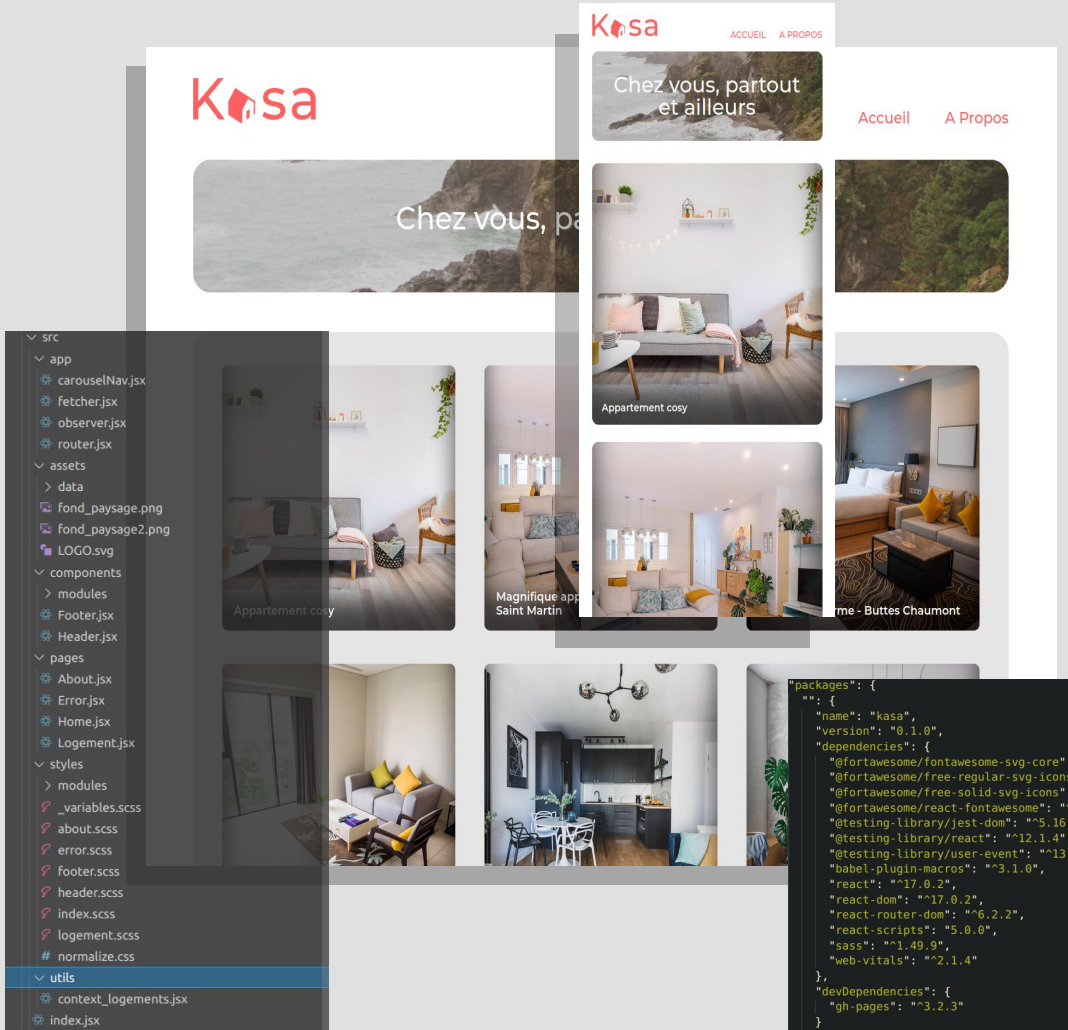
Le projet est développé en tant qu'application React en faisant usage de l'environnement fourni par la commande ***create-react-app***.

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:

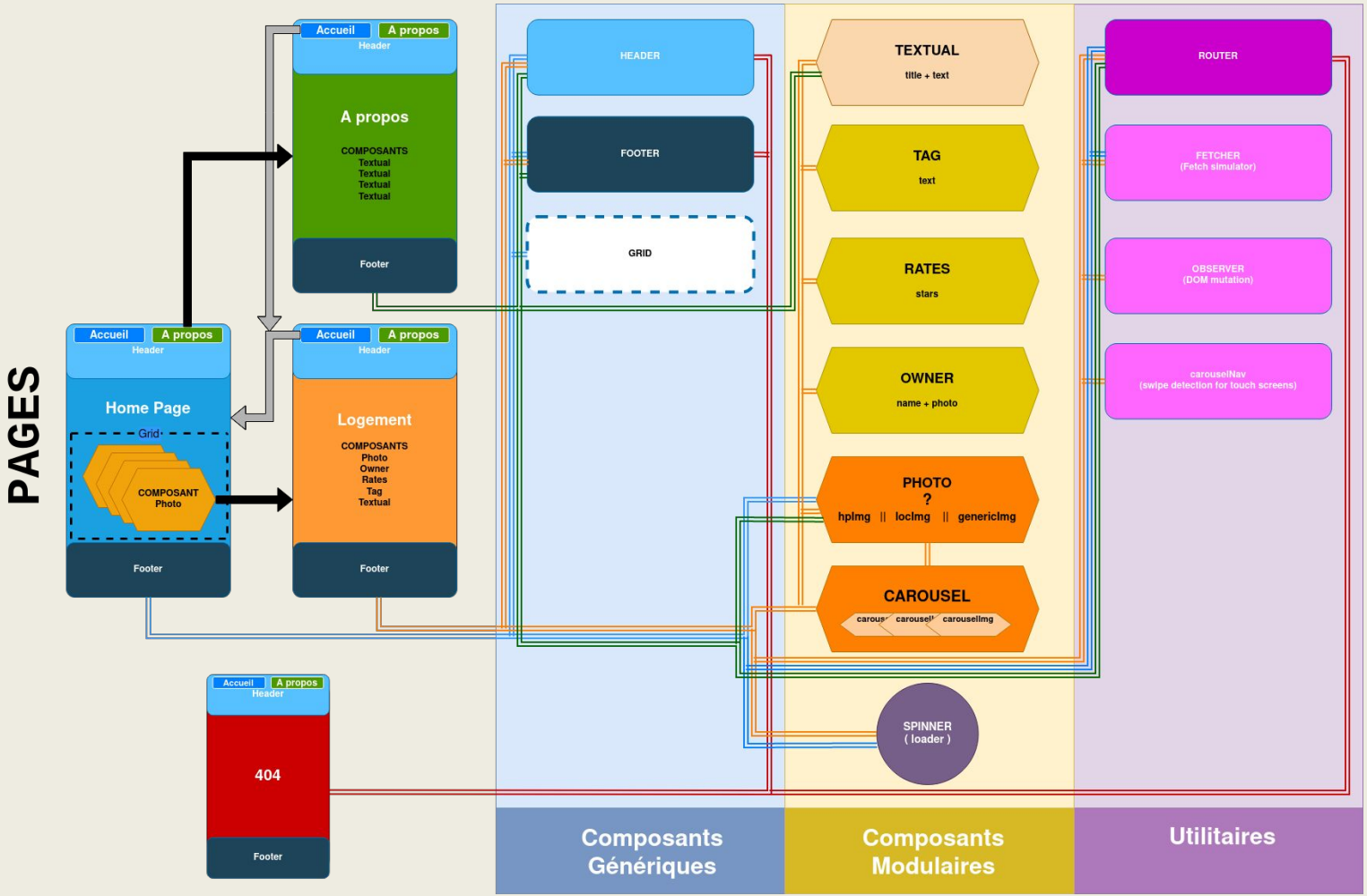
- **app** – fonctions
- **assets** – médias
- **components** – composants
- **pages** – pages
- **styles** – scss & css
- **utils** – context



Algorigramme
Relations entre composants

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

Cet algorigramme présente les relations qui les lient.



Le Router

centralisation de la navigation

```jsx
import { BrowserRouter, Route, Routes } from "react-router-dom";

import Header from "../components/Header";
import Footer from "../components/Footer";
import Home from "../pages/Home";
import Logement from "../pages/Logement";
import About from "../pages/About";
import Error404 from "../pages/Error";
import { ProviderLogements } from "../utils/context_logements";
// import fetcher from "../app/fetcher"

function Router() {
    // const allIDs = fetcher.get().map(loc => loc.id)

    return (
        <React.StrictMode>
            <BrowserRouter className="container" basename="/P11_Kasa">
                <ProviderLogements>
                    <div className="App">
                        <Header />
                        <Routes>
                            <Route path="/" element={<Home />} />
                            <Route path="/logement/:idPage" element={<Logement />} />
                            <Route path="/apropos" element={<About />} />
                            <Route path="*" element={<Error404 />} />
                        </Routes>
                    </div>
                </ProviderLogements>
                <Footer className="footer" />
            </BrowserRouter>
        </React.StrictMode>
    );
}

export default Router;
```

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.

Fonctionnalités

Home Page

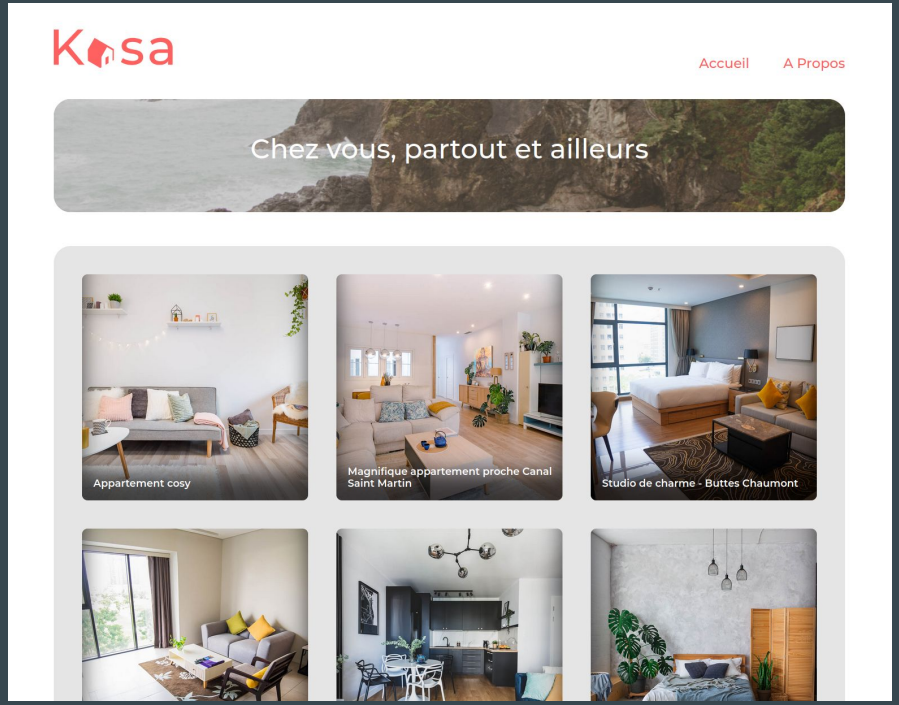
*Les composants développés sur ce projet sont des **composants fonction.***

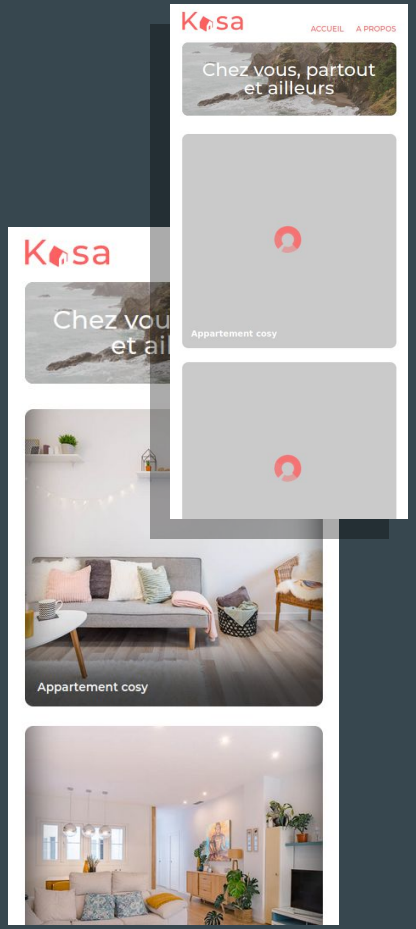
Le chargement des images est géré par l'affichage d'un **loader Spinner**.

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**.

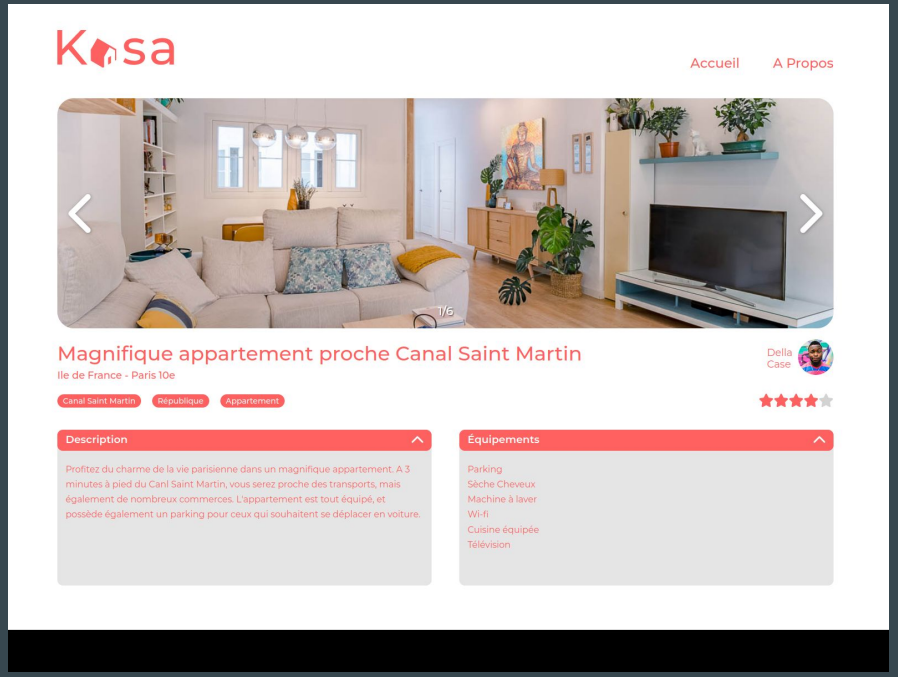
Des break-points sont définis à 964px pour les **tablettes** et 768 px pour les **mobiles**.





Fonctionnalités

Logement

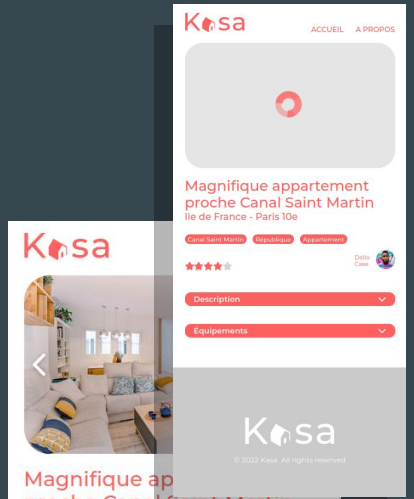


Le **Carousel d'images** boucle de la dernière image à la première, et vice-versa.
La gestion des écrans tactile est également implémenté.
Le chargement des images est géré par l'affichage du loader.

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

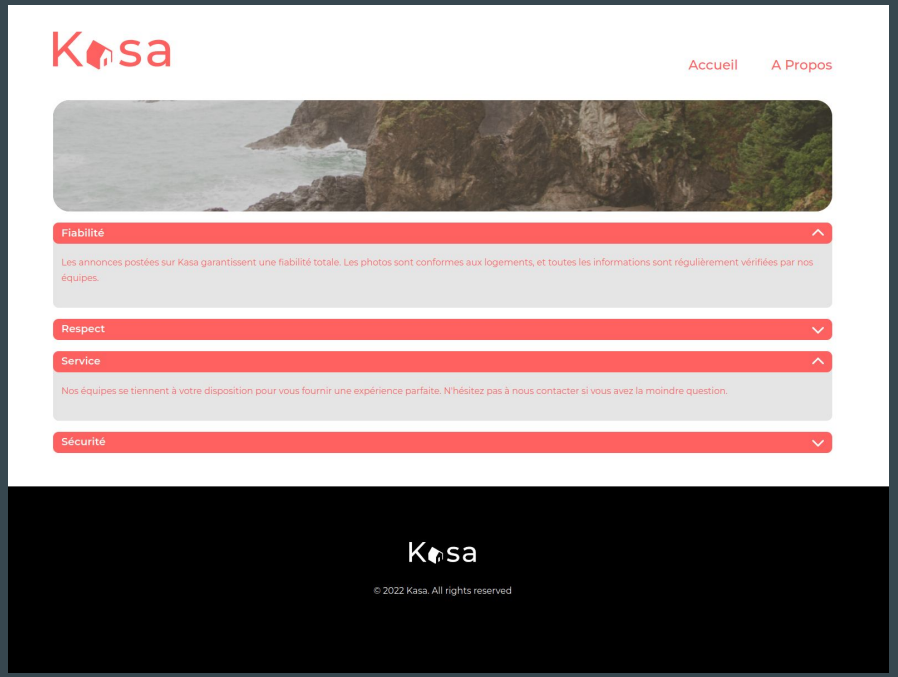
Les "id" erronés entrées par l'utilisateur dans l'URL déclenchent une **redirection** sur la page d'erreur.

L'ensemble des éléments du Figma sont en place.



Fonctionnalités

A propos



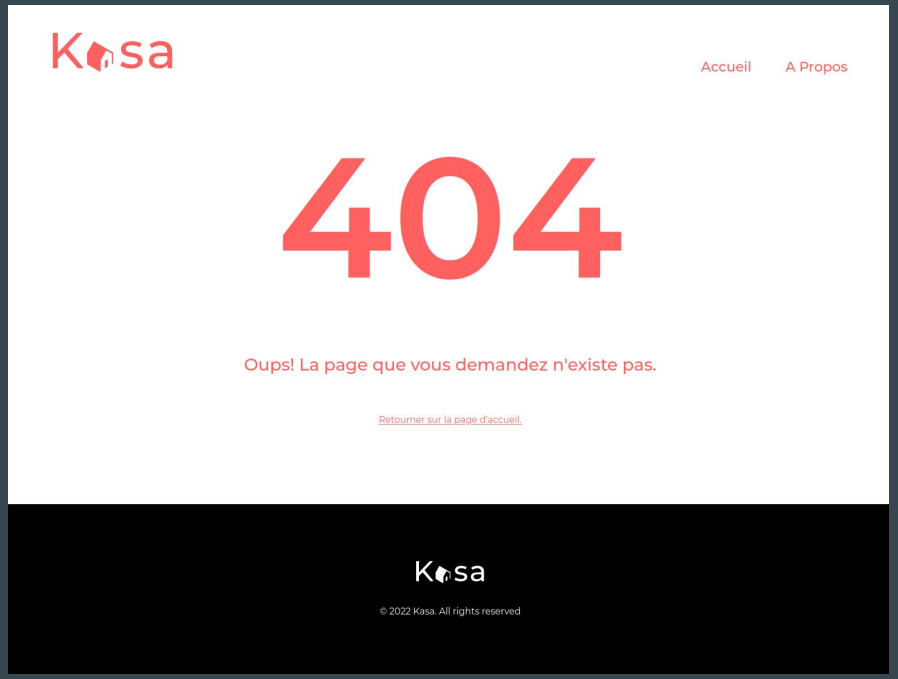
Le composant **Textual** est employé de nouveau quatre fois.

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

```scss
@import "../styles/normalize.css";
@import "../styles/variables";

*,
*::before,
*::after {
  box-sizing: border-box;
}

body {
  margin: 0;
  font-family: $Montserrat, "Segoe UI", "Roboto", "Oxygen", "Ubuntu",
    "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue", sans-serif;
  font-weight: 400;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

.Link {
  color: $red;
  text-decoration: none;
  &:hover {
    text-decoration: underline;
  }
}
```

_variables.scss

```scss
@import url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,300;0,400;0,500;0,600;1,300;1,400;1,500;1,600&display=swap');


// Font variables
$Montserrat: 'Montserrat', sans-serif;

// Colors variables
$red: #FF6060;
$greyl: #E5E5E5;
$greym: #cacaca;
```

```scss
@media (max-width: 964px) {
  .App {
    padding: 0 40px;
  }
}

@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.