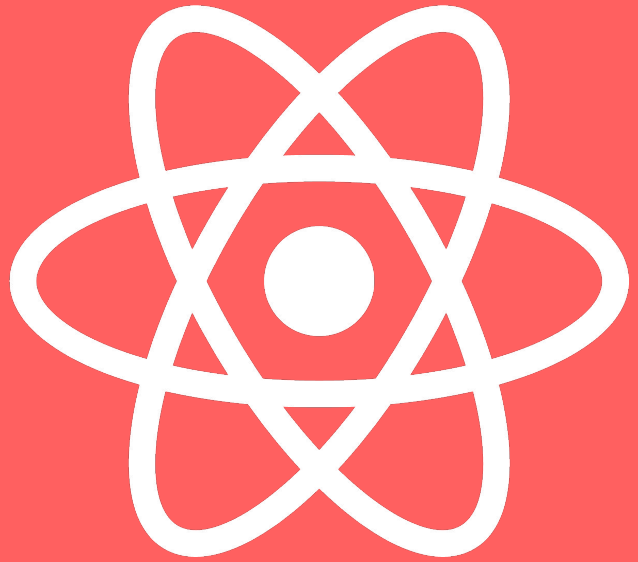
Project 7

REACT FRONT-END FOR:

Kasa



OpenClassrooms web developer program

Molly Felts Vallin | January 2023

Avant coding :

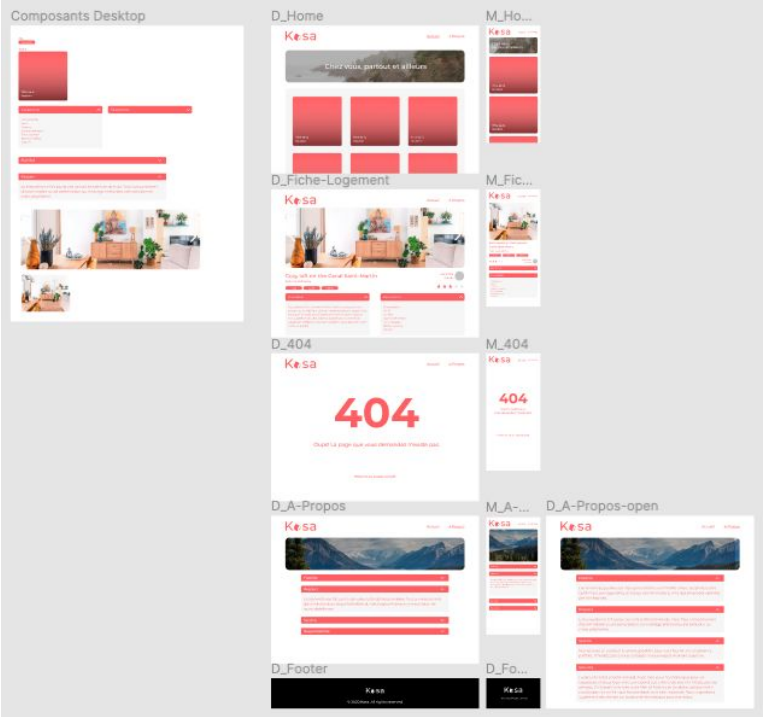
comprendre les demandes du projet et étudier la maquette fournie en Figma (Desktop + Mobile)

➔ **liste des routes :**

- Home (accueil)
- Logement
- About (a propos)
- Error (404)

➔ **composants React:**

- Header
- Banner
- Accordion / Collapse
- Footer
- Card Gallery (thumbnail images)
- Fiche Logement :
  - Carrousel (image slider)
  - Logement Title
  - Rating (étoiles)



Utilisation de 'React' :

dans le terminal de VSC, mettre les commandes suivent pour utiliser React comme bibliothèque

INITIALISER :

```
yarn create react-app kasa
```

yarn (ou npm) comme package manager:

- commencer avec 'yarn init'
- créer un nouveau app React
- installer les dépendances (dans node_modules)
- mettre l'architecture du site front-end dans vsc (dossiers)

CRÉER ROUTER :

```
yarn add react-router-dom
```

React router:

- permettre plusieurs 'pages' dans un seul téléchargement du site
- définir les urls/pages dans votre app (routes)

LANCER L'APP :

```
yarn start / yarn build
```

Start:

- mettre le site sur localhost (dev mode)

Build:

- optimiser les fichiers afin déploiement/versionnent

Create react-app :

l'architecture de directory de base est fourni par React

```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   ├── logo192.png
│   ├── logo512.png
│   ├── manifest.json
│   └── robots.txt
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    ├
serviceWorker.js
    └── setupTests.js
```

->

```
kasa
├── Readme, modules, package, gitignore
├── yarn.lock
└── src
    ├── assets
    ├── components
    │   ├── Accordion.jsx
    │   ├── Banner.jsx
    │   ├── Carrousel.jsx
    │   ├── Footer.jsx
    │   ├── Header.jsx
    │   ├── LogementsGallery.jsx
    │   ├── LogementTitle.jsx
    │   └── Rating.jsx
    ├── data
    │   └──logementList.js
    ├── pages
    │   └──Home, About, Error, Logement
.jsx
    ├── styles
    │   ├── _base.scss
    │   ├── _components.scss
    │   ├── _layout.scss
    │   └── app.scss
    ├── App.jsx
    └── index.jsx
```

Affecter le DOM :

mettre App.jsx comme le root du site qui gère les éléments rendent dans le DOM virtuel par React

```
import { createRoot } from "react-dom/client";
import "./styles/app.scss";
import App from "./App";
import { BrowserRouter } from "react-router-dom";
const root = createRoot(document.getElementById("root"));

root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

Kasa/App.jsx :

gérer les routes utilisent par le React router et afficher le header et footer à chaque page (rendre)

PAGE :	*FICHIER :*	*COMPOSANTS :*
TOUS	App.jsx	Header Footer
accueil	Home.jsx	Banner LogementsGallery
a propos	About.jsx	Banner Accordion
logement	Logement.jsx	Carrousel LogementTitle Accordion
404	Error.jsx	aucun

```jsx
function App() {
  return (
    <div className="App">
      <Header />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="logement/:id/*" element={<Logement />} />
        <Route path="about/" element={<About />} />
        <Route path="*" element={<Error />} />
      </Routes>
      <Footer />
    </div >
  )
}

export default App
```

Kasa/pages/Logement.jsx :

code pour les 'logement-fichier' pages

```
import { useParams, Routes, Route } from "react-router-dom";
import { logementList } from "../data/logementList"
import Carrousel from "../components/Carrousel";
import LogementTitle from "../components/LogementTitle";
import Accordion from "../components/Accordion";
import Error from "../pages/Error"

function Logement() {
  const { id } = useParams();
  const logement = logementList.filter((logement => logement.id === id));
  console.log(logement)

  if (logement.length === 0 || !logement) {
    return (
      <Routes>
        <Route path="/*" element={<Error />} />
      </Routes>)
  }
```

importer :

- ➔ les routes, id mettre dans l'url
- ➔ le .json fourni ({logementList})
- ➔ les composants (components)

function de définir :

- ➔ les props d'utiliser
- ➔ les 'class' pour .css/styles
- ➔ les composants

```
else {
    return (
      <div className="container_main">
        <Carrousel
          key={`${id}_Carrousel`}
          logementId={id} />
        <LogementTitle
          key={`${id}_LogementTitle`}
          logementId={id} />
        <div className="container_details">
          <Accordion
            key={`${id}_LogementDescription`}
            title="Description"
            content={logement[0].description} />
          <Accordion
            key={`${id}_LogementEquipments`}
            title="Équipements"
            content={logement[0].equipments} />
        </div>
      </div>
    );
  }
}

export default Logement;
```

Kasa/components/Carrousel.jsx :

le useState React hook permet de stocker valeurs entre rendus différents

- ➔ il faut définir l'état initial (0)
- ➔ returns un value (current) + function qui le change
  - nextSlide() faire +1
  - previousSlide() faire -1
- ➔ cette value est utilisée pour rendre l'image d'array qui a le même valeur comme index

```
function Carrousel({ logementId }) {
  const [current, setCurrent] = useState(0);
  const logement = logementList.filter(
    (logement => logement.id === logementId));

  const slides = logement[0].pictures
  const slidesTotal = slides.length
  console.log(slides)
  console.log(slidesTotal)

  const nextSlide = () => {
    setCurrent(current === slidesTotal - 1 ? 0 : current + 1)};

  const previousSlide = () => {
    setCurrent(current === 0 ? slidesTotal - 1 : current - 1)};
```

```
    return (
      <section className="carrousel">
        <div className="carrousel_images">
          {slides.map((picture, index) => (
            <div className={index === current ? 'image_active' : 'image_hidden'}
            key={index}>
              <img src={picture} alt={`${logement[0].title} ${current + 1}`} />
            </div>
          ))}
        </div>
        <div className="carrousel_controls">
          <div className="controls_prev" onClick={previousSlide}>
            <img src={arrow} alt="flèche précédente" />
          </div>
          <div className="controls_next" onClick={nextSlide}>
            <img src={arrow} alt="flèche suivant" />
          </div>
        </div>
        <p className="slide_index">{`${current + 1} / ${slidesTotal}`}</p>
      </section>
    )
  }
};

export default Carrousel;
```

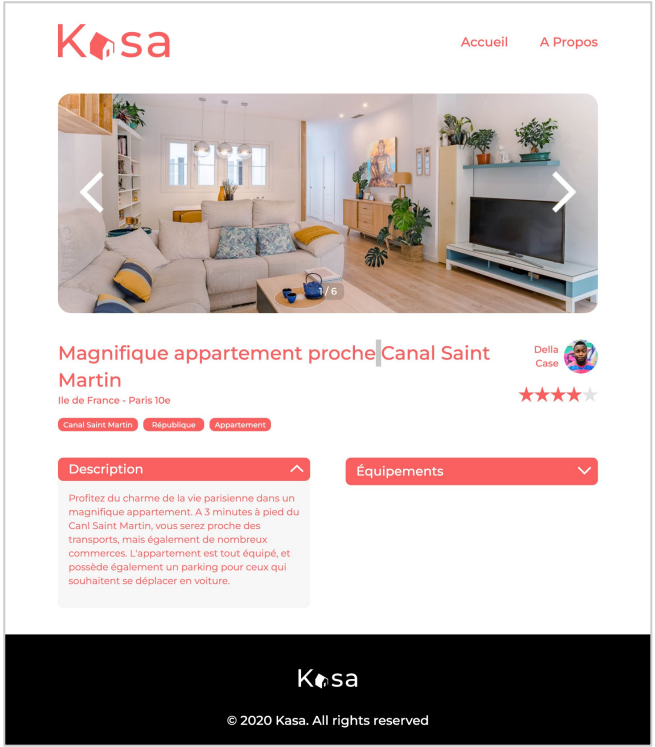
Design Responsive

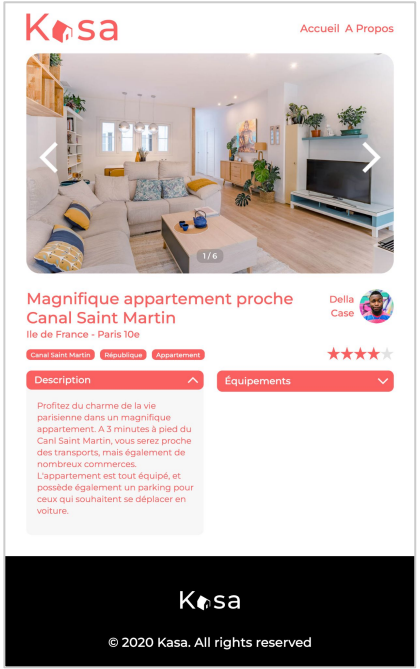
utilisation du SASS mixin pour gérer les tailles d'écrans différents

```scss
@mixin for-mobile {
  @media (max-width: 480px) {
    @content;
  }
}

@mixin for-tablet {
  @media (min-width: 481px) and
  (max-width: 991px) {
    @content;
  }
}

@mixin for-desktop-small {
  @media (min-width: 992px) and
  (max-width: 1024px) {
    @content;
  }
}
```





Fonctions particuliers

Page 404 (erreur)

➔ la route est gérer avec un wildcard (*)
➔ mettre si l'id n'existe pas dans logementList.json
➔ mettre si un page/route n'existe pas

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="logement/:id/*" element={<Logement />} />
  <Route path="about/" element={<About />} />
  <Route path="*" element={<Error />} />
</Routes>
```

Accordion

➔ utiliser le même composent pour les pages:
- ◆ Logement Fichier
- ◆ A Propos

Gallery Controls

➔ n'affiche pas s'il y a moins que 2 images

Magnifique appartement Rivoli
Ile de France - Paris 13e

En conclusion :

React permet de :

- ➔ créer un site SPA (single page-application)
- ➔ rendre les contenus du page comme éléments sans avoir rechargé toute la page
- ➔ utiliser les composants entre pages différents
- ➔ contrôler les styles avec .css et est compatible avec SASS

Merci