Jayson LEPAGE

PROJET BILAN : Créez le site Web "Trouve ton artisan"

Trouve ton artisan !
Avec la région
Auvergne-Rhône-Alpes

TABLE DES MATIERES

INTRODUCTION

Résumé du projet

Projet Trouve ton artisan

Mon projet bilan est un site internet ayant pour but de faciliter la vie des habitants de la région en leurs permettant de trouver rapidement et facilement un artisan pour leurs différents besoins. Que ce soit pour des travaux de bâtiments, des services, des produits de fabrication ou de l'alimentation, notre plateforme offrira un accès simple et intuitif à une multitude d'artisans qualifiés.

Le site comprend plusieurs fonctionnalités clés. Tout d'abord, les utilisateurs peuvent naviguer à travers différentes catégories d'artisanat grâce à un menu clair et convivial. Ensuite, ils ont la possibilité de consulter les profils des artisans, qui comprennent leur nom, leur spécialité, leur localisation et leurs évaluations par d'autres utilisateurs.

Pour faciliter la prise de contact, un formulaire de contact est présent permettant aux utilisateurs d'envoyer directement leurs demandes aux artisans. De plus, une page dédiée à chaque artisan fournit des informations détaillées, une présentation, des coordonnées et un lien vers leur site web.

Il y a également les éléments essentiels sur un site internet tels que la gestion de la vie privée et des cookies, les mentions légales, l'accessibilité ainsi que le contact, dont celui comprendra différents éléments pour la contacter comme le numéro de téléphone, l'adresse de l'entreprise etc..

En termes de conception, nous avons choisi un design responsive et épuré, mettant en avant la facilité d'utilisation et la clarté de l'information. Nous nous avons engager à respecter les normes de sécurité les plus strictes pour garantir la confidentialité et la sécurité des utilisateurs.

En résumé, « Trouve ton artisan » aspire à devenir la référence en matière de recherche d'artisans dans la région Auvergne Rhône-Alpes, offrant une expérience utilisateur optimale et une plateforme fiable pour connecter les particuliers aux artisans locaux.

CONCEPTION

Besoins du site internet

A qui s'adresse-t-il ?

Le client est un site internet permettant de trouver des artisans .

A quoi sert-il ?

Ce site web sert à fournir une plateforme où les utilisateurs peuvent trouver des artisans qualifié dans différents domaines (alimentation, bâtiment, fabrication, services).

Quel est le but de ce site web ?

Le but de ce site web est de fournir une plateforme permettant aux utilisateurs de trouver facilement des artisans dans divers domaines tels que l'alimentation, le bâtiment, la fabrication et les services. L'objectif principal est de faciliter la mise en relation entre les artisans et les personnes recherchant leurs services. Le site vise également à fournir des informations pertinentes sur les artisans, y compris leurs spécialités, leurs localisations et éventuellement les notes attribuées par les utilisateurs. En outre, le site peut être conçu pour permettre aux utilisateurs de contacter directement les artisans via un formulaire de contact, facilitant ainsi les demandes de renseignements ou les demandes de services. En résumé, le but principal est de créer une plateforme conviviale et fonctionnelle pour faciliter la la recherche et la prise de contact avec des artisans qualifiés.

Spécifications fonctionnelles

Les différentes parties du site internet

1 . Header (En-tête) : Il s'agit de la section située en haut de chaque page du site, contenant le logo, le menu de navigation et la barre de recherche.

2 . Footer (Pied de page) : C'est la section située en bas de chaque page du site contenant des coordonnées de contact, numéro de téléphone, adresse etc.. et les mentions légales, données personnelles, l'accessibilité et les cookies.

3 . Pages principales :

- Page d'accueil : Elle présente une vue d'ensemble avec une explications sous différentes étapes pour trouver son artisan ainsi elle présente les trois artisans du mois.
- Pages spécifiques à chaque catégorie d'artisans (Alimentation, Bâtiment, Fabrication, Services) : Elles fournissent des informations sur les artisans dans chaque catégorie, avec des filtres de recherche pour aider les utilisateurs à trouver ce qu'ils cherchent plus facilement.

4 . Page détail de l'artisan :

- Nom de l'artisan : Le nom de l'artisan est affiché en haut de la page.
- Note : Une évaluation ou une note moyenne attribuée à l'artisan est affichée pour donner aux utilisateurs une idée de sa réputation.
- Spécialité : Une description de la spécialité de l'artisan est fournie, par exemple, s'il s'agit d'un maçon, d'un plombier, etc.
- A propos : Une brève description de l'artisan.
- Localisation : L'emplacement de l'artisan est indiqué.
- Site Web : Le lien du site web de l'utilisateur permettant aux utilisateurs de se faire une idée.
- Formulaire de contact : Un formulaire permettant aux utilisateurs de contacter directement l'artisan pour poser des questions, demander des devis ou autres.

5 . Page détail de l'artisan :

- Mentions légales : Cette page fournit des informations sur l'entreprise derrière le site web, ainsi que les conditions d'utilisation.
- Données personnelles : Cette section explique comment les données personnelles des utilisateurs sont collectées, utilisées et protégées sur le site.
- Accessibilité : Des informations sur l'accessibilité du site aux personnes handicapées sont fournies, conformément aux réglementations en vigueur.
- Cookies : Cette page informe les utilisateurs sur l'utilisation des cookies sur le site et leur donne la possibilité de gérer leurs préférences en matière de cookies.

CONCEPTION

Technologies utilisées

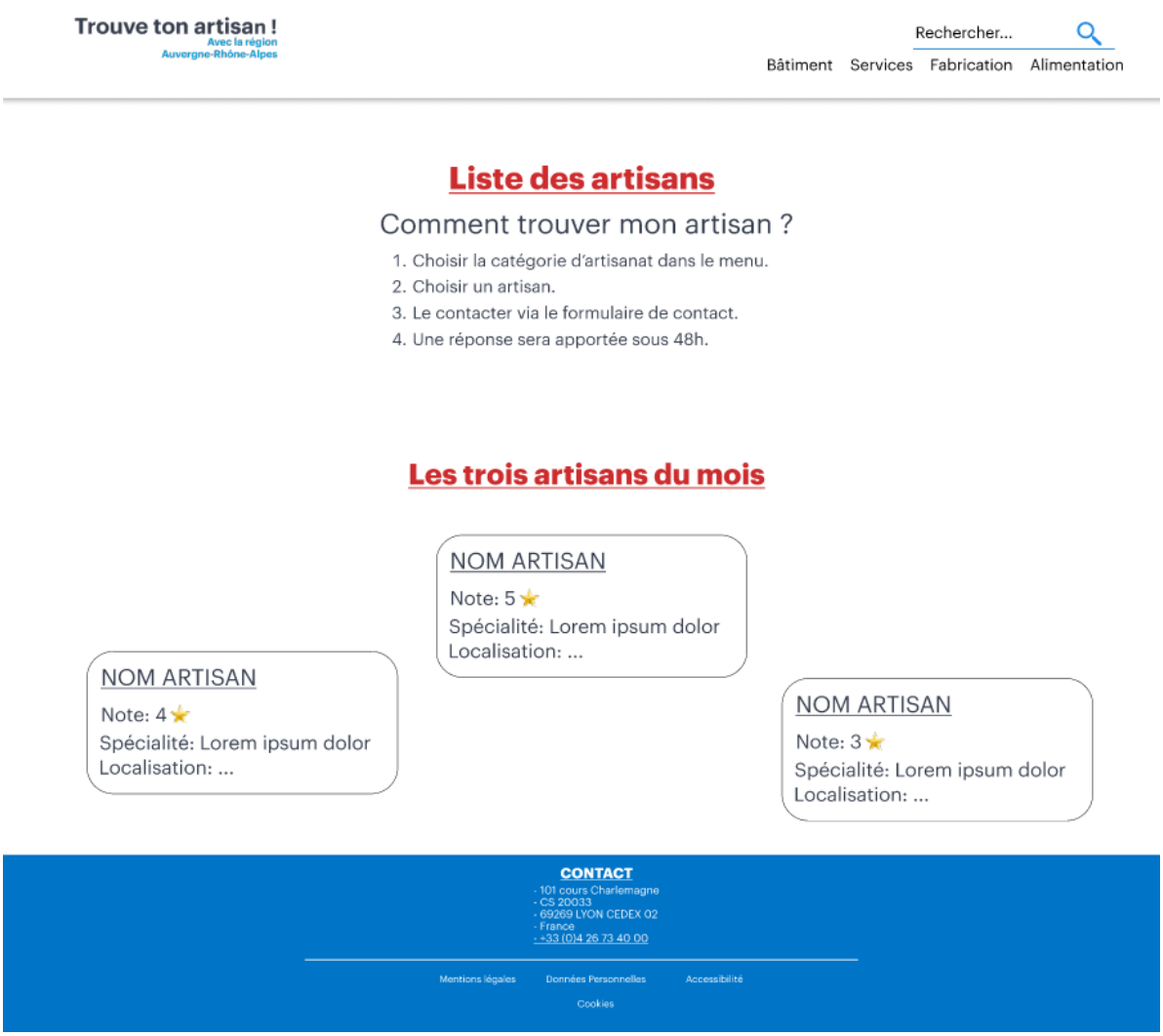
Afin de réaliser mon projet, j'ai décidé d'utiliser du JavaScript avec le framework React.JS et du CSS pour la partie Front-end.

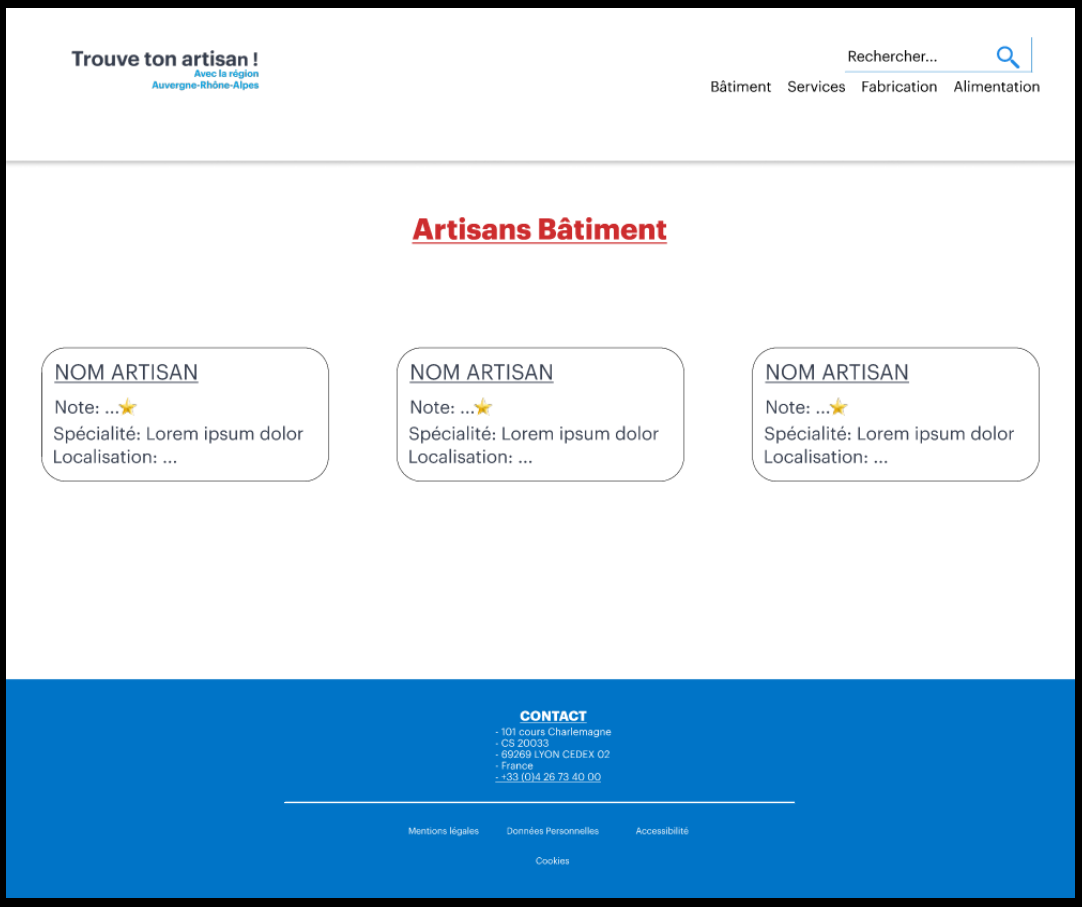
Maquettage

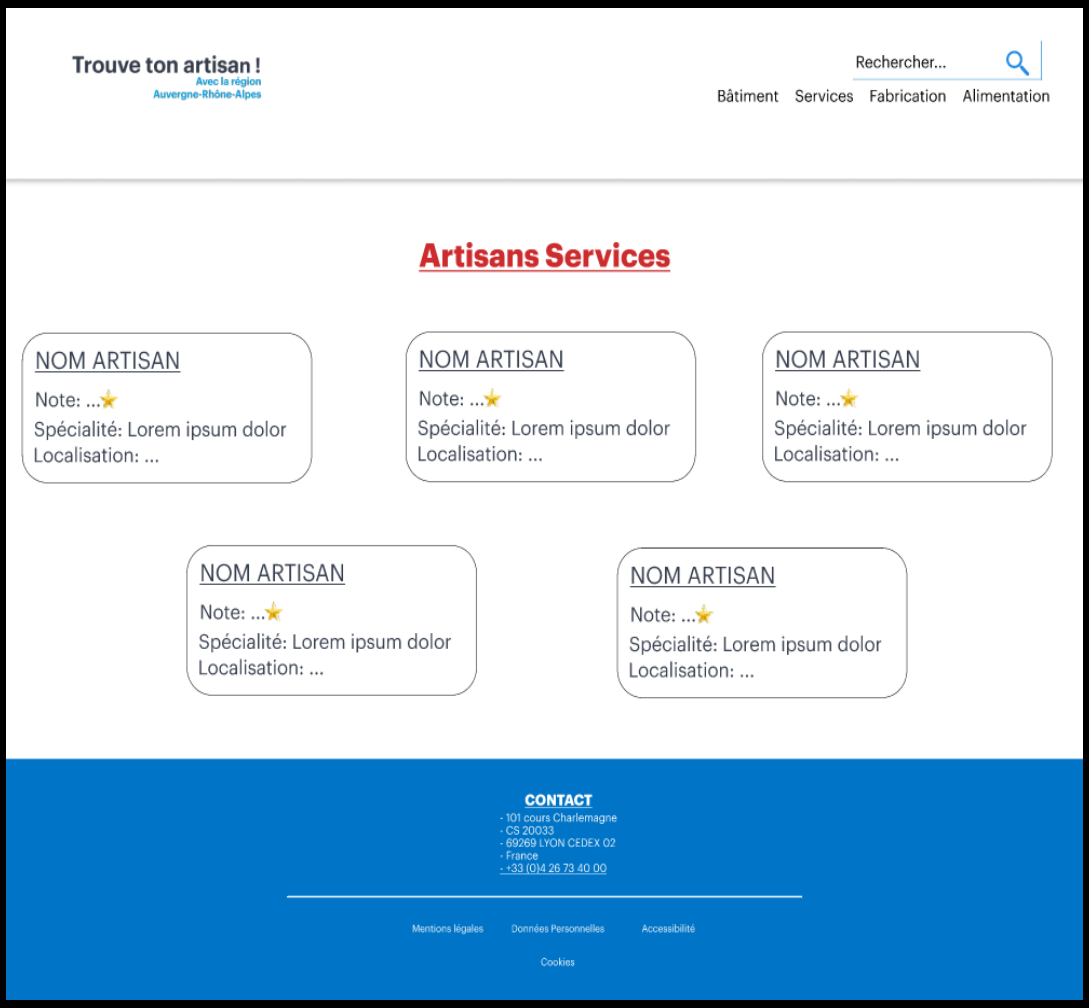
Interface graphique

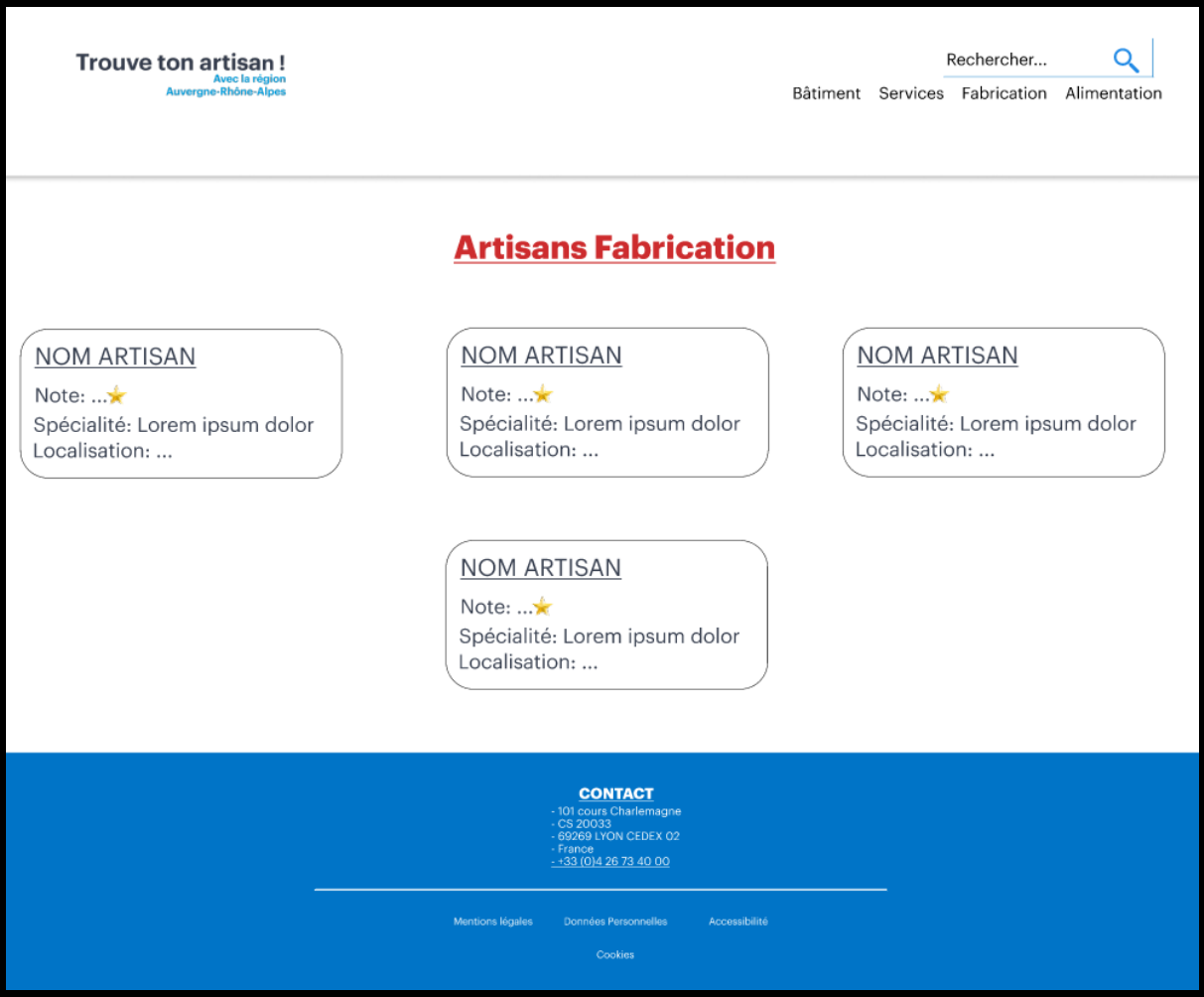
Ces maquettes ont été réalisées avec Figma.

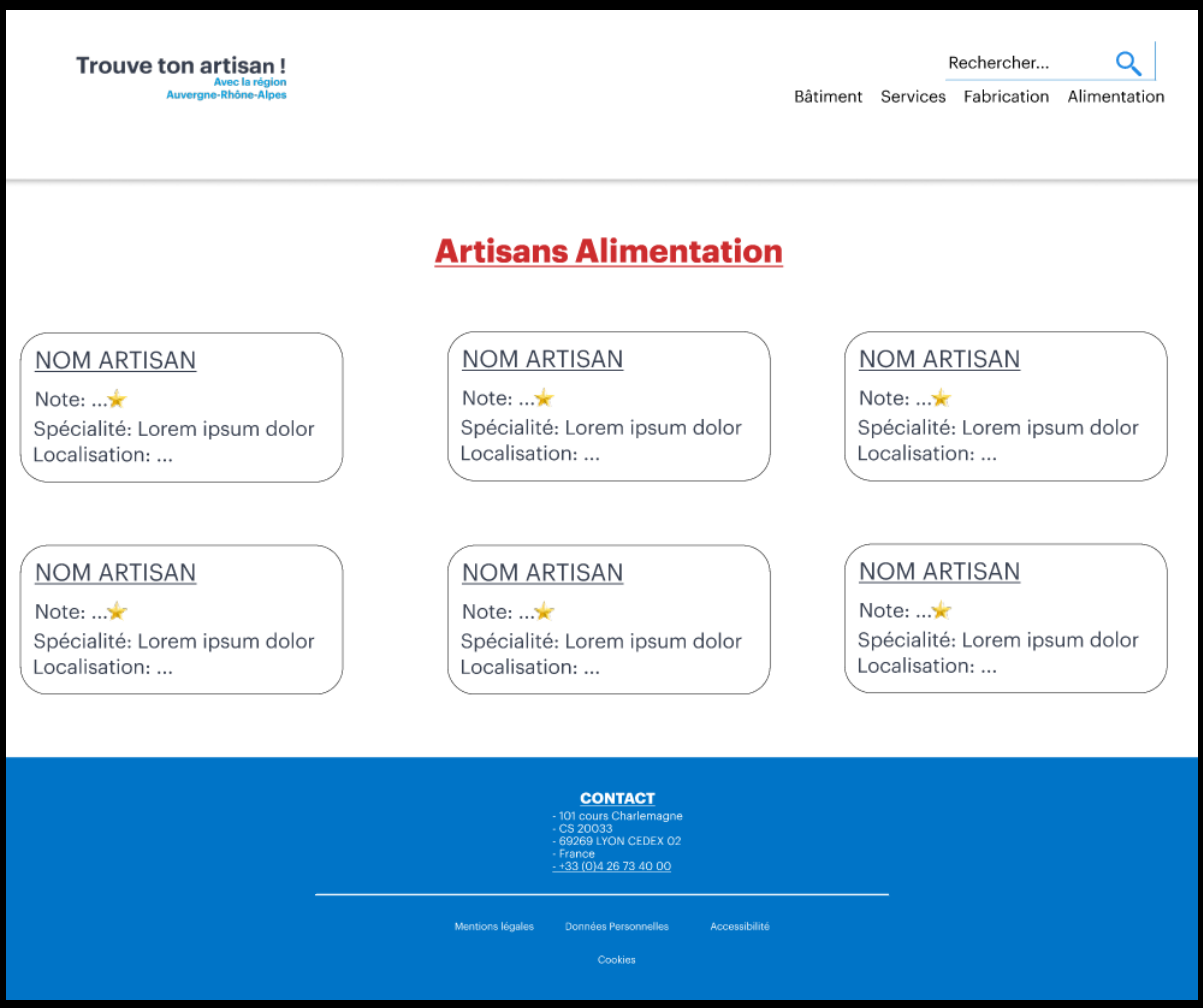
Design grand format



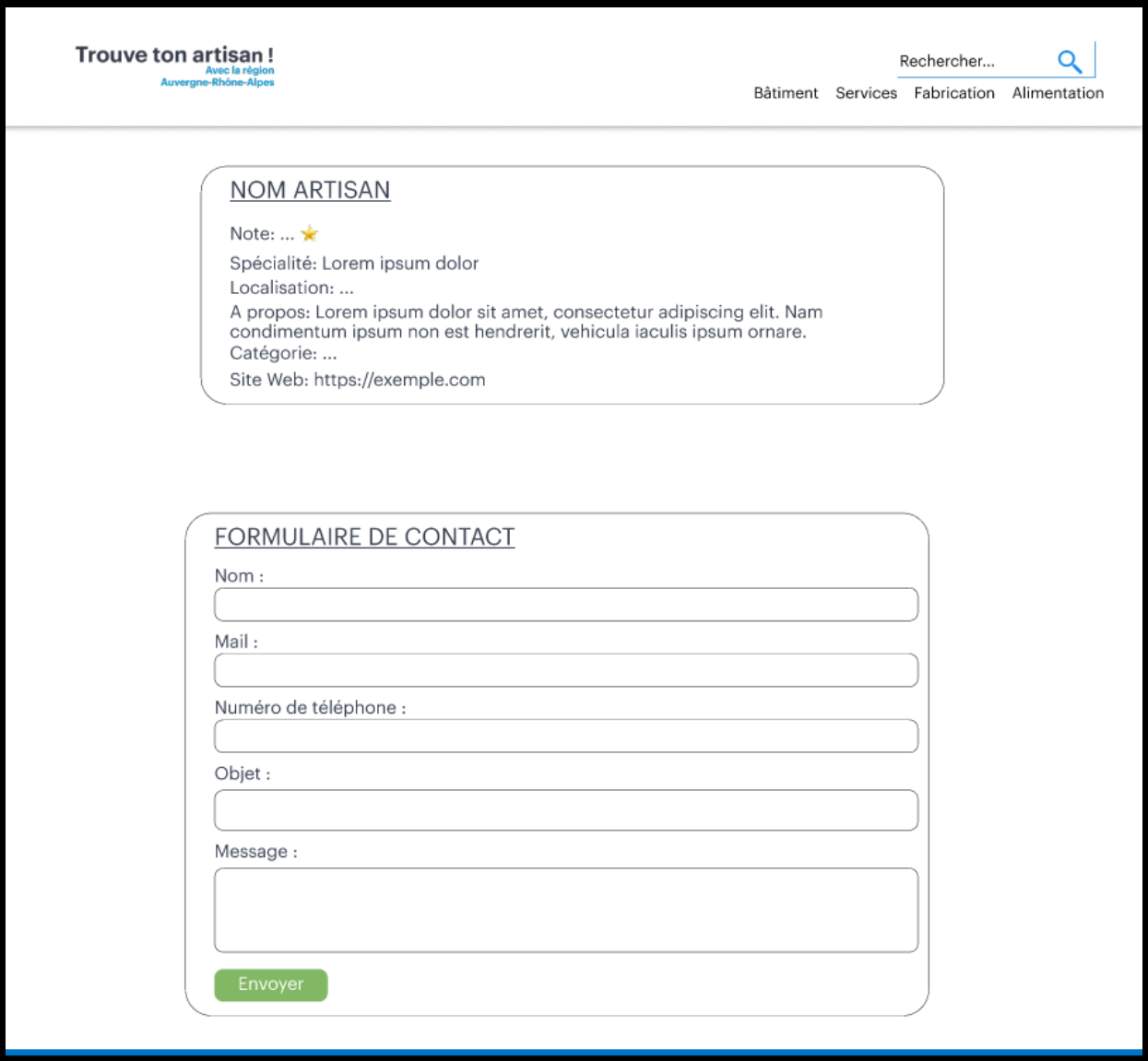

Trouve ton artisan !
Avec la région
Auvergne-Rhône-Alpes
Rechercher...
Bâtiment Services Fabrication Alimentation
Artisans Bâtiment
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
CONTACT
- 101 cours Charlemagne
- CS 20033
- 69269 LYON CEDEX 02
- France
- +33 (0)4 26 73 40 00
Mentions légales
Données Personnelles
Accessibilité
Cookies

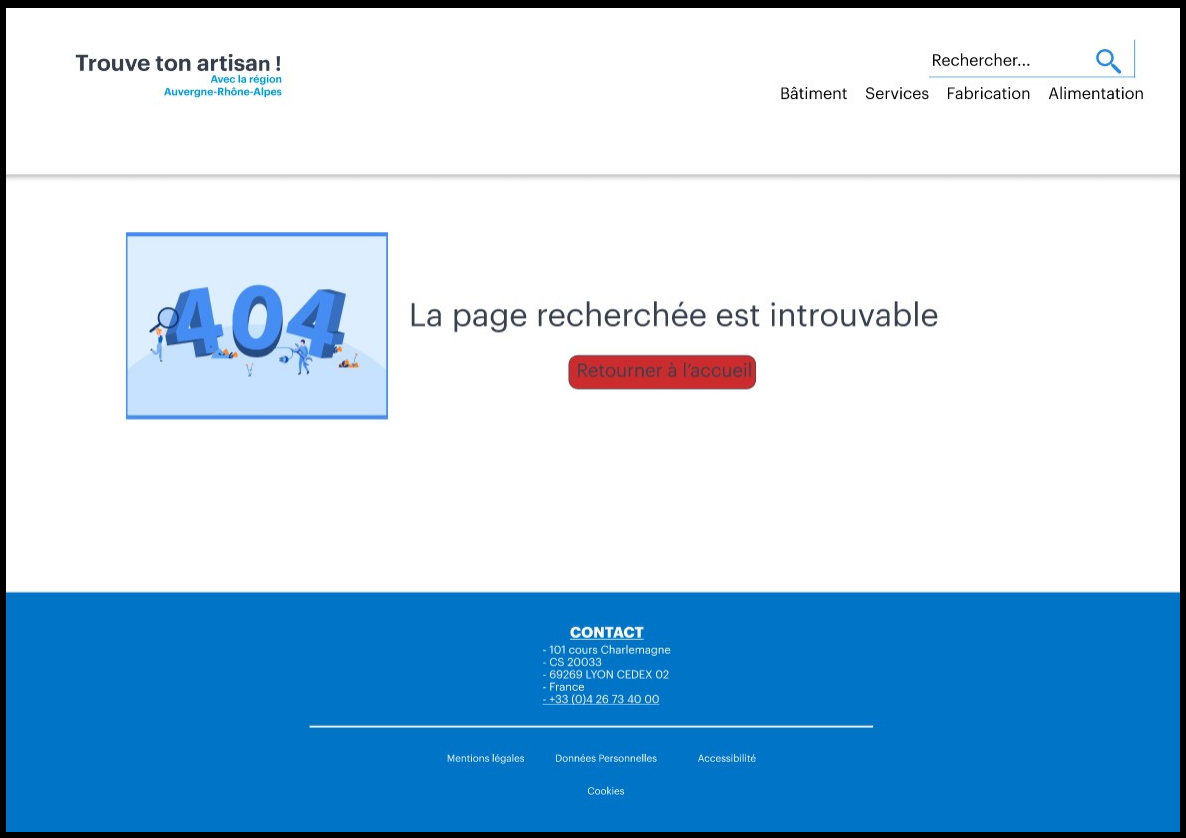

Trouve ton artisan !
Avec la région
Auvergne-Rhône-Alpes
Rechercher...
Bâtiment Services Fabrication Alimentation
Artisans Services
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
CONTACT
- 101 cours Charlemagne
- CS 20033
- 69269 LYON CEDEX 02
- France
- +33 (0)4 26 73 40 00
Mentions légales
Données Personnelles
Accessibilité
Cookies


Trouve ton artisan !
Avec la région
Auvergne-Rhône-Alpes
Rechercher...
Bâtiment Services Fabrication Alimentation
Artisans Fabrication
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
CONTACT
- 101 cours Charlemagne
- CS 20033
- 69269 LYON CEDEX 02
- France
- +33 (0)4 26 73 40 00
Mentions légales
Données Personnelles
Accessibilité
Cookies

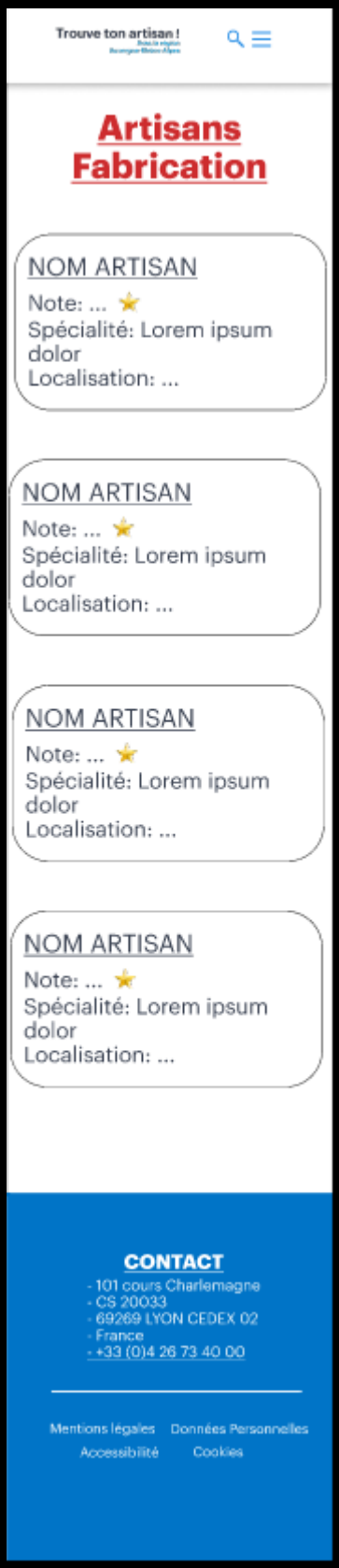

Trouve ton artisan !
Avec la région
Auvergne-Rhône-Alpes
Rechercher...
Bâtiment Services Fabrication Alimentation
Artisans Alimentation
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
CONTACT
- 101 cours Charlemagne
- CS 20033
- 69269 LYON CEDEX 02
- France
- +33 (0)4 26 73 40 00
Mentions légales
Données Personnelles
Accessibilité
Cookies

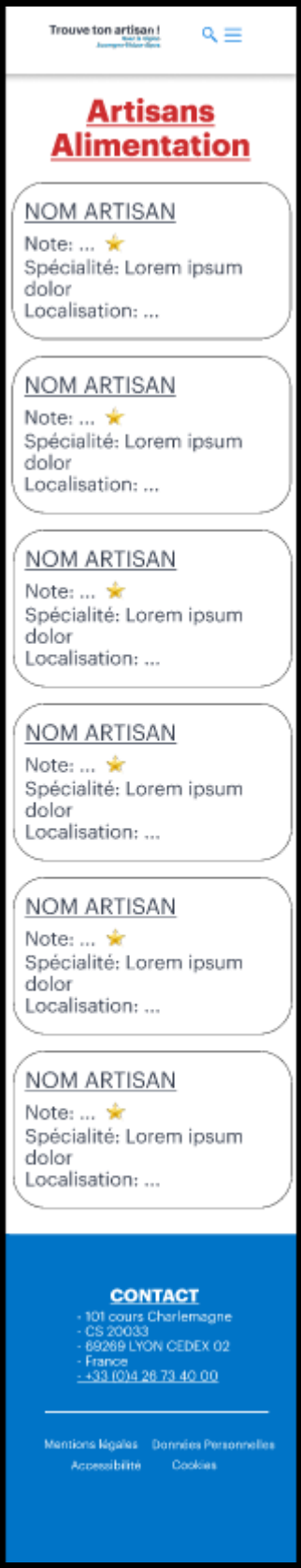
« Le footer est bien présent en bas de la page »





Design smartphone


Trouve ton artisan !
Artisans Fabrication
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
CONTACT
- 101 cours Charlemagne
- CS 20033
- 69269 LYON CEDEX 02
- France
- +33 (0)4 26 73 40 00
Mentions légales Données Personnelles
Accessibilité Cookies


Trouve ton artisan !
Artisans Alimentation
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
CONTACT
- 101 cours Charlemagne
- CS 20033
- 69269 LYON CEDEX 02
- France
- +33 (0)4 26 73 40 00
Mentions légales Données Personnelles
Accessibilité Cookies


Trouve ton artisan !
NOM ARTISAN
Note: ...
Spécialité: Lorem ipsum dolor
Localisation: ...
A propos: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam condimentum ipsum non est hendrerit, vehicula iaculis ipsum ornare.
Catégorie: ...
Site Web: https://exemple.com
NOM ARTISAN
Nom :
Mail :
Numéro de téléphone :
Objet :
Message :
Envoyer
CONTACT
- 101 cours Charlemagne
- CS 20033
- 69269 LYON CEDEX 02
- France
- +33 (0)4 26 73 40 00
Mentions légales Données Personnelles
Accessibilité Cookies


Trouve ton artisan !
404
La page recherchée est introuvable
Retourner à l'accueil
CONTACT
- 101 cours Charlemagne
- CS 20033
- 69269 LYON CEDEX 02
- France
- +33 (0)4 26 73 40 00
Mentions légales
Données Personnelles
Accessibilité
Cookies

REALISATION DU PROJET

Développement de partie Front-end

Dans un premier temps, nous découvrirons les éléments qui ont été utiles pour créer l'interface graphique du site internet.

JavaScript Framework React.JS

App.js

```jsx
import React, { useState } from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Header from "../src/components/Header/Header";
import Footer from "../src/components/Footer/Footer";
import Home from "../src/components/Home/Home";
import Artisan from "../src/components/Artisan/Artisan";
import Alimentation from "../src/components/Services/Alimentation";
import Batiment from "../src/components/Services/Batiment";
import Fabrication from "../src/components/Services/Fabrication";
import Services from "../src/components/Services/Services";
import artisans from "../src/datas/datas.json";
import LegalMentions from "../src/components/Legal/Mentions";
import LegalPersonal from "../src/components/Legal/Personal";
import LegalAccessibility from "../src/components/Legal/Accessibility";
import LegalCookies from "../src/components/Legal/Cookies";
import Error404 from "./components/404/404";

const App = () => {
  const [searchTerm, setSearchTerm] = useState("");
  const [filteredArtisans, setFilteredArtisans] = useState(artisans);

  const handleSearch = (searchTerm) => {
    const lowerCaseSearchTerm = searchTerm.toLowerCase().trim();

    if (!lowerCaseSearchTerm) {
      setFilteredArtisans(artisans);
      return;
    }

    const filteredResults = artisans.filter((artisan) => {
      const lowerCaseName = artisan.name.toLowerCase();
      const lowerCaseSpecialty = artisan.specialty.toLowerCase();
      const lowerCaseLocation = artisan.location.toLowerCase();

      return (
        lowerCaseName.includes(lowerCaseSearchTerm) ||
        lowerCaseSpecialty.includes(lowerCaseSearchTerm) ||
        lowerCaseLocation.includes(lowerCaseSearchTerm)
      );
    });

    setFilteredArtisans(filteredResults);
    setSearchTerm(lowerCaseSearchTerm);
  };

  return (
    <Router>
      <Header
        onSearch={handleSearch}
        searchTerm={searchTerm}
        setSearchTerm={setSearchTerm}
      />

      <Routes>
        <Route path="/" element={<Home artisans={filteredArtisans} />} />
        <Route path="/artisan/:id" element={<Artisan />} />
        <Route
          path="/alimentation"
          element={
            <Alimentation
              artisans={filteredArtisans}
              searchResults={filteredArtisans}
            />
          }
        />
        <Route
          path="/batiment"
          element={
            <Batiment
              artisans={filteredArtisans}
              searchResults={filteredArtisans}
            />
```

```
            }
          />
          <Route
            path="/fabrication"
            element={
              <Fabrication
                artisans={filteredArtisans}
                searchResults={filteredArtisans}
              />
            }
          />
          <Route
            path="/services"
            element={
              <Services
                artisans={filteredArtisans}
                searchResults={filteredArtisans}
              />
            }
          />
          <Route path="/legal/mentions" element={<LegalMentions />} />
          <Route path="/legal/personal" element={<LegalPersonal />} />
          <Route path="/legal/accessibility" element={<LegalAccessibility />} />
          <Route path="/legal/cookies" element={<LegalCookies />} />
          <Route path="*" element={<Error404 />} />
        </Routes>
        <Footer />
      </Router>
    );
};

export default App;
```

En premier temps, le code importe les modules nécessaires à React, notamment « React » et le hook « useState » depuis React. Il importe également des composants de navigation depuis « react-router-dom » pour gérer la navigation au sein de l'application.

En deuxième temps, le composant fonctionnel « App » est défini. Ce composant est le composant racine de l'application.

En troisième temps, il déclare deux états locaux à l'aide du hook « useState » : « searchTerm » pour stocker le terme de recherche saisi par l'utilisateur, et « filteredArtisans » pour stocker la liste filtrée d'artisans en fonction du terme de recherche.

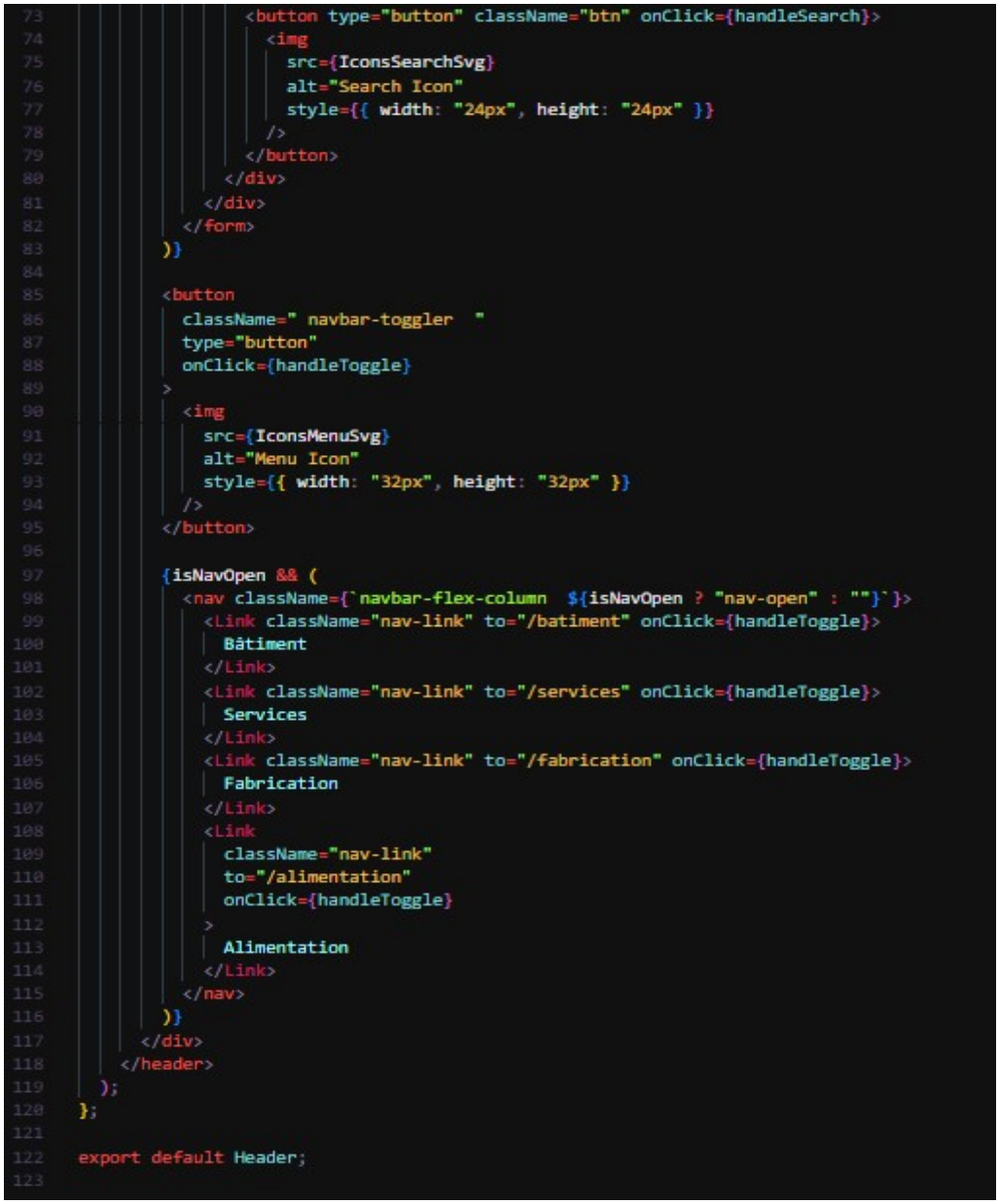
En quatrième temps, une fonction « handleSearch » est définie pour gérer la recherche des artisans en fonction du terme saisi par l'utilisateur.

En cinquième temps, le rendu JSX du composant « App » est défini. Il encapsule les composants de l'application avec le routeur « BrowserRouter » pour permettre la navigation entre différentes pages.

En sixième temps, les différentes routes de l'application sont définies à l'aide du composant « Routes ». Chaque route correspond à un chemin d'URL et rend un composant spécifique en fonction de ce chemin. Par exemple, la route « / » rend le composant « Home », la route « /alimentation » rend le composant « Alimentation », etc.

En septième temps, le composant « Footer » est ajouté à la fin de l'application pour afficher les informations de contact et les liens légaux.

```jsx
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import Logo from "../assets/images/Logo.png";
import IconsMenuSvg from "../assets/images/icons-menu.svg";
import IconsSearchSvg from "../assets/images/icons-search.svg";

const Header = ({ onSearch, searchTerm, setSearchTerm }) => {
  const [isNavOpen, setIsNavOpen] = useState(window.innerWidth > 1024);
  const [isSearchOpen, setIsSearchOpen] = useState(window.innerWidth > 1024);

  const handleSearch = (e) => {
    e.preventDefault();
    onSearch(searchTerm.trim());
  };

  const handleToggle = () => {
    if (window.innerWidth <= 1024) {
      setIsNavOpen(!isNavOpen);
      setIsSearchOpen(false);
    }
  };

  const handleSearchToggle = () => {
    if (window.innerWidth <= 1024) {
      setIsSearchOpen(!isSearchOpen);
      setIsNavOpen(false);
    }
  };

  useEffect(() => {
    const handleResize = () => {
      setIsNavOpen(window.innerWidth > 1024);
      setIsSearchOpen(window.innerWidth > 1024);
    };

    window.addEventListener("resize", handleResize);

    return () => {
      window.removeEventListener("resize", handleResize);
    };
  }, []);

  return (
    <header className="d-flex align-items-center justify-content-between">
      <Link to="/">
        <img src={Logo} alt="Company Logo" style={{ width:"15rem"}} />
      </Link>

      <div>
        <button id="boutonMobil" className="btn " onClick={handleSearchToggle}>
          <img
            src={IconsSearchSvg}
            alt="Search Icon"
            style={{ width: "32px", height: "32px" }}
          />
        </button>
        {isSearchOpen && (
          <form
            onSubmit={handleSearch}
            className={`d-flex justify-content-end search-bar ${
              isSearchOpen ? "search-open" : ""
            }`}
          >
            <div className="input-group d-flex">
              <input
                type="text"
                className="form-control input"
                placeholder="Rechercher..."
                value={searchTerm}
                onChange={(e) => setSearchTerm(e.target.value)}
              />
              <div className="input-group-append">
```



```
                <button type="button" className="btn" onClick={handleSearch}>
                  <img
                    src={IconsSearchSvg}
                    alt="Search Icon"
                    style={{ width: "24px", height: "24px" }}
                  />
                </button>
              </div>
            </div>
          </form>
        )}

        <button
          className=" navbar-toggler  "
          type="button"
          onClick={handleToggle}
        >
          <img
            src={IconsMenuSvg}
            alt="Menu Icon"
            style={{ width: "32px", height: "32px" }}
          />
        </button>

        {isNavOpen && (
          <nav className={`navbar-flex-column  ${isNavOpen ? "nav-open" : ""}`}>
            <Link className="nav-link" to="/batiment" onClick={handleToggle}>
              Bâtiment
            </Link>
            <Link className="nav-link" to="/services" onClick={handleToggle}>
              Services
            </Link>
            <Link className="nav-link" to="/fabrication" onClick={handleToggle}>
              Fabrication
            </Link>
            <Link
              className="nav-link"
              to="/alimentation"
              onClick={handleToggle}
            >
              Alimentation
            </Link>
          </nav>
        )}
      </div>
    </header>
  );
};

export default Header;
```

En premier temps, j'ai donc importer certains module nécessaire à React, tels que « useState » , « useEffect » pour la gestion des états et des effets secondaires, ainsi que « Link » de React Router pour la navigation entre les différentes pages de l'application.

En deuxième temps, j'ai importer des images qui seront utilisées dans le composant, notamment le logo de l'entrep des icônes de menu et de recherche.

En troisième temps, j'ai déclaré le composant Header qui est une fonctionnalité de React qui prend en paramètres les fonctions et les états liés à la recherche. Il renvoie un élément JSX qui représente l'en tête de l'application.

En quatrième temps, j'ai utilisé deux états locaux, « isNavOpen » et « isSearchOpen » pour gérer l'ouverture Et la fermeture de la navigation et de la barre en fonction de la largeur de la fenêtre.

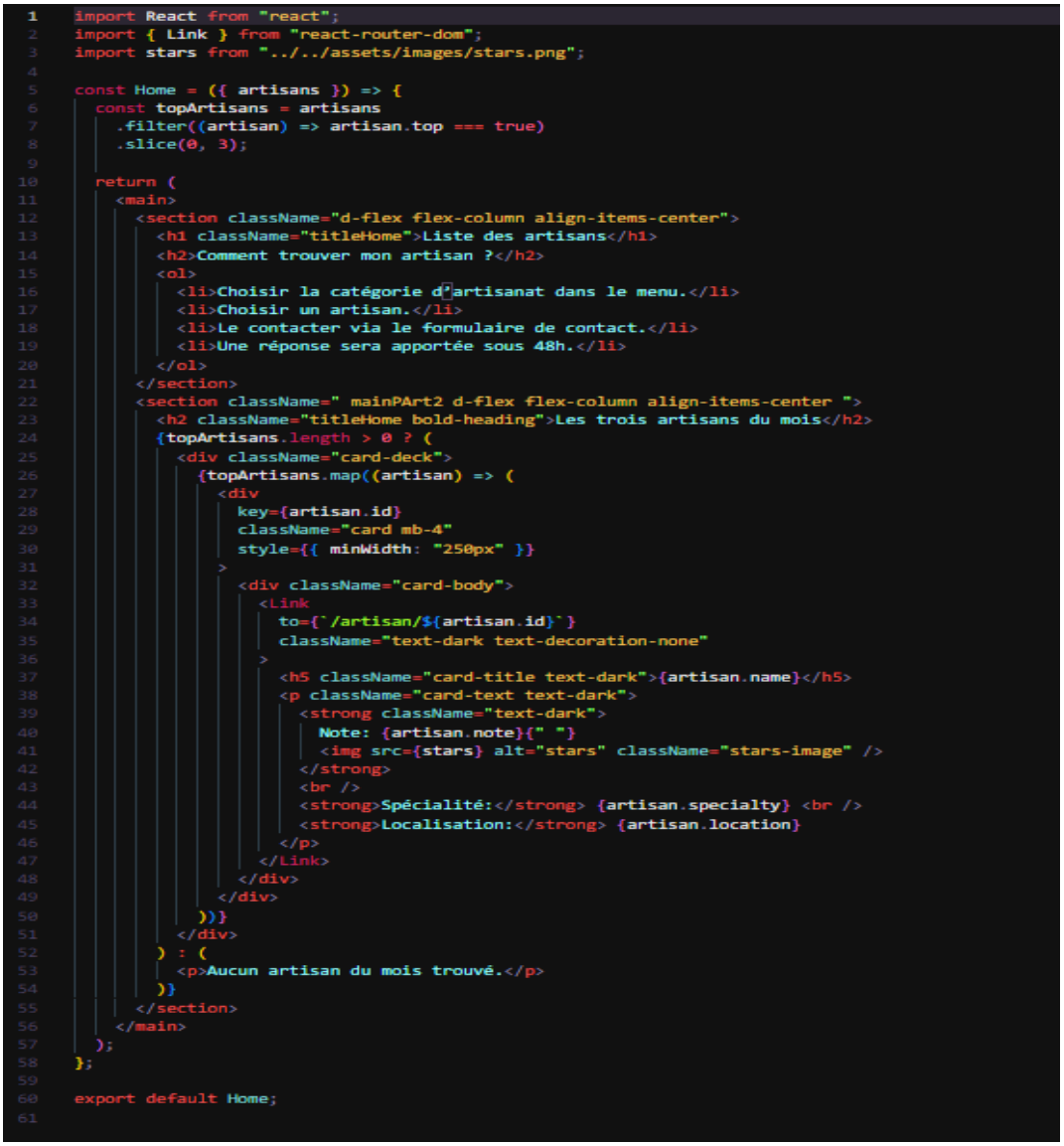
En cinquième temps, j'ai fait la gestion des événements pour gérer les interactions de l'utilisateur, telles que la recherche, le basculement de la visibilité de la barre de recherche et de la navigation en fonction de la taille de l'écran.

En sixième temps, j'ai mis en place un effet secondaire pour gérer le redimensionnement de la fenêtre. Cela garantit que l'état de la navigation et de la barre de recherche est mis à jour en temps réel lorsque la taille de la fenêtre est modifiée par l'utilisateur.

En septième temps, j'ai retourné un élément JSX représentant l'en-tête de l'application. Cet en-tête comprend des éléments tels que le logo de l'entreprise, des boutons de recherche et de menu, ainsi que des liens de navigation vers différentes pages de l'application. Ces éléments sont conditionnellement affichés en fonction de l'état des variables « isNavOpen » et « isSearchOpen ».

Pour finir le composant Header est ensuite exporté pour être utilisé dans d'autres parties de l'application.

Home (Page d'accueil)



```jsx
import React from "react";
import { Link } from "react-router-dom";
import stars from "../../assets/images/stars.png";

const Home = ({ artisans }) => {
  const topArtisans = artisans
    .filter((artisan) => artisan.top === true)
    .slice(0, 3);

  return (
    <main>
      <section className="d-flex flex-column align-items-center">
        <h1 className="titleHome">Liste des artisans</h1>
        <h2>Comment trouver mon artisan ?</h2>
        <ol>
          <li>Choisir la catégorie d'artisanat dans le menu.</li>
          <li>Choisir un artisan.</li>
          <li>Le contacter via le formulaire de contact.</li>
          <li>Une réponse sera apportée sous 48h.</li>
        </ol>
      </section>
      <section className=" mainPArt2 d-flex flex-column align-items-center ">
        <h2 className="titleHome bold-heading">Les trois artisans du mois</h2>
        {topArtisans.length > 0 ? (
          <div className="card-deck">
            {topArtisans.map((artisan) => (
              <div
                key={artisan.id}
                className="card mb-4"
                style={{ minWidth: "250px" }}
              >
                <div className="card-body">
                  <Link
                    to={`/artisan/${artisan.id}`}
                    className="text-dark text-decoration-none"
                  >
                    <h5 className="card-title text-dark">{artisan.name}</h5>
                    <p className="card-text text-dark">
                      <strong className="text-dark">
                        Note: {artisan.note}{" "}
                        <img src={stars} alt="stars" className="stars-image" />
                      </strong>
                      <br />
                      <strong>Spécialité:</strong> {artisan.specialty} <br />
                      <strong>Localisation:</strong> {artisan.location}
                    </p>
                  </Link>
                </div>
              </div>
            ))}
          </div>
        ) : (
          <p>Aucun artisan du mois trouvé.</p>
        )}
      </section>
    </main>
  );
};

export default Home;
```

En premier temps, le code commence par importer les modules nécessaires à React, tels que « React » et « Link » de « React Router » pour la navigation entre les différentes pages de l'application. Il importe également l'image des étoiles qui sera utilisée dans la représentation graphique des notes des artisans.

En deuxième temps, le composant Home est défini. C'est une fonction de React qui prend en paramètre « artisans », représentant les données des artisans. Le composant renvoie un élément JSX qui représente la page d'accueil de l'application.

En troisième temps, le composant filtre les artisans pour ne garder que ceux qui sont considérés comme les meilleurs, en se basant sur la propriété « top » définie sur true. Il sélectionne ensuite les trois premiers artisans de cette liste pour les afficher comme les meilleurs artisans du mois.

En quatrième temps, le rendu JSX du composant Home est défini. Il contient deux sections principales :

- La première section présente un titre principal, un sous-titre et une liste d'étapes pour guider l'utilisateur dans la recherche d'un artisan.
- La deuxième section affiche les trois meilleurs artisans du mois sous forme de cartes, avec leur nom, leur note, leur spécialité et leur localisation.

En cinquième temps, le composant vérifie s'il y a des artisans à afficher. S'il n'y en a pas, un message indiquant « Aucun artisan du mois trouvé » est affiché à la place.

En sixième temps, le composant Home est exporté à la fin du fichier pour pouvoir être utilisé dans d'autres parties de l'application.

Alimentation (Page de service)

```jsx
import React from "react";
import { Link } from "react-router-dom";
import stars from "../../assets/images/stars.png";

const Alimentation = ({ artisans, searchResults }) => {
  const displayArtisans =
    searchResults && searchResults.length > 0 ? searchResults : artisans;

  const alimentationArtisans = displayArtisans.filter(
    (artisan) => artisan.category === "Alimentation"
  );

  return (
    <main>
      <section className="d-flex flex-column align-items-center">
        <h1 className="titleHome">Artisans Alimentation</h1>
      </section>
      <section className="mainPArt2 d-flex flex-column align-items-center">
        {alimentationArtisans.length > 0 ? (
          <div className="card-deck">
            {alimentationArtisans.map((artisan) => (
              <div
                key={artisan.id}
                className="card mb-4"
                style={{ minWidth: "250px" }}
              >
                <div className="card-body">
                  <Link
                    to={`/artisan/${artisan.id}`}
                    className="text-dark text-decoration-none"
                  >
                    <h5 className="card-title text-dark">{artisan.name}</h5>
                    <p className="card-text text-dark">
                      <strong className="text-dark">
                        Note: {artisan.note}{" "}
                        <img src={stars} alt="stars" className="stars-image" />
                      </strong>{" "}
                      <br />
                      <strong>Spécialité:</strong> {artisan.specialty} <br />
                      <strong>Localisation:</strong> {artisan.location}
                    </p>
                  </Link>
                </div>
              </div>
            ))}
          </div>
        ) : (
          <p>Aucun artisan Alimentation trouvé.</p>
        )}
      </section>
    </main>
  );
};

export default Alimentation;
```

En premier temps, le code commence par importer les modules nécessaires à React, à savoir « React » et le composant « Link » de « React Router » pour la navigation entre les différentes pages de l'application. Il importe également l'image des étoiles qui sera utilisée pour représenter les notes des artisans.

En deuxième temps, le composant fonctionnel « Alimentation » est défini. Ce composant représente la page dédiée aux artisans spécialisés dans la catégorie « Alimentation » . Il prend en paramètres « artisans » et « searchResults », où « artisans » représente la liste complète des artisans et « searchResults » contient les résultats de recherche.

En troisième temps, le composant filtre les artisans en fonction de la catégorie « Alimentation » pour les afficher spécifiquement dans cette section.

En quatrième temps, le rendu JSX du composant « Alimentation » est défini. Il consiste en un élément <main> qui contient deux sections principales :

- La première section présente un titre « Artisans Alimentation ».
- La deuxième section affiche les cartes des artisans de la catégorie « Alimentation ». Chaque carte affiche le nom de l'artisan, sa note avec une représentation graphique sous forme d'étoiles, sa spécialité et sa localisation.

En cinquième temps, le composant effectue une vérification pour s'assurer qu'il y a des artisans à afficher. S'il n'y en a pas, un message "Aucun artisan Alimentation trouvé" est affiché à la place.

En sixième temps, le composant Alimentation est exporté à la fin du fichier pour pouvoir être utilisé dans d'autres parties de l'application.

Artisan (Page d'artisan)

```jsx
import React, { useState } from "react";
import { useParams } from "react-router-dom";
import artisansData from "../../datas/datas.json";
import stars from "../../assets/images/stars.png";

const ArtisanDetail = () => {
  const { id } = useParams();
  const artisan = artisansData.find((a) => a.id === id);

  const [contactForm, setContactForm] = useState({
    name: "",
    mail: "",
    tel: "",
    subject: "",
    message: "",
  });

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setContactForm({ ...contactForm, [name]: value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
  };

  if (!artisan) {
    return <p>Artisan non trouvé.</p>;
  }

  return (
    <main className="d-flex flex-column align-items-center ">
      <div className="card w-50 mt-5">
        <div className="card-body">
          <h1>{artisan.name}</h1>

          <p className="text-dark">
            <strong>Note:</strong> {artisan.note}{" "}
            <img src={stars} alt="stars" className="stars-image" />
          </p>
          <p>
            <strong>Spécialité:</strong> {artisan.specialty}
          </p>
          <p>
            <strong>Localisation:</strong> {artisan.location}
          </p>
          <p>
            <strong>A propos:</strong> {artisan.about}
          </p>
          <p>
            <strong>Catégorie:</strong> {artisan.category}
          </p>
          <p>
            <strong>Site Web:</strong>{" "}
            <a href={artisan.website}>{artisan.website}</a>
          </p>
        </div>
      </div>
      <div className="mainPArt2 d-flex flex-column align-items-center w-100">
        <div className="card w-50 mt-5 mb-5">
          <div className="card-body">
            <h2 className="card-title">Formulaire de contact</h2>
            <form onSubmit={handleSubmit}>
              <div className="form-group">
                <label htmlFor="name">Nom :</label>
                <input
                  type="text"
                  className="form-control"
                  id="name"
                  name="name"
                  value={contactForm.name}
                  onChange={handleInputChange}
```

```
                />
              </div>

              <div className="form-group">
                <label htmlFor="mail">Mail :</label>
                <input
                  type="text"
                  className="form-control"
                  id="mail"
                  name="mail"
                  value={contactForm.mail}
                  onChange={handleInputChange}
                />
              </div>

              <div className="form-group">
                <label htmlFor="tel">Numéro de téléphone :</label>
                <input
                  type="text"
                  className="form-control"
                  id="tel"
                  name="tel"
                  value={contactForm.tel}
                  onChange={handleInputChange}
                />
              </div>

              <div className="form-group">
                <label htmlFor="subject">Objet :</label>
                <input
                  type="text"
                  className="form-control"
                  id="subject"
                  name="subject"
                  value={contactForm.subject}
                  onChange={handleInputChange}
                />
              </div>

              <div className="form-group">
                <label htmlFor="message">Message :</label>
                <textarea
                  className="form-control"
                  id="message"
                  name="message"
                  value={contactForm.message}
                  onChange={handleInputChange}
                />
              </div>

              <button type="submit" className="boutonForm btn btn-primary mt-3">
                Envoyer
              </button>
            </form>
          </div>
        </div>
      </div>
    </main>
  );
};

export default ArtisanDetail;
```

En premier temps, le code importe les modules nécessaires à React, notamment « React » et « useState » pour gérer les états. De plus, il importe le hook « useParams » de « React Router » pour récupérer les paramètres de l'URL. Enfin, il importe les données des artisans depuis un fichier JSON et l'image des étoiles qui sera utilisée pour représenter les notes des artisans.

En deuxième temps, le composant fonctionnel « ArtisanDetail » est défini. Ce composant représente la page détaillée d'un artisan. Il utilise le hook « useParams » pour récupérer l'ID de l'artisan à partir des paramètres de l'URL, puis recherche l'artisan correspondant dans les données des artisans.

En troisième temps, le composant définit un état local « contactForm » pour stocker les données du formulaire de contact. Cet état initial est un objet contenant des champs vides pour le nom, le mail, le numéro de téléphone, le sujet et le message.

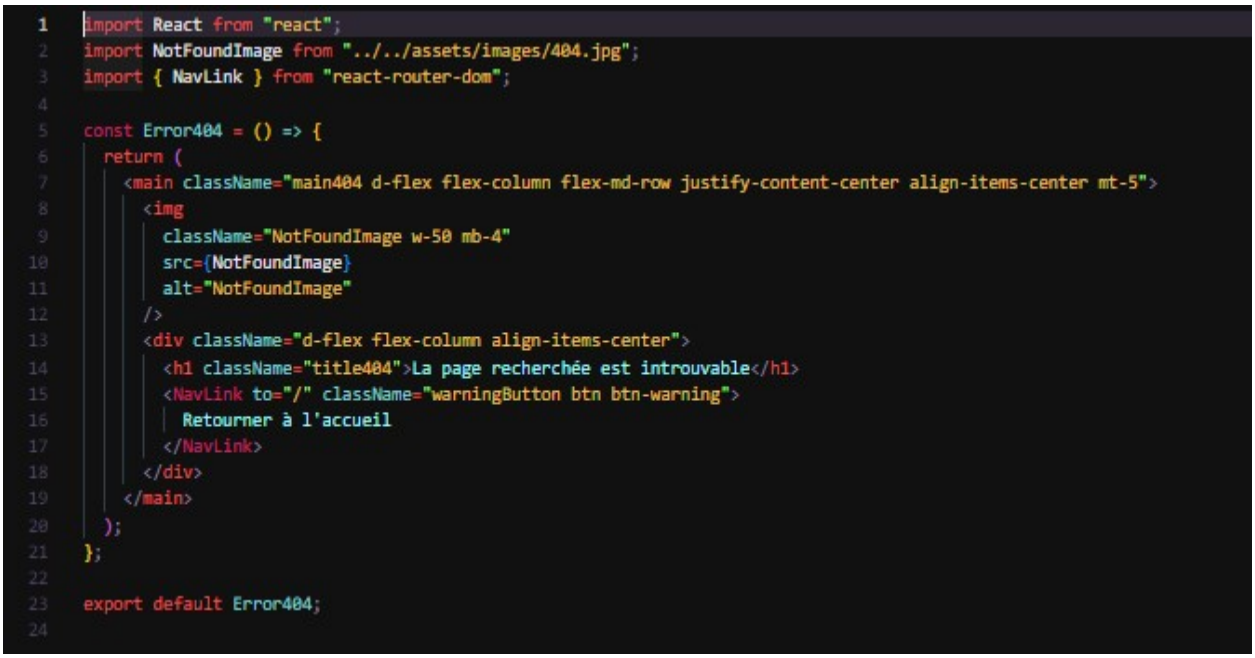
En quatrième temps, le composant définit des fonctions pour gérer les changements dans les champs du formulaire et pour soumettre le formulaire. La fonction « handleInputChange » met à jour l'état « contactForm » à chaque changement dans un champ du formulaire, tandis que « handleSubmit » est appelée lors de la soumission du formulaire.

En cinquième temps, le composant vérifie si l'artisan recherché n'est pas trouvé, auquel cas il affiche un message "Artisan non trouvé".

En sixième temps, le rendu JSX du composant « ArtisanDetail » est défini. Il consiste en un élément <main> contenant deux sections principales :

- La première section affiche les détails de l'artisan, tels que son nom, sa note, sa spécialité, sa localisation, etc.
- La deuxième section contient un formulaire de contact avec des champs pour le nom, le mail, le numéro de téléphone, l'objet et le message. Ce formulaire est soumis à l'aide de la fonction « handleSubmit. »

Enfin, le composant « ArtisanDetai »l est exporté pour être utilisé dans d'autres parties de l'application.



```jsx
import React from "react";
import NotFoundImage from "../../assets/images/404.jpg";
import { NavLink } from "react-router-dom";

const Error404 = () => {
  return (
    <main className="main404 d-flex flex-column flex-md-row justify-content-center align-items-center mt-5">
      <img
        className="NotFoundImage w-50 mb-4"
        src={NotFoundImage}
        alt="NotFoundImage"
      />
      <div className="d-flex flex-column align-items-center">
        <h1 className="title404">La page recherchée est introuvable</h1>
        <NavLink to="/" className="warningButton btn btn-warning">
          Retourner à l'accueil
        </NavLink>
      </div>
    </main>
  );
};

export default Error404;
```

En premier temps, le code définit le composant fonctionnel « Error404 » . Ce composant représente la page affichée lorsqu'une URL est introuvable (erreur 404).

En deuxième temps, le rendu JSX du composant « Error404 » est défini. Il consiste en un élément <main> qui contient deux éléments : une image de la page 404 et un conteneur de texte et de lien.

En troisième temps, l'image de la page 404 est affichée. Elle est importée à partir d'une source externe (« NotFoundImage ») et contient un texte alternatif décrivant l'image.

En quatrième temps, le conteneur de texte et de lien est défini. Il contient un titre indiquant que la page est introuvable, ainsi qu'un lien pour retourner à l'accueil de l'application. Ce lien utilise le composant « NavLink » de « react-router-dom » pour garantir une navigation efficace au sein de l'application.

En cinquième temps, le composant « Error404 » est exporté pour pouvoir être utilisé dans d'autres parties de l'application.

Footer

```jsx
import React from "react";
import { Link } from "react-router-dom";

const Footer = () => {
  return (
    <footer className="d-flex flex-column  align-items-center ">
      <h3 className="h3-contact">Contact</h3>
      <p>
        - 101 cours Charlemagne <br />
        - CS 20033 <br />
        - 69269 LYON CEDEX 02 <br />
        - France <br />
        <a className="tel" href="tel:+330426734000">
          - +33 (0)4 26 73 40 00
        </a>
      </p>

      <div className="barre-centree"></div>
      <div>
        <ul className="legalMentions d-flex justify-content-center">
          <li>
            <Link to="/legal/mentions" className="legal-mentions">
              Mentions Légales
            </Link>
          </li>
          <li>
            <Link to="/legal/personalData" className="legal-mentions">
              Données Personnelles
            </Link>
          </li>
          <li>
            <Link to="/legal/accessibility" className="legal-mentions">
              Accessibilité
            </Link>
          </li>
          <li>
            <Link to="/legal/cookies" className="legal-mentions">
              Cookies
            </Link>
          </li>
        </ul>
      </div>
    </footer>
  );
};

export default Footer;

```

En premier temps, le code importe les modules nécessaires à React, notamment « React » et le composant « Link » de « React Router ».

En deuxième temps, le composant fonctionnel « Footer » est défini. Ce composant représente le pied de page de l'application, qui contient les informations de contact et les liens légaux.

En troisième temps, le rendu JSX du composant « Footer » est défini. Il consiste en un élément <footer> qui contient :

- Un titre « Contact » indiquant le début de la section d'informations de contact.
- Les informations de contact telles que l'adresse postale et le numéro de téléphone.
- Une barre centrale pour séparer visuellement les informations de contact des liens légaux.
- Une liste de liens légaux, représentés par des liens vers différentes pages légales de l'application.

En quatrième temps, les liens vers les différentes pages légales sont définis à l'aide du composant « Link » de « React Router », qui garantit une navigation efficace au sein de l'application.

En cinquième temps, le composant « Footer » est exporté pour pouvoir être utilisé dans d'autres parties de l'application.

Cascading Style Sheets (CSS)

index.css

```css
/* --------Fonts------- */
/* ------------------- */

@font-face {
  src: url("./assets/fonts/GraphikRegular.otf");
  font-family: "Graphik";
}
@font-face {
  src: url("./assets/fonts/GraphikRegularItalic.otf");
  font-family: "Graphik Italic";
}
@font-face {
  src: url("./assets/fonts/GraphikLight.otf");
  font-family: "Graphik Light";
}
@font-face {
  src: url("./assets/fonts/GraphikLightItalic.otf");
  font-family: "Graphik Light Italic";
}
@font-face {
  src: url("./assets/fonts//GraphikBold.otf");
  font-family: "Graphik Bold";
}
@font-face {
  src: url("./assets/fonts/GraphikBoldItalic.otf");
  font-family: "Graphik Bold Italic";
}

body {
  font-family: "Graphik", sans-serif;
}

/* --------Header------- */
/* ------------------- */

header {
  padding-left: 10%;
  padding-right: 10%;
  box-shadow: 0 0.2rem 0.6rem rgba(0, 0, 0, 0.0793474);
  padding-bottom: 1rem;
}

.navbar-flex-column {
  position: absolute;
  left: 0;
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  justify-content: flex-start;
  width: 100%;
  height: 10%;
  background-color: white;
  padding-left: 5%;
}

.input-group {
  border-right: 1px solid #0074c7;
  border-bottom: 1px solid #0074c7;
  display: flex;
  align-items: center;
}

.input {
  border: none;
  height: 32px;
}

.btn-primary {
  border: none;
  color: white;
  height: 32px;
  border-radius: 4px;
  background-color: transparent;
}

.search-bar {
  position: absolute;
  left: 0;
  width: 100%;
  background-color: #fff;
  transition: top 0.3s;
  padding-left: 2.5%;
  padding-right: 2.5%;
}

@media screen and (min-width: 1024px) {
  header {
```

```
    padding-left: 120px;
    padding-right: 120px;
  }

  .navbar-toggler,
  #boutonMobil {
    display: none;
  }

  .search-bar {
    position: relative;
  }

  .navbar-flex-column {
    position: relative;
    display: flex;
    justify-content: flex-start;
    flex-direction: row;
    flex-wrap: nowrap;
    justify-content: flex-end;
    margin-top: 1rem;
  }

  .nav-link {
    margin-left: 1rem;
  }

  .input-group {
    margin-top: 1rem;
    width: 70%;
  }
}

/* --------Footer------- */
/* -------------------- */

footer {
  background-color: #0074c7;
  color: white;
  padding-top: 2rem;
  padding-bottom: 2rem;
}

.legal-mentions {
  text-decoration: none;
  color: white;
  margin-left: 0.5rem;
  margin-right: 0.5rem;
}

.tel {
  text-decoration: underline;
  color: white;
}

.barre-centree {
  width: 75%;
  margin: 0 auto;
  height: 2px;
  background-color: white;
  margin-top: 1rem;
  margin-bottom: 2rem;
}

.legalMentions {
  list-style: none;
  margin-left: 5%;
  margin-right: 5%;
  flex-wrap: wrap;
}

.h3-contact {
  font-family: Graphik Bold;
  text-transform: uppercase;
  text-decoration: underline;
}

/* --------Main------- */
/* -------------------- */

.titleHome {
  font-weight: bold;
```

```
  margin-top: 2rem;
  margin-bottom: 2rem;
  text-decoration: underline;
  color: #cd2c2e;
}

.mainPArt2 {
  background: linear-gradient(to bottom, rgb(255, 255, 255) 20%, #384050 100%;
}

.card-title {
  font-weight: bold;
  text-transform: uppercase;
  text-decoration: underline;
}

.card-deck {
  margin-bottom: 5rem;
}

@media screen and (min-width: 728px) {
  .card-deck {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
  }

  .card {
    margin: 3rem;
  }
}

@media screen and (min-width: 1024px) {
  .card-deck {
    display: flex;
    justify-content: space-around;
  }

  .card {
    margin: 5rem;
  }
}

/* --------404page------- */
/* ------------------- */

.warningButton {
  background-color: #cd2c2e;
  color: black;
  border: 1px solid black;
}

.main404 {
  margin-bottom: 2rem;
}

.title404 {
  margin-bottom: 2rem;
}

@media screen and (min-width: 768px) {
  .NotFoundImage {
    width: 200px !important;
  }
}

@media screen and (min-width: 1024px) {
  .NotFoundImage {
    width: 250px !important;
  }
}

.boutonForm {
  background-color: #82b86c;
}

.stars-image {
  width: 16px;
  height: auto;
  margin-top: -5px;
}
```

En premier temps, le code définit différentes polices à l'aide de « @font-face ». Chaque police est associée à un nom de famille de police spécifique.

En deuxième temps, des styles pour le corps de la page sont définis. La police par défaut pour le corps est définie comme « Graphik », avec une sauvegarde sur la police sans-serif.

En troisième temps, des styles pour l'en-tête de la page sont définis. Cela inclut des marges intérieures, une ombre portée et un style spécifique pour la barre de navigation en colonne.

En quatrième temps, des styles pour le pied de page de la page sont définis. Cela inclut la couleur de fond, la couleur du texte, les marges intérieures, et des styles spécifiques pour les liens légaux et le numéro de téléphone.

En cinquième temps, des styles pour la section principale de la page sont définis. Cela inclut des styles pour les titres principaux, les sections principales et les card.

En sixième temps, des styles pour la page 404 sont définis. Cela inclut des styles pour l'image de la page non trouvée, le titre de la page, et le bouton d'avertissement.

VALIDATEUR W3C

1. Info Trailing slash on void elements has no effect and interacts badly with unquoted attribute values.
   From line 2, column 38; to line 2, column 60
   `fr"><head><meta charset="utf-8"/><link`
2. Info Trailing slash on void elements has no effect and interacts badly with unquoted attribute values.
   From line 2, column 61; to line 2, column 98
   `="utf-8"/><link rel="icon" href="/favicon.ico"/><meta`
3. Info Trailing slash on void elements has no effect and interacts badly with unquoted attribute values.
   From line 2, column 99; to line 2, column 166
   `con.ico"/><meta name="viewport" content="width=device-width,initial-scale=1"/><meta`
4. Info Trailing slash on void elements has no effect and interacts badly with unquoted attribute values.
   From line 2, column 167; to line 2, column 210
   `scale=1"/><meta name="theme-color" content="#000000"/><meta`
5. Info Trailing slash on void elements has no effect and interacts badly with unquoted attribute values.
   From line 2, column 211; to line 2, column 286
   `#000000"/><meta name="description" content="Web site created using create-react-app"/><link`
6. Info Trailing slash on void elements has no effect and interacts badly with unquoted attribute values.
   From line 2, column 287; to line 2, column 336
   `act-app"/><link rel="apple-touch-icon" href="/logo192.png"/><link`
7. Info Trailing slash on void elements has no effect and interacts badly with unquoted attribute values.
   From line 2, column 337; to line 2, column 380
   `192.png"/><link rel="manifest" href="/manifest.json"/><title`

Document checking completed. No errors or warnings to show.