

# 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 :

SmartShop

All phones

Q

Search

Connexion

Entrez vos identifiants pour accéder à votre compte

Nom d'utilisateur

Admin

Mot de passe

.....

☐

Se souvenir de moi

Se connecter

Vous n'avez pas de compte? [S'inscrire](#)

Code associé :

- Fichier : Login.jsx

```
src > pages > LoginPage > Login.jsx > ...
1  import React, { useState } from 'react';
2  import { useNavigate } from 'react-router-dom';
3  import { Eye, EyeOff, User, Lock, LogIn } from 'lucide-react';
4
5  Tabnine | Edit | Explain
6  const Login = () => {
7    const [nom, setNom] = useState('');
8    const [motDePasse, setMotDePasse] = useState('');
9    const [erreur, setErreur] = useState('');
10   const [afficherMotDePasse, setAfficherMotDePasse] = useState(false);
11   const navigate = useNavigate();
12
13   // Fonction handleLogin originale, non modifiée
14   const handleLogin = async (e) => {
15     e.preventDefault();
16
17     try {
18       const response = await fetch(`http://localhost:3001/utilisateurs?nom=${nom}&motDePasse=${motDePasse}`);
19       const data = await response.json();
20
21       if (data.length > 0) {
22         const user = data[0];
23         // Enregistre l'utilisateur dans le localStorage
24         localStorage.setItem('utilisateurConnecte', JSON.stringify(user));
25         navigate('/app');
26       } else {
27         setErreur('Nom ou mot de passe incorrect.');
```

```

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   </div>
68 </div>
69
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     </div>
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     </div>

```

```

src > pages > LoginPage > Login.jsx > ...
5   const Login = () => {
101
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-transparent
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" />

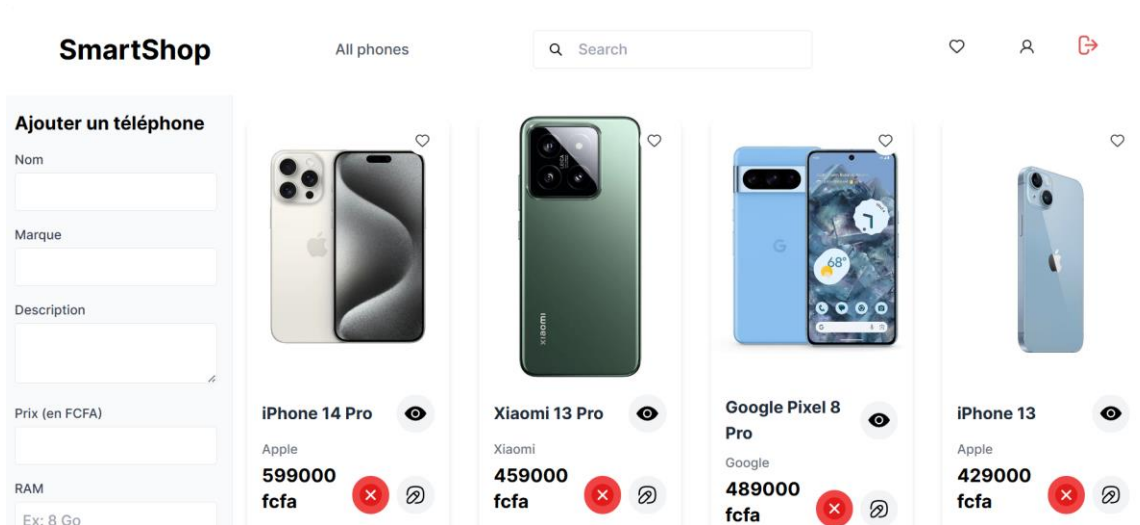
```

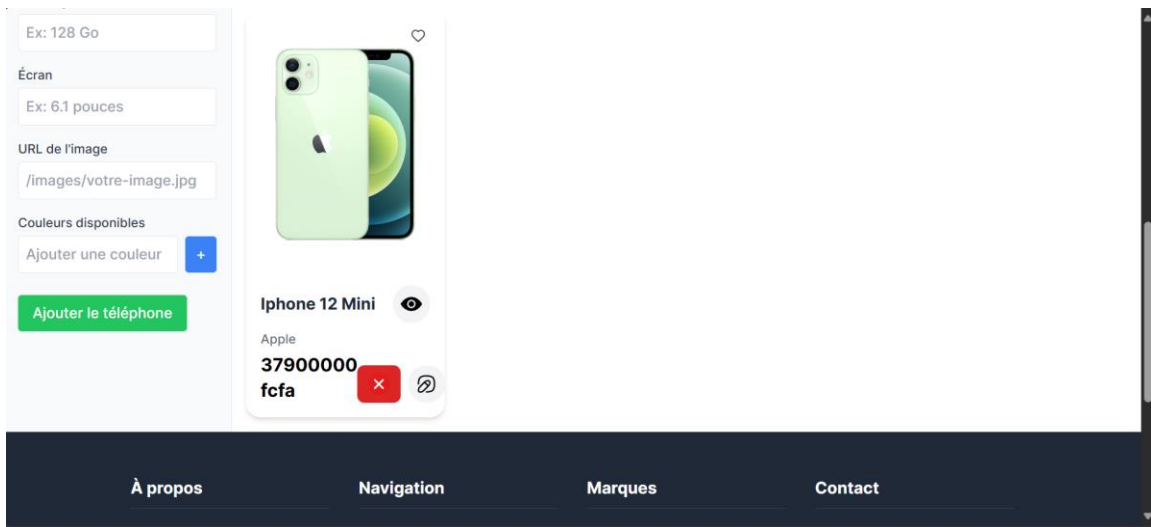
## 3.2 Interface de Liste des Smartphones(Admin)

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

```
src > pages > AllPhonePage > AllPhonePage.jsx > [⌕] AllPhonePage
1  import React, { useEffect, useState } from 'react';
2  import PhoneCard from '../../components/Card/PhoneCard';
3  import AjoutPhone from '../../components/AjoutPhone/AjoutPhone';
4  import axios from 'axios';
5
6  Tabnine | Edit | Explain
7  const AllPhonePage = () => {
8    const [smartPhoneListItems, setSmartPhoneListItems] = useState([]);
9    const [isLoading, setIsLoading] = useState(true);
10   const [error, setError] = useState(null);
11
12   // Charger les smartphones depuis l'API
13   const loadSmartphones = () => {
14     setIsLoading(true);
15     axios.get('http://localhost:3001/smartphones')
16       .then(res => {
17         setSmartPhoneListItems(res.data);
18         setIsLoading(false);
19       })
20       .catch(err => {
21         console.error("Erreur lors de la récupération des smartphones:", err);
22         setError(err);
23         setIsLoading(false);
24       });
25   };
26
27   useEffect(() => {
28     loadSmartphones();
29   }, []);
30
31   // Suppression d'un smartphone
32   const handleDeleteSmartphone = (id) => {
```

### 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 :

# SmartShop

## Ajouter un téléphone

Nom

Marque

Description

Prix (en FCFA)

RAM

Stockage

Écran

URL de l'image

Couleurs disponibles

Ajouter le téléphone

Code associé :

- Fichier : AjoutPhone.jsx

```
src > components > AjoutPhone > AjoutPhone.jsx > ...
1  import React, { useState } from 'react';
2  import axios from 'axios';
3
4  Tabnine | Edit | Explain
5  const AjoutPhone = ({ onSmartphoneAdded }) => {
6    // État pour les champs du formulaire
7    const [formData, setFormData] = useState({
8      name: '',
9      brand: '',
10     description: '',
11     price: '',
12     ram: '',
13     rom: '',
14     ecran: '',
15     imageUrl: ''
16   });
17
18   // État pour les couleurs
19   const [couleur, setCouleur] = useState('');
20   const [couleursDisponibles, setCouleursDisponibles] = useState([]);
21
22   // État pour le statut de soumission
23   const [isSubmitting, setIsSubmitting] = useState(false);
24   const [submitStatus, setSubmitStatus] = useState({ success: false, message: '' });
25
26   // Gestion du changement des champs du formulaire
27   const handleInputChange = (e) => {
28     const { name, value } = e.target;
29     setFormData({
30       ...formData,
31       [name]: value
32     });
33   };
34 }
```

```
src > components > AjoutPhone > AjoutPhone.jsx > ...
1  import React, { useState } from 'react';
2  import axios from 'axios';
3
4  Tabnine | Edit | Explain
5  const AjoutPhone = ({ onSmartphoneAdded }) => {
6    // État pour les champs du formulaire
7    const [formData, setFormData] = useState({
8      name: '',
9      brand: '',
10     description: '',
11     price: '',
12     ram: '',
13     rom: '',
14     ecran: '',
15     imageUrl: ''
16   });
17
18   // État pour les couleurs
19   const [couleur, setCouleur] = useState('');
20   const [couleursDisponibles, setCouleursDisponibles] = useState([]);
21
22   // État pour le statut de soumission
23   const [isSubmitting, setIsSubmitting] = useState(false);
24   const [submitStatus, setSubmitStatus] = useState({ success: false, message: '' });
25
26   // Gestion du changement des champs du formulaire
27   const handleInputChange = (e) => {
28     const { name, value } = e.target;
29     setFormData({
30       ...formData,
31       [name]: value
32     });
33   };
34 }
```

```
src > components > AjoutPhone > AjoutPhone.jsx > AjoutPhone > ajouterCouleur
4   const AjoutPhone = ({ onSmartphoneAdded }) => {
53  const handleSubmit = (e) => {
54    // Envoi à l'API
55    axios.post('http://localhost:3001/smartphones', newSmartphone)
56      .then(response => {
57        setSubmitStatus({
58          success: true,
59          message: 'Smartphone ajouté avec succès!'
60        });
61
62        // Réinitialisation du formulaire
63        setFormData({
64          name: '',
65          brand: '',
66          description: '',
67          price: '',
68          ram: '',
69          rom: '',
70          ecran: '',
71          imageUrl: ''
72        });
73        setCouleursDisponibles([]);
74
75        // Notification au composant parent
76        if (onSmartphoneAdded) {
77          onSmartphoneAdded(response.data);
78        }
79      })
80      .catch(error => {
81        console.error('Erreur lors de l\'ajout du smartphone:', error);
82        setSubmitStatus({
83          success: false,
84          message: 'Erreur lors de l\'ajout du smartphone.'
85        });
86      });
87    finally(() => {
88      setIsSubmitting(false);
89    });
90  };
91
92  return (
93    <div className='p-4 border-r bg-gray-50 w-1/4'>
94      <h2 className='text-xl font-bold mb-4'>Ajouter un téléphone</h2>
95      <div className='mb-4 p-2 rounded ${submitStatus.success ? 'bg-green-100 text-green-800' : ''}>
96        {submitStatus.message}
97      </div>
98      <form className='space-y-4' onSubmit={handleSubmit}>
99        <div>
100          <label className='block text-sm font-medium text-gray-700 mb-1'>Nom</label>
101          <input
102            type='text'
103            name='name'
104            value={formData.name}
105            onChange={handleInputChange}
106            className='w-full p-2 border rounded'
107            required
108          />
109        </div>
110      </form>
111    </div>
112  );
113}
```

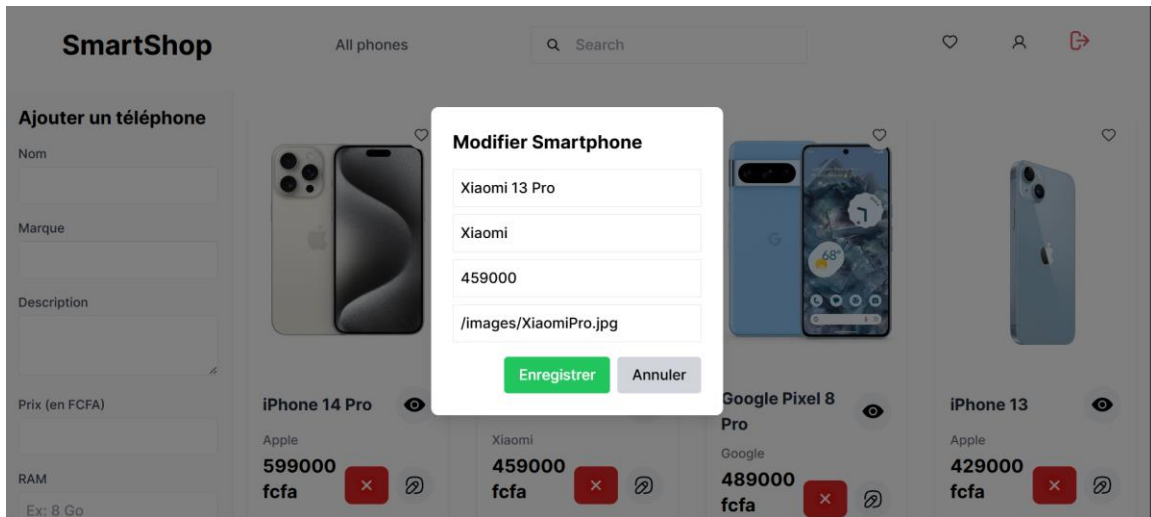
```
src > components > AjoutPhone > AjoutPhone.jsx > AjoutPhone > ajouterCouleur
4   const AjoutPhone = ({ onSmartphoneAdded }) => {
53  const handleSubmit = (e) => {
92    .catch(error => {
93    });
94  });
95  .finally(() => {
96    setIsSubmitting(false);
97  });
98  };
99
100  return (
101    <div className='p-4 border-r bg-gray-50 w-1/4'>
102      <h2 className='text-xl font-bold mb-4'>Ajouter un téléphone</h2>
103      <div className='mb-4 p-2 rounded ${submitStatus.success ? 'bg-green-100 text-green-800' : ''}>
104        {submitStatus.message}
105      </div>
106      <form className='space-y-4' onSubmit={handleSubmit}>
107        <div>
108          <label className='block text-sm font-medium text-gray-700 mb-1'>Nom</label>
109          <input
110            type='text'
111            name='name'
112            value={formData.name}
113            onChange={handleInputChange}
114            className='w-full p-2 border rounded'
115            required
116          />
117        </div>
118      </form>
119    </div>
120  );
121}
```



### 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 }) => {
70
71   /* Modal de Modification */
72   {showEdit && (
73     <div className='fixed inset-0 bg-black bg-opacity-50 flex justify-center items-center z-50'>
74       <div className='bg-white p-6 rounded-lg shadow-lg w-80'>
75         <h2 className='text-xl font-bold mb-4'>Modifier Smartphone</h2>
76         <input
77           className='border p-2 mb-2 w-full'
78           value={editData.name}
79           onChange={(e) => setEditData({ ...editData, name: e.target.value })}
80           placeholder='Nom'
81         />
82         <input
83           className='border p-2 mb-2 w-full'
84           value={editData.brand}
85           onChange={(e) => setEditData({ ...editData, brand: e.target.value })}
86           placeholder='Marque'
87         />
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>
102    </div>
103  )}
104 }
```

```

src > components > Card > PhoneCard.jsx > PhoneCard
8   const PhoneCard = ({ id, name, brand, price, imageUrl, onDelete, onEdit }) => {
88
89     <input
90       className='border p-2 mb-2 w-full'
91       value={editData.price}
92       onChange={(e) => setEditData({ ...editData, price: e.target.value })}
93       placeholder='Prix'
94       type='number'
95     />
96
97     <input
98       className='border p-2 mb-4 w-full'
99       value={editData.imageUrl}
100      onChange={(e) => setEditData({ ...editData, imageUrl: e.target.value })}
101      placeholder='URL Image'
102    />
103
104    <div className='flex justify-end space-x-2'>
105      <button
106        onClick={handleEditSubmit}
107        className='bg-green-500 text-white px-4 py-2 rounded hover:bg-green-600'
108      >
109        Enregistrer
110      </button>
111      <button
112        onClick={() => setShowEdit(false)}
113        className='bg-gray-300 px-4 py-2 rounded hover:bg-gray-400'
114      >
115        Annuler
116      </button>
117    </div>
118  </div>
119  )}

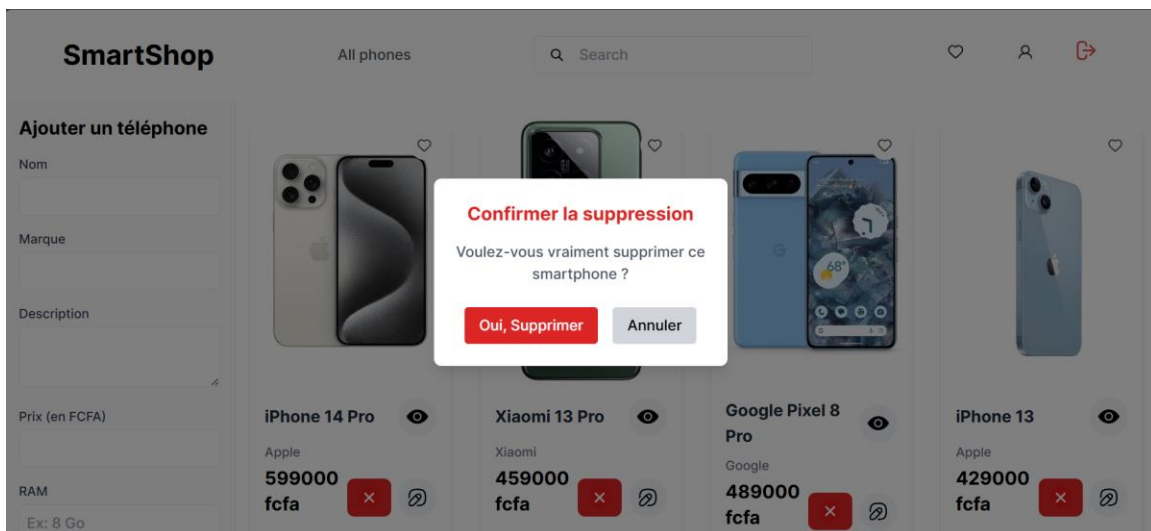
```

### 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

```

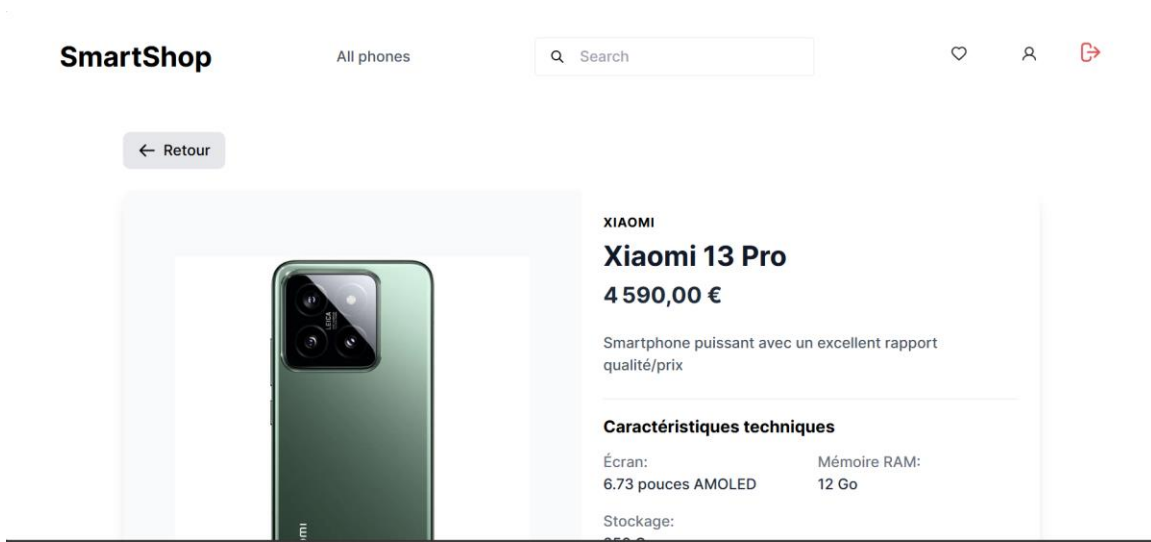
src > components > Card > PhoneCard.jsx > PhoneCard
8   const PhoneCard = ({ id, name, brand, price, imageUrl, onDelete, onEdit }) => {
118
119     /* Modal de Confirmation Suppression */
120     {showDeleteConfirm && (
121       <div className='fixed inset-0 bg-black bg-opacity-50 flex justify-center 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

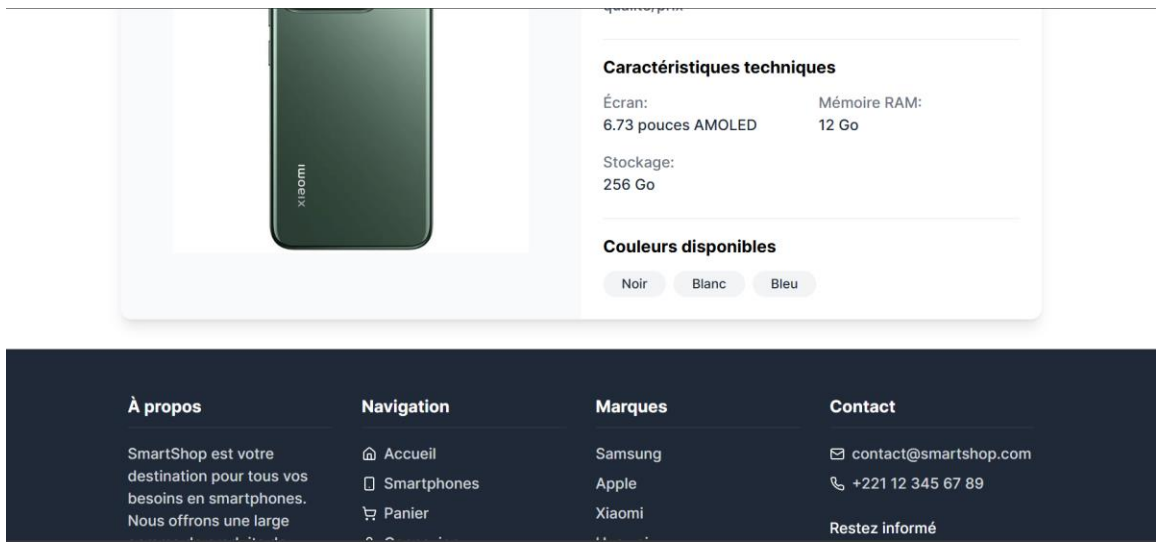
```

### 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 > ...
1 import React, { useEffect, useState } from 'react';
2 import { useParams, useNavigate } from 'react-router-dom';
3 import axios from 'axios';
4
5 Tabnine | Edit | Explain
6 const PhoneDetailPage = () => {
7   const { phoneId } = useParams();
8   const navigate = useNavigate();
9   const [smartphone, setSmartphone] = useState(null);
10  const [isLoading, setIsLoading] = useState(true);
11  const [error, setError] = useState(null);
12
13  useEffect(() => {
14    // Récupération des données avec axios
15    axios.get(`http://localhost:3001/smartphones/${phoneId}`)
16      .then(res => {
17        setSmartphone(res.data);
18        setIsLoading(false);
19      })
20      .catch(err => {
21        console.error("Erreur lors de la récupération du smartphone:", err);
22        setError(err);
23        setIsLoading(false);
24      });
25  }, [phoneId]);
26
27  // Si les données sont en cours de chargement
28  if (isLoading) {
29    return <div className="p-8 text-center">Chargement en cours...</div>;
30  }
31
32  // Si une erreur s'est produite
```

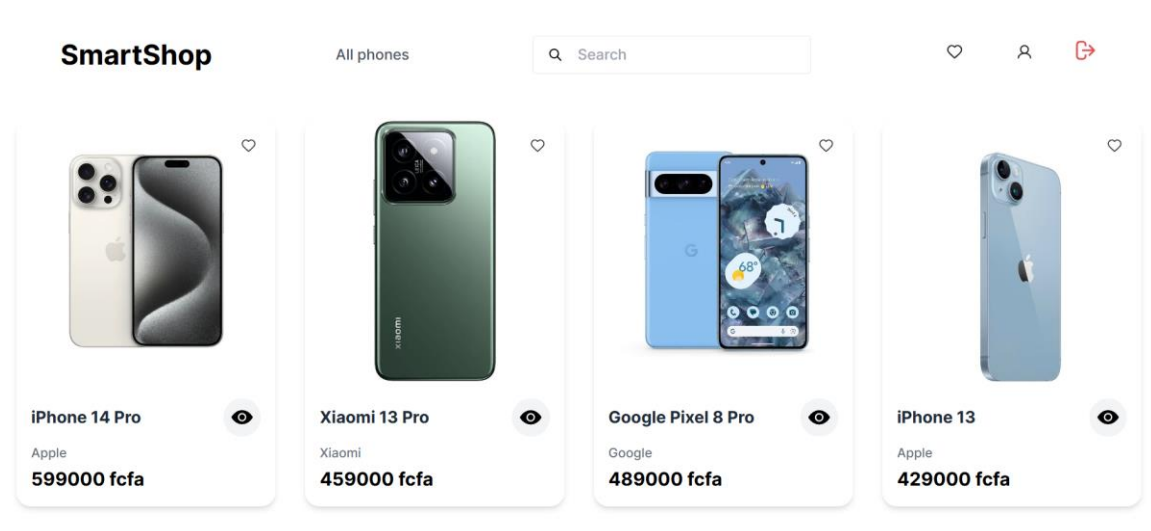
```
src > pages > PhoneDetails > PhoneDetailPage.jsx > ...
5   const PhoneDetailPage = () => {
29   }
30
31   // Si une erreur s'est produite
32   if (error) {
33     return (
34       <div className="flex flex-col items-center justify-center min-h-screen px-4 py-8">
35         <h2 className="text-2xl 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}.</p>
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>
60     );
61   }
62 }
```

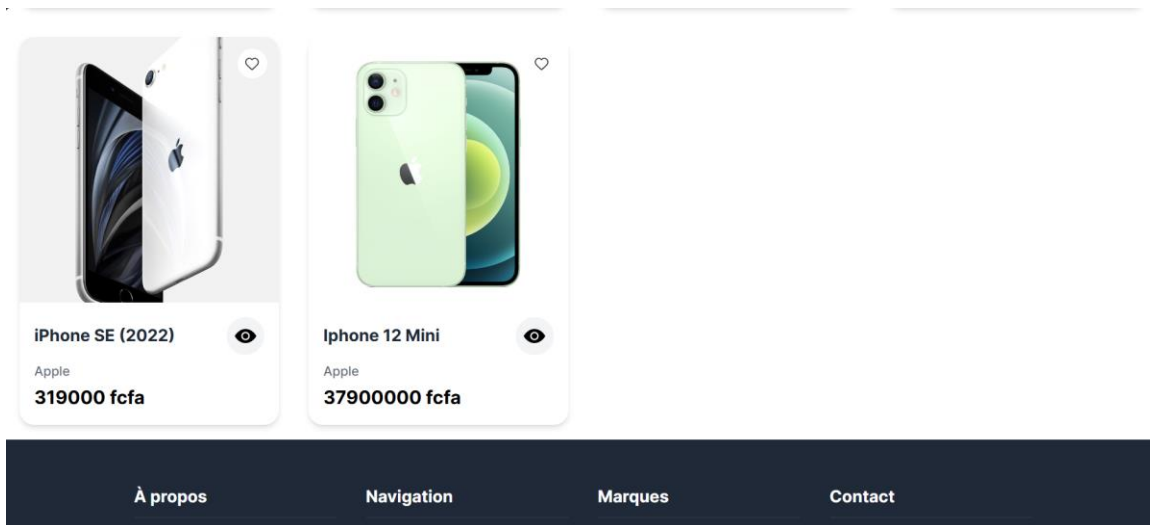
### 3.7 Interface de Consultation des Smartphones (Utilisateur)

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

```
src > pages > AllPhonePage > AllPhonePage.jsx > AllPhonePage > useEffect() callback
6   const AllPhonePage = () => {
81     <>
82       <div className='flex flex-row'>
83         {isAdmin && (
84           <AjoutPhone onSmartphoneAdded={handleSmartphoneAdded} />
85         )}
86       <div className='p-4 pt-5 grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-8'>
87         {smartPhoneListItems?.map((item, index) => (
88           <PhoneCard
89             key={index}
90             {...item}
91             onDelete={handleDeleteSmartphone}
92             onEdit={handleEditSmartphone}
93           />
94         ))}
95       <div className='col-span-4 text-center py-8'>
96         {smartPhoneListItems?.length === 0 && !isLoading && (
97           <p className='text-gray-500'>Aucun smartphone disponible</p>
98         )}
99       </div>
100     </div>
101   </div>
102 </div>
103 </>
104 );
105 };
106
107 export default AllPhonePage;
108
```

### 3.8 Interface d'accueil (Utilisateur)

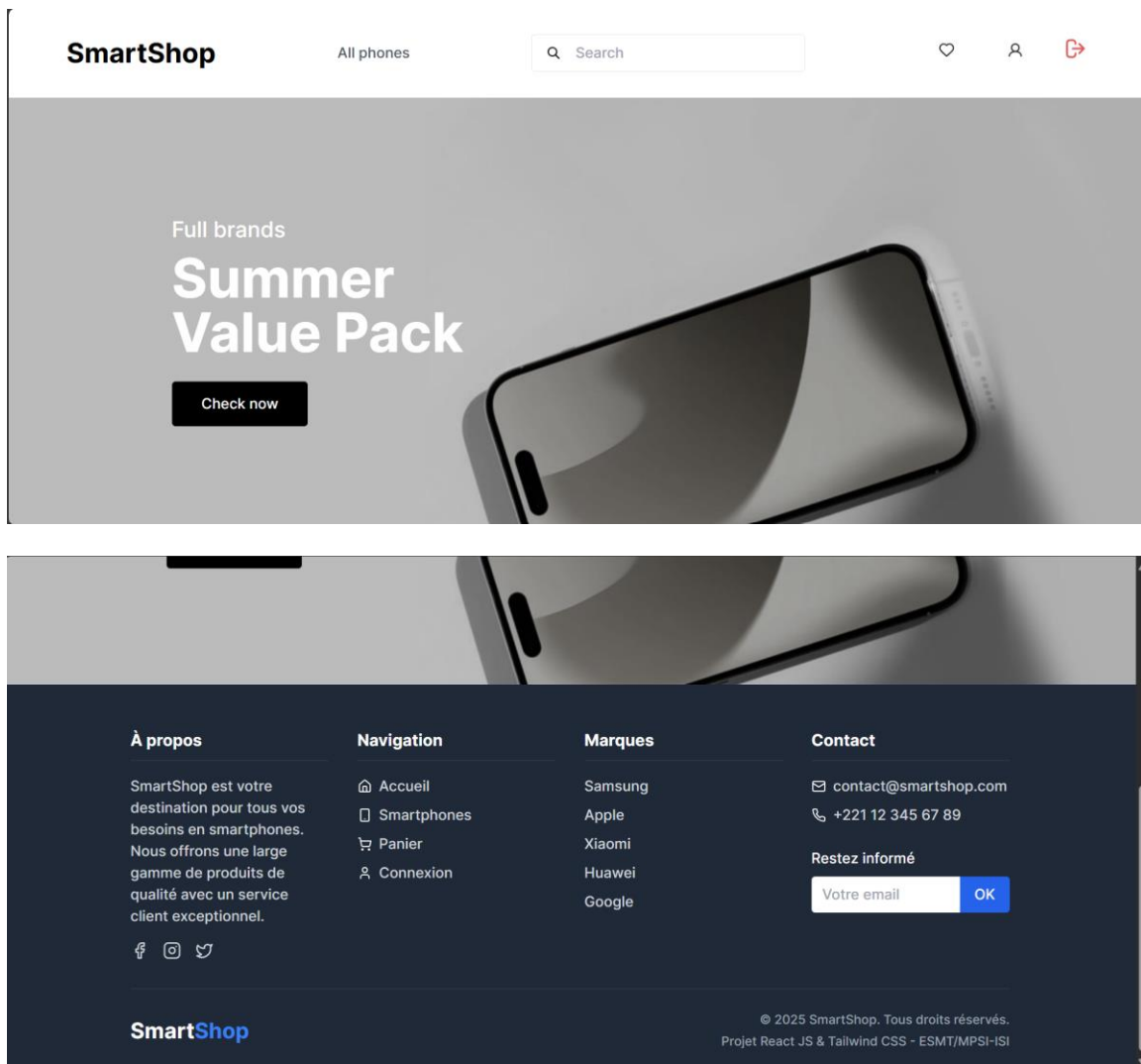
La page d'accueil propose une vue d'ensemble de l'application avec :

- 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 > ...
1 import React from 'react'
2 import PhoneHeroImage from './../../assets/images/PhoneHeroImage2.jpg'
3 import { NavLink } from 'react-router-dom'
4
5 Tabnine | Edit | Explain
6 const HeroSection = () => {
7   return (
8     <div className='relative flex items-center bg-cover bg-center text-left w-full h-280px' style={{
9       backgroundImage: `url(${PhoneHeroImage})`,
10       height: 'calc(100vh - 80px)' // Ajustez selon la hauteur de votre navbar
11     }}>
12       <div className='absolute inset-0 bg-black bg-opacity-20'></div>
13       <div className='container mx-auto px-6 lg:px-16 z-10'>
14         <div className='max-w-lg'>
15           <h2 className='text-xl md:text-2xl text-white font-medium'>Full brands</h2>
16           <h1 className='mt-2 text-4xl md:text-6xl text-white font-bold leading-tight'>
17             Summer<br />Value Pack
18           </h1>
19
20           <button className='mt-6 py-3 px-8 border rounded border-black hover:bg-white hover:text-bl
21             <NavLink to="/allPhones">Check now</NavLink>
22           </button>
23         </div>
24       </div>
25     </div>
26   )
27 }
28 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.