# Application de Gestion de Smartphones

**Auteurs** : WANE Couro, SADIO Mohamed

**Enseignant** : Monsieur Abdoulaye DIENG

**Établissement** : ESMT

**Niveau** : MPISI1

## 1. Introduction

Ce projet a pour objectif de développer une application web permettant de gérer un catalogue de smartphones. L'application est réalisée avec ReactJS pour le frontend et json-server pour simuler le backend. Elle inclut des fonctionnalités de gestion des rôles (Administrateur et Utilisateur) et les opérations CRUD sur les smartphones.

## 2. Environnement de Développement

• **Framework :** ReactJS pour le développement rapide d'interfaces modernes et réactives.

• **Backend :** json-server pour simuler une API REST sans avoir besoin de créer un vrai backend.
• **Librairies :**

-Axios : pour effectuer des requêtes HTTP.

-React Router DOM : pour la gestion des routes et de la navigation.

-TailwindCSS : pour le design rapide et responsive.

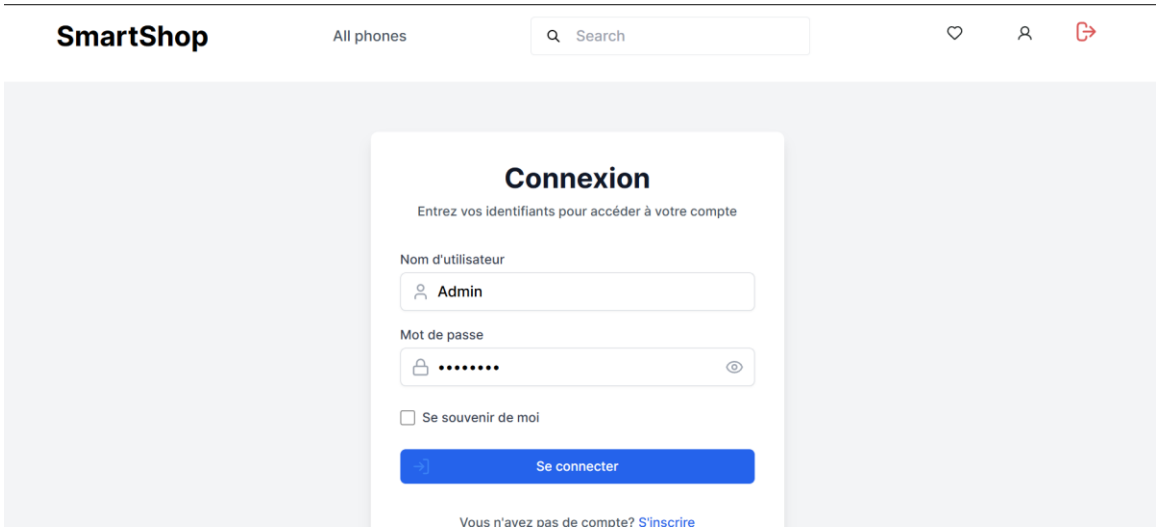
• **Outils :** VS Code pour le développement.

## 3. Interfaces et Codes Associés

### 3.1 Interface de Connexion

Cette interface permet à l'utilisateur de se connecter à l'application en fournissant son nom d'utilisateur et mot de passe.
Elle vérifie également le rôle de l'utilisateur pour lui attribuer les droits correspondants.

Capture d'écran :




Code associé :

• Fichier : Login.jsx

```
src > pages > LoginPage > Login.jsx > ...
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { Eye, EyeOff, User, Lock, LogIn } from 'lucide-react';

Tabnine | Edit | Explain
const Login = () => {
  const [nom, setNom] = useState('');
  const [motDePasse, setMotDePasse] = useState('');
  const [erreur, setErreur] = useState('');
  const [afficherMotDePasse, setAfficherMotDePasse] = useState(false);
  const navigate = useNavigate();

  // Fonction handleLogin originale, non modifiée
  const handleLogin = async (e) => {
    e.preventDefault();

    try {
      const response = await fetch(`http://localhost:3001/utilisateurs?nom=${nom}&motDePasse=${motDe
      const data = await response.json();

      if (data.length > 0) {
        const user = data[0];
        // Enregistre l'utilisateur dans le localStorage
        localStorage.setItem('utilisateurConnecte', JSON.stringify(user));
        navigate('/app');
      } else {
        setErreur('Nom ou mot de passe incorrect.');
      }
    } catch (err) {
      console.error(err);
      setErreur('Erreur lors de la connexion.');
```

src > pages > LoginPage > Login.jsx > ...

```
5   const Login = () => {
34    const basculerAffichageMotDePasse = () => {
35      setAfficherMotDePasse(!afficherMotDePasse);
36    };
37
38    return (
39      <div className="flex items-center justify-center min-h-screen bg-gray-100">
40        <div className="w-full max-w-md p-8 space-y-8 bg-white rounded-lg shadow-lg">
41          <div className="text-center">
42            <h2 className="text-3xl font-extrabold text-gray-900">Connexion</h2>
43            <p className="mt-2 text-sm text-gray-600">
44              Entrez vos identifiants pour accéder à votre compte
45            </p>
46          </div>
47
48          <form className="mt-8 space-y-6" onSubmit={handleLogin}>
49            <div className="space-y-4">
50              <div>
51                <label htmlFor="nom" className="block text-sm font-medium text-gray-700">
52                  Nom d'utilisateur
53                </label>
54                <div className="relative mt-1">
55                  <div className="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none
56                    <User className="w-5 h-5 text-gray-400" />
57                  </div>
58                  <input
59                    id="nom"
60                    name="nom"
61                    type="text"
62                    required
63                    className="block w-full pl-10 pr-3 py-2 border border-gray-300 rounded-md shadow-s
```

src > pages > LoginPage > Login.jsx > ...

```
5   const Login = () => {
64                    placeholder="Votre nom d'utilisateur"
65                    value={nom}
66                    onChange={(e) => setNom(e.target.value)}
67                  />
68                </div>
69              </div>
70
71              <div>
72                <label htmlFor="password" className="block text-sm font-medium text-gray-700">
73                  Mot de passe
74                </label>
75                <div className="relative mt-1">
76                  <div className="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none
77                    <Lock className="w-5 h-5 text-gray-400" />
78                  </div>
79                  <input
80                    id="password"
81                    name="password"
82                    type={afficherMotDePasse ? "text" : "password"}
83                    required
84                    className="block w-full pl-10 pr-10 py-2 border border-gray-300 rounded-md shadow-
85                    placeholder="Votre mot de passe"
86                    value={motDePasse}
87                    onChange={(e) => setMotDePasse(e.target.value)}
88                  />
89                  <div className="absolute inset-y-0 right-0 flex items-center pr-3">
90                    <button
91                      type="button"
92                      onClick={basculerAffichageMotDePasse}
93                      className="text-gray-400 hover:text-gray-500 focus:outline-none"
94                    >
```

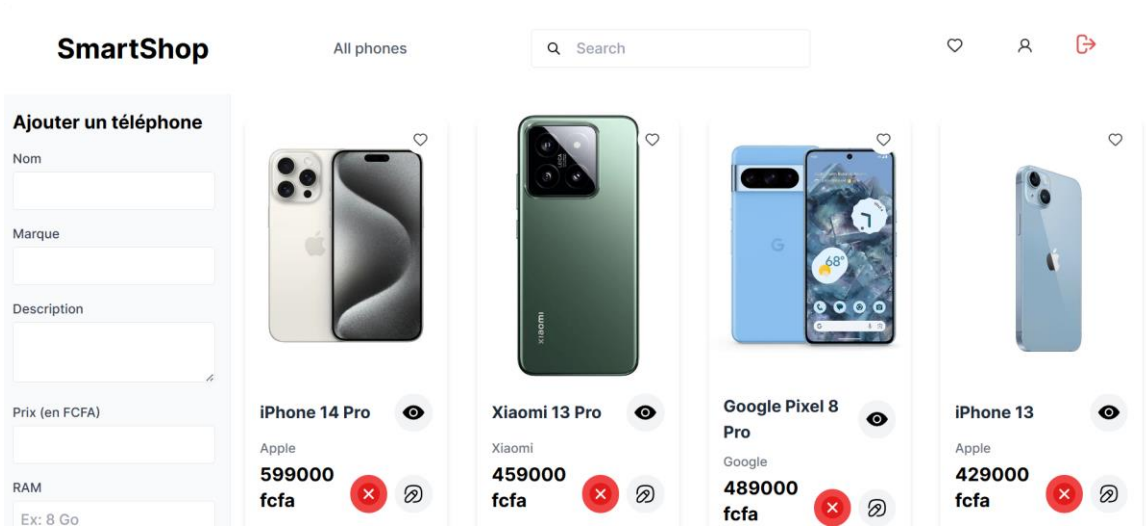
```
src > pages > LoginPage > Login.jsx > ...
  5   const Login = () => {
101                   </div>
102                 </div>
103               </div>
104             </div>
105
106             {erreur && (
107               <div className="p-3 text-sm text-red-600 bg-red-100 rounded-md">
108                 {erreur}
109               </div>
110             )}
111
112             <div className="flex items-center">
113               <input
114                 id="remember-me"
115                 name="remember-me"
116                 type="checkbox"
117                 className="w-4 h-4 text-blue-600 border-gray-300 rounded focus:ring-blue-500"
118               />
119               <label htmlFor="remember-me" className="block ml-2 text-sm text-gray-700">
120                 Se souvenir de moi
121               </label>
122             </div>
123
124             <div>
125               <button
126                 type="submit"
127                 className="group relative w-full flex justify-center py-2 px-4 border border-transpare
128               >
129                 <span className="absolute left-0 inset-y-0 flex items-center pl-3">
130                   <LogIn className="w-5 h-5 text-blue-500 group-hover:text-blue-400" />
```

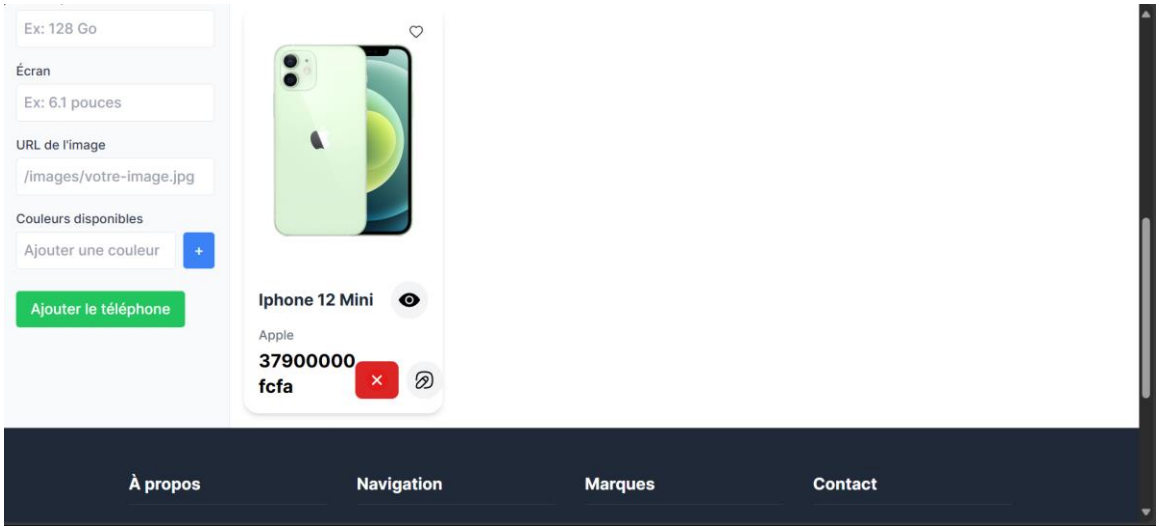
## 3.2 Interface de Liste des Smartphones(Admin)

Affiche la liste complète des smartphones disponibles.

- Les administrateurs disposent de boutons supplémentaires pour modifier ou supprimer des smartphones.

Capture d'écran :





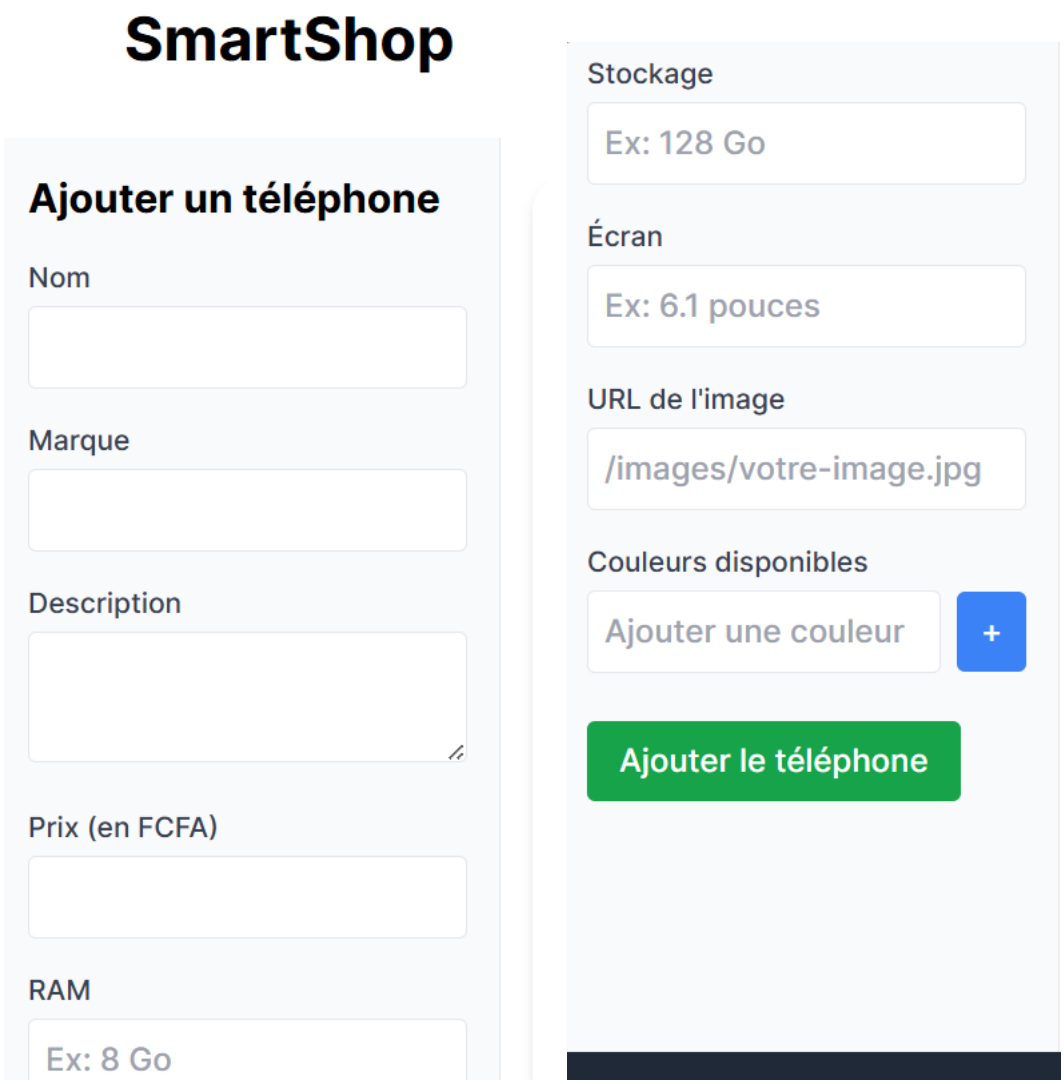

Code associé :

• Fichier : AllPhonePage.jsx


## 3.3 Interface d'Ajout de Smartphone

Accessible uniquement aux administrateurs, cette interface permet d'ajouter un nouveau smartphone dans le catalogue en remplissant un formulaire détaillé.

Capture d’écran :




Code associé :

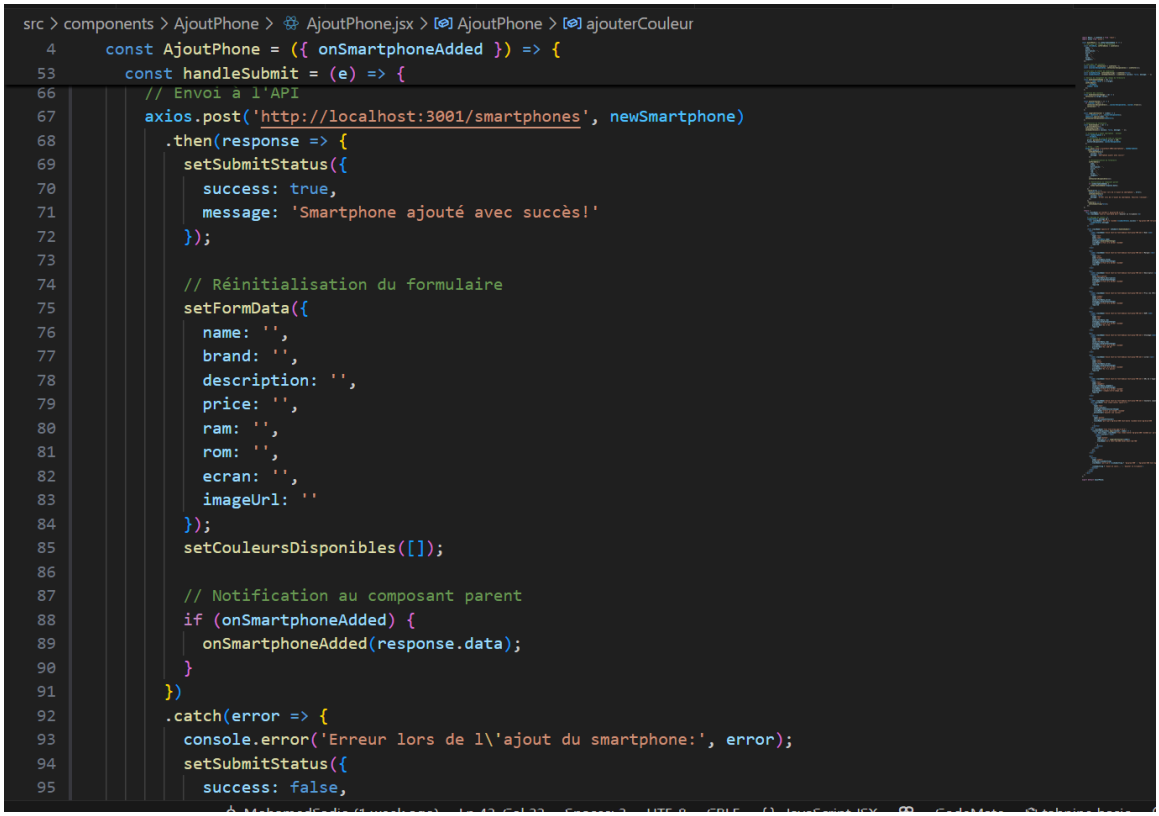
• Fichier : AjoutPhone.jsx

src > components > AjoutPhone > AjoutPhone.jsx > ...

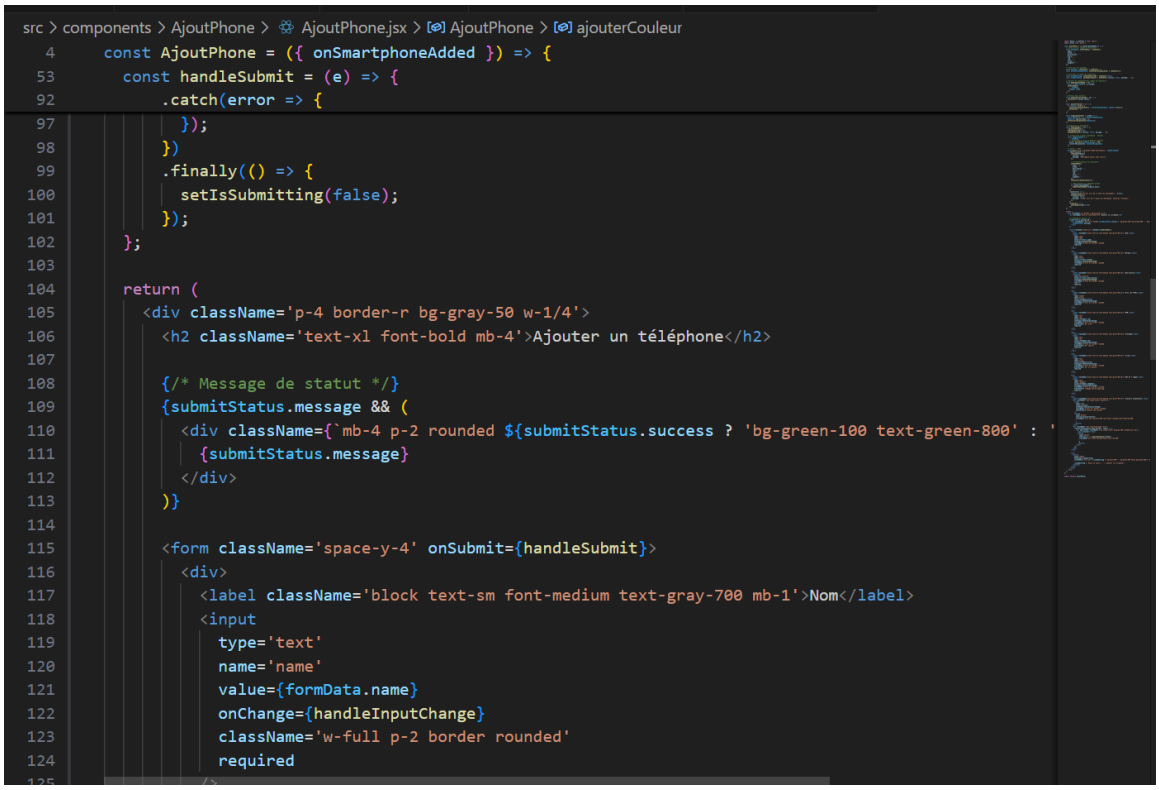
```jsx
import React, { useState } from 'react';
import axios from 'axios';

Tabnine | Edit | Explain
const AjoutPhone = ({ onSmartphoneAdded }) => {
  // État pour les champs du formulaire
  const [formData, setFormData] = useState({
    name: '',
    brand: '',
    description: '',
    price: '',
    ram: '',
    rom: '',
    ecran: '',
    imageUrl: ''
  });

  // État pour les couleurs
  const [couleur, setCouleur] = useState('');
  const [couleursDisponibles, setCouleursDisponibles] = useState([]);

  // État pour le statut de soumission
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [submitStatus, setSubmitStatus] = useState({ success: false, message: '' });

  // Gestion du changement des champs du formulaire
  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value
    });
```

src > components > AjoutPhone > AjoutPhone.jsx > ...

```jsx
import React, { useState } from 'react';
import axios from 'axios';

Tabnine | Edit | Explain
const AjoutPhone = ({ onSmartphoneAdded }) => {
  // État pour les champs du formulaire
  const [formData, setFormData] = useState({
    name: '',
    brand: '',
    description: '',
    price: '',
    ram: '',
    rom: '',
    ecran: '',
    imageUrl: ''
  });

  // État pour les couleurs
  const [couleur, setCouleur] = useState('');
  const [couleursDisponibles, setCouleursDisponibles] = useState([]);

  // État pour le statut de soumission
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [submitStatus, setSubmitStatus] = useState({ success: false, message: '' });

  // Gestion du changement des champs du formulaire
  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value
    });
```



```
src > components > AjoutPhone > AjoutPhone.jsx > AjoutPhone > ajouterCouleur
 4    const AjoutPhone = ({ onSmartphoneAdded }) => {
53      const handleSubmit = (e) => {
66        // Envoi à l'API
67        axios.post('http://localhost:3001/smartphones', newSmartphone)
68          .then(response => {
69            setSubmitStatus({
70              success: true,
71              message: 'Smartphone ajouté avec succès!'
72            });
73
74            // Réinitialisation du formulaire
75            setFormData({
76              name: '',
77              brand: '',
78              description: '',
79              price: '',
80              ram: '',
81              rom: '',
82              ecran: '',
83              imageUrl: ''
84            });
85            setCouleursDisponibles([]);
86
87            // Notification au composant parent
88            if (onSmartphoneAdded) {
89              onSmartphoneAdded(response.data);
90            }
91          })
92          .catch(error => {
93            console.error('Erreur lors de l\'ajout du smartphone:', error);
94            setSubmitStatus({
95              success: false,
```

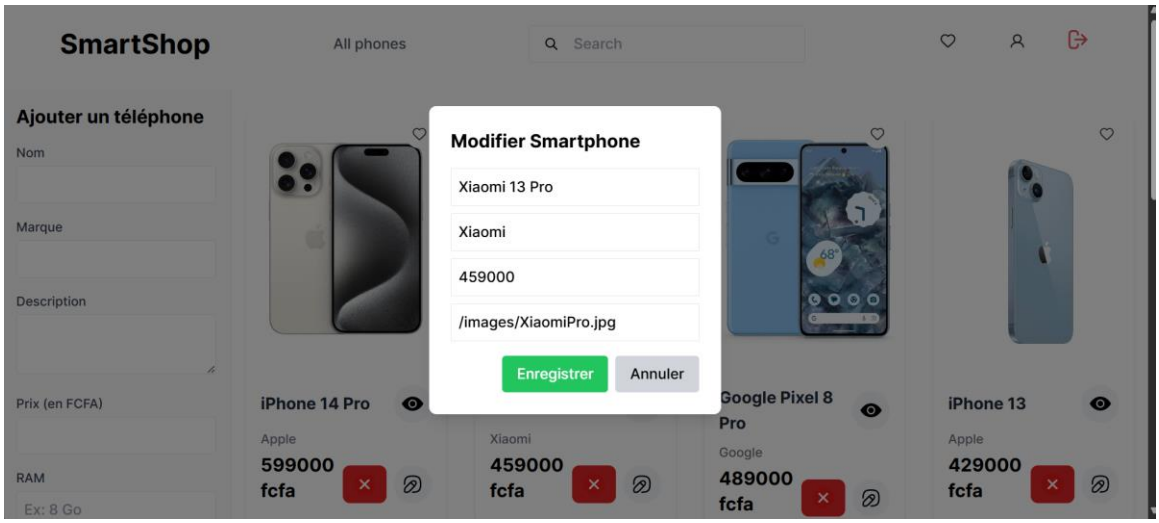


```
src > components > AjoutPhone > AjoutPhone.jsx > AjoutPhone > ajouterCouleur
 4    const AjoutPhone = ({ onSmartphoneAdded }) => {
53      const handleSubmit = (e) => {
92          .catch(error => {
97            });
98          })
99          .finally(() => {
100           setIsSubmitting(false);
101         });
102     };
103
104     return (
105       <div className='p-4 border-r bg-gray-50 w-1/4'>
106         <h2 className='text-xl font-bold mb-4'>Ajouter un téléphone</h2>
107
108         {/* Message de statut */}
109         {submitStatus.message && (
110           <div className={`mb-4 p-2 rounded ${submitStatus.success ? 'bg-green-100 text-green-800' : '
111             {submitStatus.message}
112           </div>
113         )}
114
115         <form className='space-y-4' onSubmit={handleSubmit}>
116           <div>
117             <label className='block text-sm font-medium text-gray-700 mb-1'>Nom</label>
118             <input
119               type='text'
120               name='name'
121               value={formData.name}
122               onChange={handleInputChange}
123               className='w-full p-2 border rounded'
124               required
```

## 3.4 Interface de Modification de Smartphone (Admin)

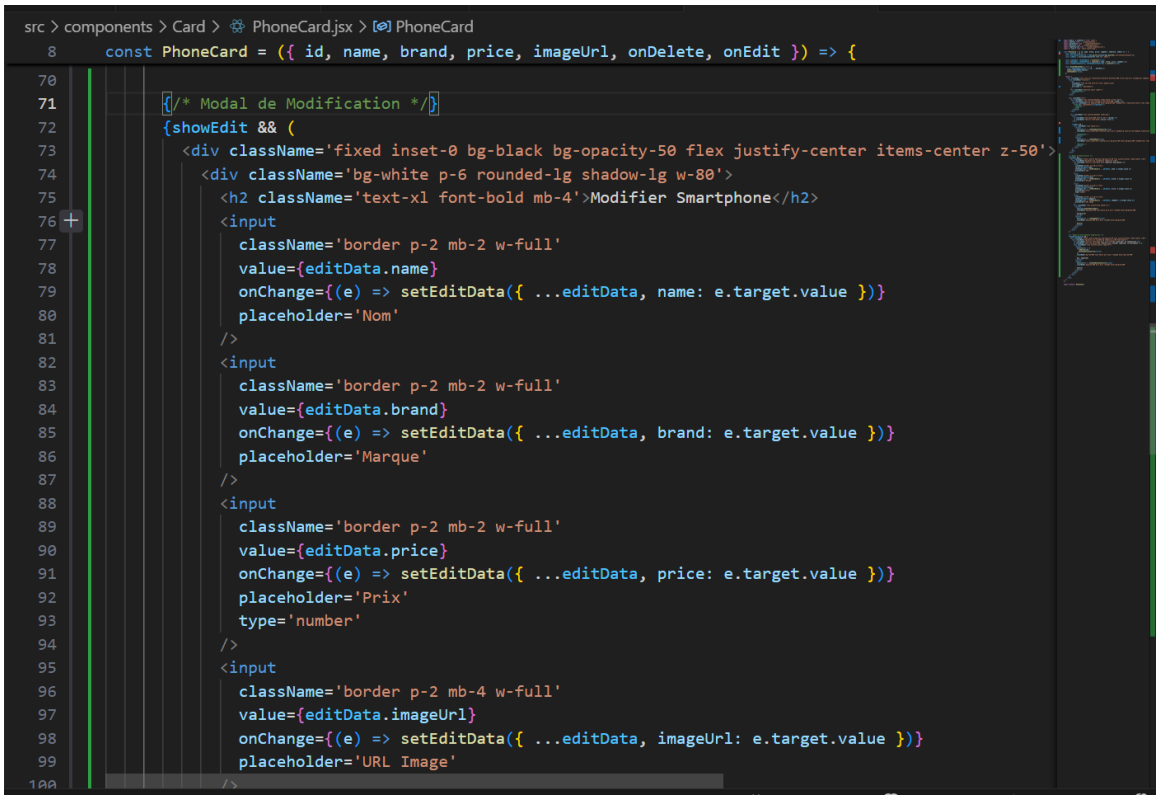
Permet à l'administrateur de mettre à jour les informations d'un smartphone existant. Les champs du formulaire sont préremplis avec les données actuelles, facilitant ainsi la modification.

Capture d'écran :




Code associé :

• Fichier : PhoneCard.jsx - Méthode de modification

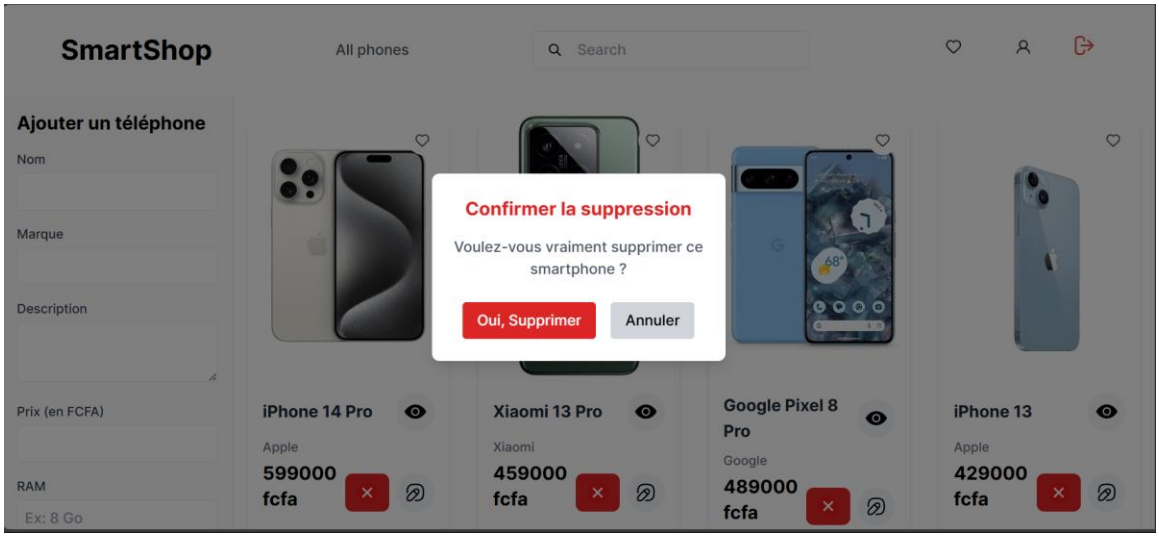


```
src > components > Card > PhoneCard.jsx > PhoneCard
8     const PhoneCard = ({ id, name, brand, price, imageUrl, onDelete, onEdit }) => {
88              <input
89                className='border p-2 mb-2 w-full'
90                value={editData.price}
91                onChange={(e) => setEditData({ ...editData, price: e.target.value })}
92                placeholder='Prix'
93                type='number'
94              />
95              <input
96                className='border p-2 mb-4 w-full'
97                value={editData.imageUrl}
98                onChange={(e) => setEditData({ ...editData, imageUrl: e.target.value })}
99                placeholder='URL Image'
100             />
101             <div className='flex justify-end space-x-2'>
102               <button
103                 onClick={handleEditSubmit}
104                 className='bg-green-500 text-white px-4 py-2 rounded hover:bg-green-600'
105               >
106                 Enregistrer
107               </button>
108               <button
109                 onClick={() => setShowEdit(false)}
110                 className='bg-gray-300 px-4 py-2 rounded hover:bg-gray-400'
111               >
112                 Annuler
113               </button>
114             </div>
115           </div>
116         </div>
117       )}
```

## 3.5 Interface de Suppression de Smartphone (Admin)

Cette interface permet aux administrateurs de supprimer définitivement un smartphone du catalogue.
Une alerte de confirmation est affichée avant la suppression pour éviter les erreurs.

Capture d'écran :




Code associé :

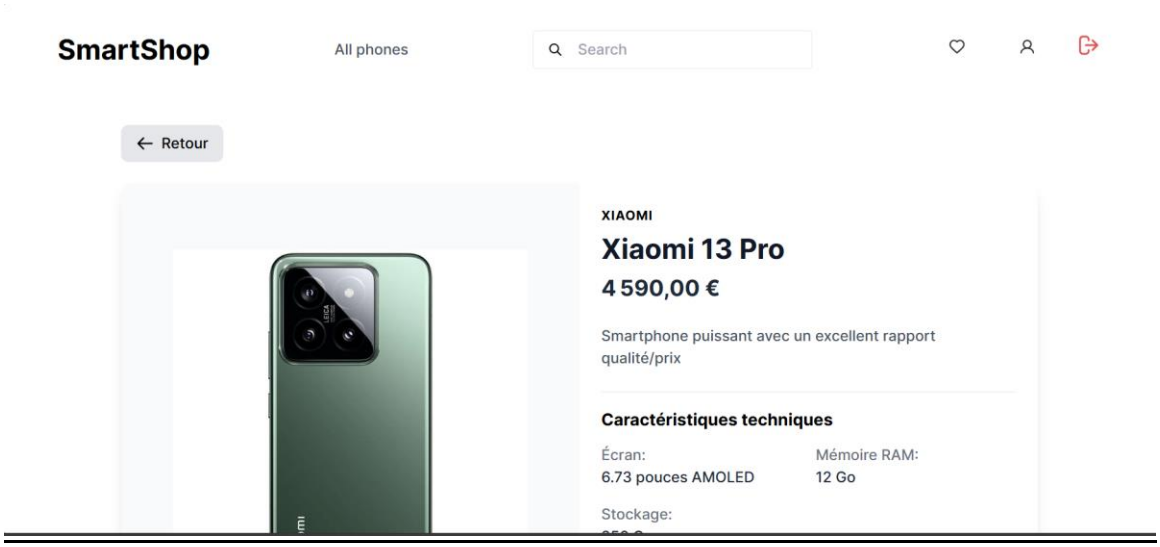
• Fichier : PhoneCard.jsx - Méthode de suppression

```
src > components > Card > PhoneCard.jsx > PhoneCard
8   const PhoneCard = ({ id, name, brand, price, imageUrl, onDelete, onEdit }) => {
118
119       {/*  Modal de Confirmation Suppression */}
120       {showDeleteConfirm && (
121         <div className='fixed inset-0 bg-black bg-opacity-50 flex justify-center items-center z-50'>
122           <div className='bg-white p-6 rounded-lg shadow-lg w-80 text-center'>
123             <h2 className='text-xl font-bold mb-4 text-red-600'>Confirmer la suppression</h2>
124             <p className='mb-6 text-gray-600'>Voulez-vous vraiment supprimer ce smartphone ?</p>
125             <div className='flex justify-center space-x-4'>
126               <button
127                 onClick={() => {
128                   onDelete(id);
129                   setShowDeleteConfirm(false);
130                 }}
131                 className='bg-red-600 text-white px-4 py-2 rounded hover:bg-red-700'
132               >
133                 Oui, Supprimer
134               </button>
135               <button
136                 onClick={() => setShowDeleteConfirm(false)}
137                 className='bg-gray-300 px-4 py-2 rounded hover:bg-gray-400'
138               >
139                 Annuler
140               </button>
141             </div>
142           </div>
143         </div>
144       )}
145     </div>
146   );
147 };
148
```

## 3.6 Interface de Détails de Smartphone

Permet à l'utilisateur de consulter les informations détaillées d'un smartphone sélectionné (prix, marque, RAM, ROM, couleurs disponibles...).

Capture d'écran :




Code associé :

• Fichier : PhoneDetailPage.jsx

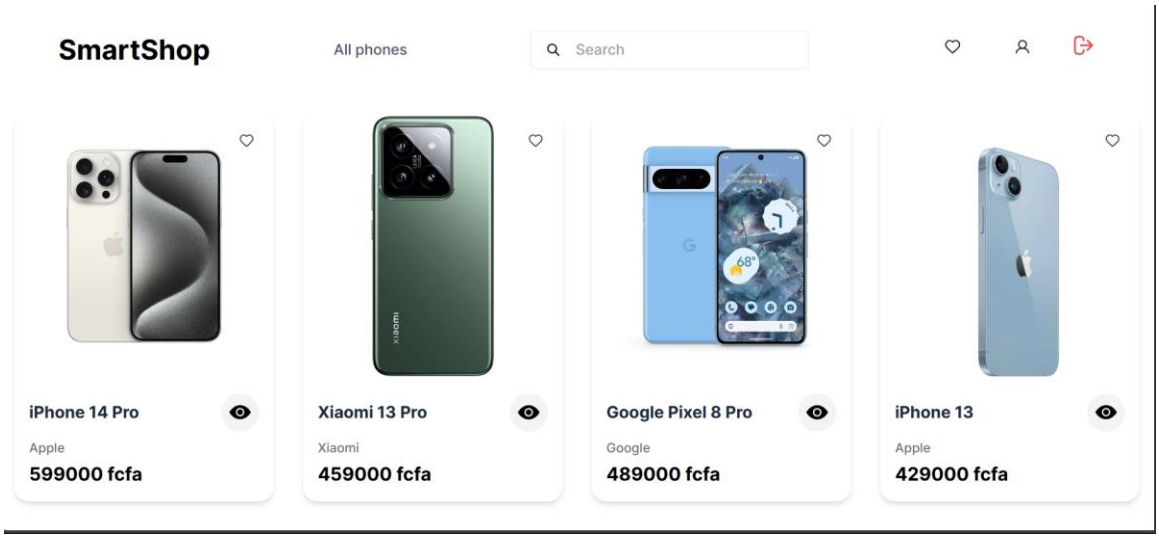
```
src > pages > PhoneDetails > PhoneDetailPage.jsx > ...
5     const PhoneDetailPage = () => {
29      }
30
31      // Si une erreur s'est produite
32      if (error) {
33        return (
34          <div className="flex flex-col items-center justify-center min-h-screen px-4 py-8">
35            <h2 className="text-2xl font-bold text-red-600 mb-4">Erreur</h2>
36            <p className="text-lg mb-6">Une erreur s'est produite lors du chargement des données.</p>
37            <button
38              onClick={() => navigate(-1)}
39              className="px-4 py-2 bg-blue-600 text-white rounded hover:bg-blue-700"
40            >
41              Retour
42            </button>
43          </div>
44        );
45      }
46
47      // Handle case when smartphone is not found
48      if (!smartphone) {
49        return (
50          <div className="flex flex-col items-center justify-center min-h-screen px-4 py-8">
51            <h2 className="text-2xl font-bold text-red-600 mb-4">Smartphone non trouvé</h2>
52            <p className="text-lg mb-6">Nous n'avons pas pu trouver un smartphone avec l'ID {phoneId}.</
53            <button
54              onClick={() => navigate(-1)}
55              className="px-4 py-2 bg-blue-600 text-white rounded hover:bg-blue-700"
56            >
57              Retour
58            </button>
59          </div>
```

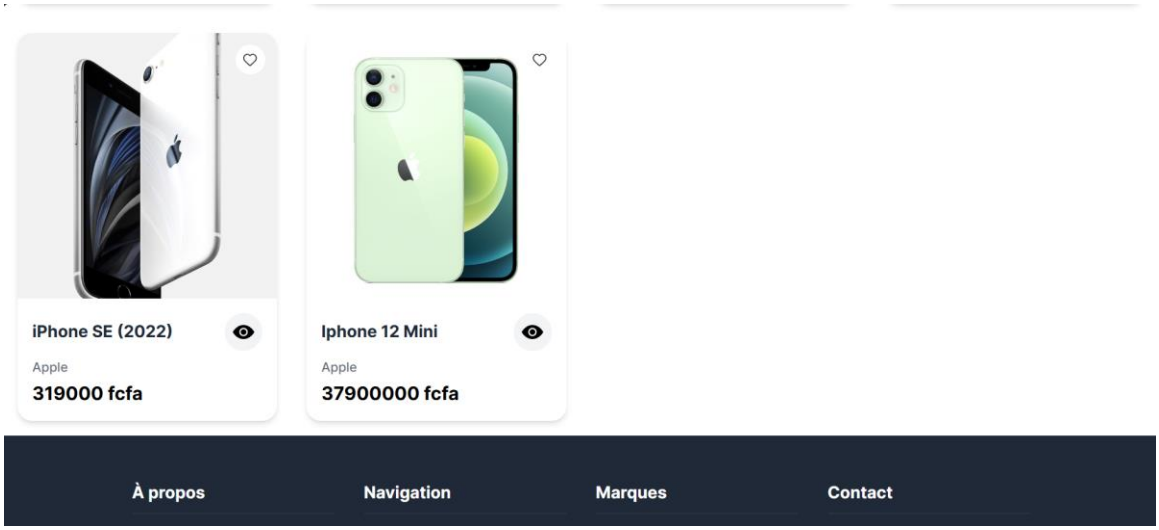
## 3.7 Interface de Consultation des Smartphones (Utilisateur)

Affiche la liste complète des smartphones disponibles.

- Les utilisateurs peuvent consulter les détails des téléphones.

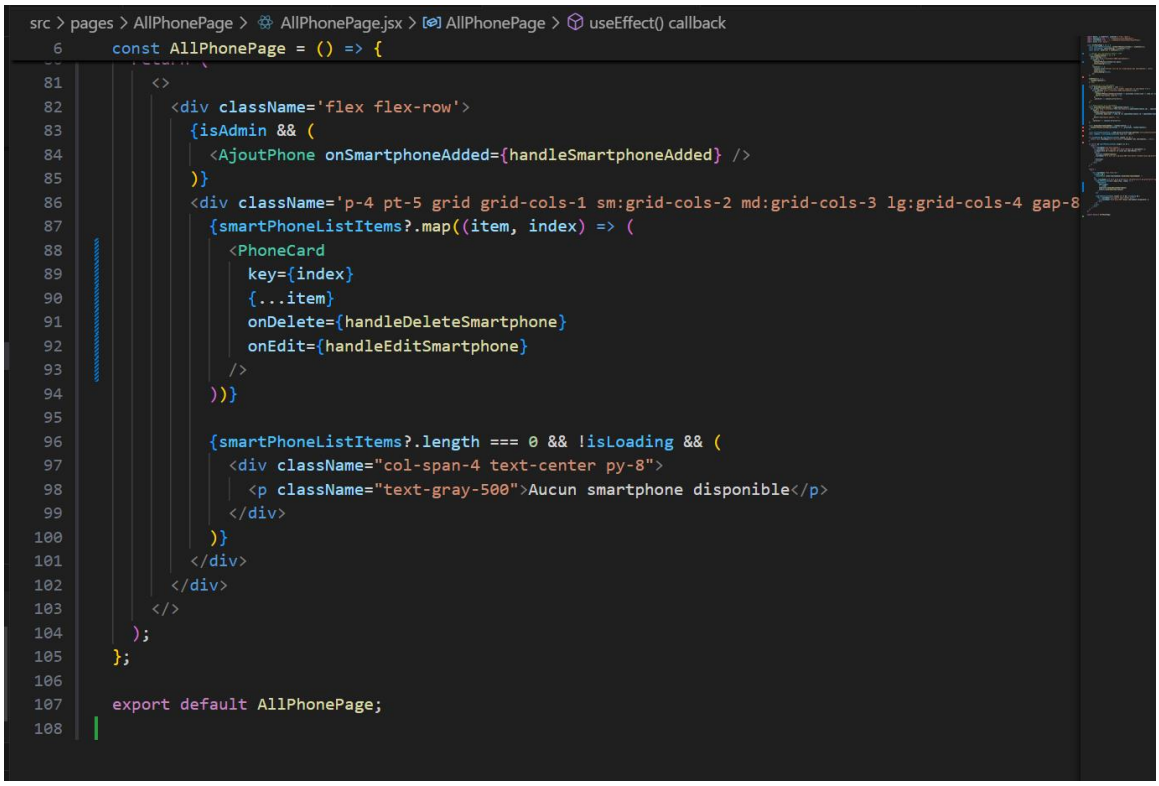
Capture d'écran :






Code associé :

• Fichier : AllPhonePage.jsx - Affichage sans actions Admin




## 3.8 Interface d'accueil (Utilisateur)

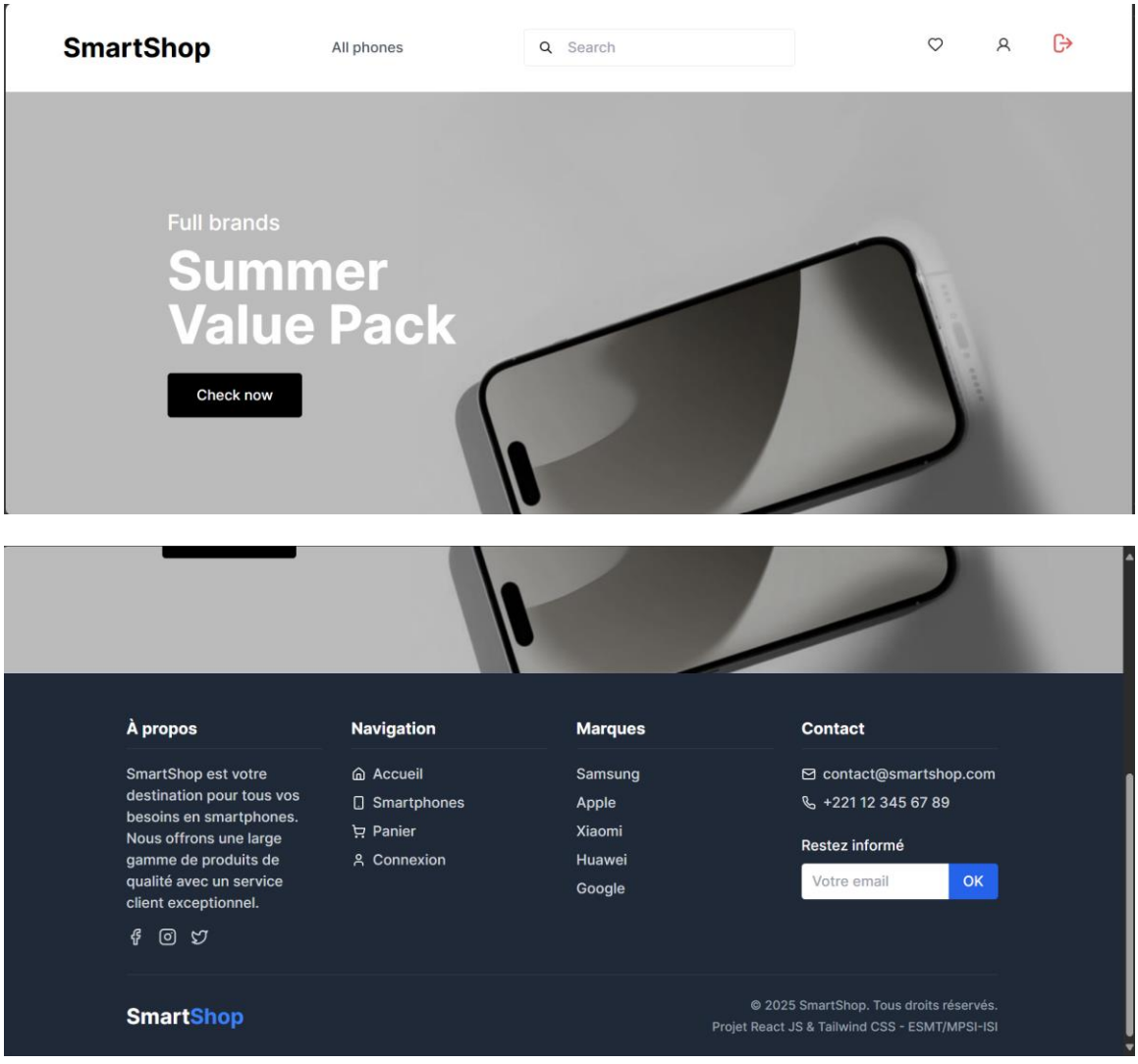
La page d'accueil propose une vue d'ensemble de l'application avec :

- Un **bannière visuelle** pour mettre en avant les offres et les nouveautés.

- Un **menu de navigation** permettant d'accéder rapidement aux différentes sections de l'application : liste des téléphones, recherche, connexion, etc.
- Une interface épurée et agréable, conçue avec TailwindCSS pour assurer une excellente expérience utilisateur.

Cette page donne la première impression de l'application et met en avant l'identité visuelle du projet.

Capture d'écran :




Code associé :

• Fichier : HeroSection.jsx

src > components > HeroSection > HeroSection.jsx > ...

```jsx
import React from 'react'
import PhoneHeroImgage from './../../assets/images/PhoneHeroImage2.jpg'
import { NavLink } from 'react-router-dom'

Tabnine | Edit | Explain
const HeroSection = () => {
  return (
    <div className='relative flex items-center bg-cover bg-center text-left w-full h-280px' style={{
      backgroundImage: `url(${PhoneHeroImgage})`,
      height: 'calc(100vh - 80px)' // Ajustez selon la hauteur de votre navbar
    }}>
      <div className='absolute inset-0 bg-black bg-opacity-20'></div>
      <div className='container mx-auto px-6 lg:px-16 z-10'>
        <div className='max-w-lg'>
          <h2 className='text-xl md:text-2xl text-white font-medium'>Full brands</h2>
          <h1 className='mt-2 text-4xl md:text-6xl text-white font-bold leading-tight'>
            Summer<br />Value Pack
          </h1>

          <button className='mt-6 py-3 px-8 border rounded border-black hover:bg-white hover:text-bl
            <NavLink to="/allPhones">Check now</NavLink>
          </button>
        </div>
      </div>
    </div>
  )
}

export default HeroSection
```

## 4. Conclusion

Ce projet a permis de mettre en œuvre les concepts de base du développement web avec ReactJS et la gestion de données via une API factice. L'application permet une gestion fluide des smartphones, intégrant les rôles utilisateur et administrateur. Les fonctionnalités CRUD sont complètement opérationnelles, et l'interface utilisateur est intuitive et ergonomique.