Guide de démarrage rapide - React/Vite

📋 Prérequis

Avant de commencer, assurez-vous d'avoir installé :

- **Node.js** (version 18 ou supérieure) : https://nodejs.org
- **npm** (inclus avec Node.js)
- **Git** (optionnel mais recommandé)
- **VS Code** ou un autre éditeur de code

Vérifiez vos installations :

```
node --version    # Doit afficher v18.x.x ou supérieur
npm --version     # Doit afficher 9.x.x ou supérieur
```

🛠️ Installation du projet React

1. Naviguer vers le dossier React

```
cd
/Users/macbookm1pro/Document/Diplome_final/Code/plateforme_porsche/React
```

2. Installer les dépendances existantes

```
npm install
```

3. Installer les dépendances supplémentaires nécessaires

```
# React Router pour la navigation
npm install react-router-dom

# Axios pour les requêtes HTTP
npm install axios

# (Optionnel) Bibliothèques utiles
npm install react-icons          # Icônes
npm install date-fns            # Manipulation dates
npm install @stripe/stripe-js @stripe/react-stripe-js  # Stripe pour
paiement
```

📂 Créer la structure de dossiers

Méthode automatique (Bash/Zsh - MacOS/Linux)

Copiez et collez ce script dans votre terminal :

```
cd src

# Créer les dossiers principaux
mkdir -p config
mkdir -p services/api
mkdir -p hooks
mkdir -p contextes
mkdir -p utils
mkdir -p styles

# Créer les dossiers de composants
mkdir -p
composants/communs/{Bouton,Chargement,Modale,Notification,Formulaire,Pag
ination,Tableau}
mkdir -p composants/layout/{EnTete,Navigation,PiedDePage,MiseEnPage}
mkdir -p
composants/voiture/{CarteVoiture,ListeVoitures,GaleriePhotos,FichesTechn
iques,FiltreVoiture}
mkdir -p
composants/configurateur/{SelecteurCouleur,SelecteurJantes,SelecteurSieg
es,SelecteurPackage,RecapitulatifConfig}
mkdir -p composants/accessoire/{CarteAccessoire,ListeAccessoires}
mkdir -p composants/panier/{ArticlePanier,ResumePanier,IconePanier}
mkdir -p
composants/commande/{FormulaireCommande,RecapitulatifCommande,StatutComm
ande}
mkdir -p composants/paiement/{FormulairePaiement,ConfirmationPaiement}
mkdir -p
composants/admin/{TableauStatistiques,FormulaireVoiture,GestionPhotos}
mkdir -p composants/protection

# Créer les dossiers de pages
mkdir -p pages/Accueil
mkdir -p pages/Authentification
mkdir -p pages/Catalogue
mkdir -p pages/Configurateur
mkdir -p pages/Accessoires
mkdir -p pages/Panier
mkdir -p pages/Commande
mkdir -p pages/Profil
mkdir -p pages/Administration
mkdir -p pages/Erreur
```

```
echo "✅ Structure créée avec succès !"
```

Méthode manuelle (si le script ne fonctionne pas)

Créez les dossiers un par un dans VS Code :

1. Clic droit sur `src` → Nouveau dossier
2. Créez d'abord les dossiers principaux : config, services, hooks, contextes, composants, pages, utils, styles

---

📝 Créer les fichiers de base

1. Configuration API

Copiez le contenu depuis `EXEMPLES_CODE/config-api.js` vers `src/config/api.js`

2. Service d'authentification

Copiez `EXEMPLES_CODE/authService.js` vers `src/services/api/authService.js`

3. Contexte d'authentification

Copiez `EXEMPLES_CODE/AuthContexte.jsx` vers `src/contextes/AuthContexte.jsx`

4. Hook useAuth

Copiez `EXEMPLES_CODE/useAuth.js` vers `src/hooks/useAuth.js`

5. Routes protégées

- Copiez `EXEMPLES_CODE/RoutePrivee.jsx` vers `src/composants/protection/RoutePrivee.jsx`
- Copiez `EXEMPLES_CODE/RouteAdmin.jsx` vers `src/composants/protection/RouteAdmin.jsx`

6. Composant Bouton

- Copiez `EXEMPLES_CODE/Bouton.jsx` vers `src/composants/communs/Bouton/Bouton.jsx`
- Copiez `EXEMPLES_CODE/Bouton.css` vers `src/composants/communs/Bouton/Bouton.css`

⚙️ Configuration des variables d'environnement

1. Créer le fichier `.env.development`

À la racine du dossier React, créez `.env.development` :

```
# URL du backend
VITE_API_URL=http://localhost:3000

# Clé publique Stripe (pour paiements)
VITE_STRIPE_PUBLIC_KEY=votre_cle_publique_stripe
```

2. Créer le fichier `.env.production`

```
# URL du backend en production
VITE_API_URL=https://votre-api.com

# Clé publique Stripe production
VITE_STRIPE_PUBLIC_KEY=votre_cle_publique_stripe_prod
```

⚠ **Important** : Les variables DOIVENT commencer par `VITE_` pour être accessibles dans Vite.

🎨 Créer les styles globaux

1. Fichier `src/styles/variables.css`

```
:root {
  /* Couleurs Porsche */
  --couleur-principale: #000000;
  --couleur-secondaire: #d5001c;
  --couleur-accent: #c0a062;

  --couleur-texte: #333333;
  --couleur-texte-clair: #666666;
  --couleur-fond: #ffffff;
  --couleur-fond-gris: #f5f5f5;

  /* Espacements */
  --espace-xs: 0.25rem;
  --espace-sm: 0.5rem;
  --espace-md: 1rem;
  --espace-lg: 1.5rem;
  --espace-xl: 2rem;
  --espace-xxl: 3rem;

  /* Typographie */
  --font-principale: -apple-system, BlinkMacSystemFont, "Segoe UI",
Arial,
    sans-serif;
  --font-taille-xs: 0.75rem;
  --font-taille-sm: 0.875rem;
  --font-taille-base: 1rem;
```

```
  --font-taille-lg: 1.125rem;
  --font-taille-xl: 1.25rem;
  --font-taille-xxl: 1.5rem;
  --font-taille-titre: 2rem;

  /* Ombres */
  --ombre-sm: 0 1px 2px rgba(0, 0, 0, 0.05);
  --ombre-md: 0 4px 6px rgba(0, 0, 0, 0.1);
  --ombre-lg: 0 10px 15px rgba(0, 0, 0, 0.1);

  /* Transitions */
  --transition-rapide: 150ms ease-in-out;
  --transition-normale: 300ms ease-in-out;
  --transition-lente: 500ms ease-in-out;

  /* Bordures */
  --rayon-bordure-sm: 4px;
  --rayon-bordure-md: 8px;
  --rayon-bordure-lg: 12px;
}
```

2. Importer dans `src/index.css`

En haut du fichier :

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

/* Votre CSS existant... */
```

🔧 Configuration du fichier `vite.config.js`

Modifiez votre `vite.config.js` pour ajouter les alias :

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";
import path from "path";

export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: {
      "@": path.resolve(__dirname, "./src"),
      "@config": path.resolve(__dirname, "./src/config"),
      "@services": path.resolve(__dirname, "./src/services"),
      "@hooks": path.resolve(__dirname, "./src/hooks"),
      "@contextes": path.resolve(__dirname, "./src/contextes"),
      "@composants": path.resolve(__dirname, "./src/composants"),
```

```
      "@pages": path.resolve(__dirname, "./src/pages"),
      "@utils": path.resolve(__dirname, "./src/utils"),
      "@styles": path.resolve(__dirname, "./src/styles"),
    },
  },
  server: {
    port: 5173,
    open: true, // Ouvre automatiquement le navigateur
  },
});
```

Cela vous permet d'utiliser :

```
import api from "@/config/api";
import { useAuth } from "@/hooks/useAuth";
```

Au lieu de :

```
import api from "../../../config/api";
import { useAuth } from "../../../hooks/useAuth";
```

🚦 Démarrer le projet

1. Démarrer le backend (Terminal 1)

```
cd Node
npm run dev
```

Le backend démarre sur `http://localhost:3000`

2. Démarrer le frontend (Terminal 2)

```
cd React
npm run dev
```

Le frontend démarre sur `http://localhost:5173`

3. Ouvrir dans le navigateur

Accédez à : http://localhost:5173

📁 Structure finale attendue

```
React/src/
├── config/
│   └── api.js                    ✅ Créé
├── services/
│   └── api/
│       └── authService.js        ✅ Créé
├── hooks/
│   └── useAuth.js                ✅ Créé
├── contextes/
│   └── AuthContexte.jsx          ✅ Créé
├── composants/
│   ├── communs/
│   │   └── Bouton/
│   │       ├── Bouton.jsx        ✅ Créé
│   │       └── Bouton.css        ✅ Créé
│   ├── protection/
│   │   ├── RoutePrivee.jsx       ✅ Créé
│   │   └── RouteAdmin.jsx        ✅ Créé
│   └── [autres composants...]    ⏳ À créer
├── pages/
│   └── [pages...]                ⏳ À créer
├── utils/
│   └── [utilitaires...]          ⏳ À créer
├── styles/
│   └── variables.css             ✅ Créé
├── App.jsx
├── main.jsx
└── index.css
```

🎯 Prochaines étapes

Étape 1 : Modifier `App.jsx`

```jsx
import { BrowserRouter } from "react-router-dom";
import { AuthProvider } from "./contextes/AuthContexte";
import "./App.css";

function App() {
  return (
    <BrowserRouter>
      <AuthProvider>
        <div className="App">
          <h1>Plateforme Porsche</h1>
          <p>Configuration en cours...</p>
        </div>
      </AuthProvider>
```

```
    </BrowserRouter>
  );
}

export default App;
```

Étape 2 : Créer une page de connexion simple

Créez `src/pages/Authentification/Connexion.jsx` :

```
import { useState } from "react";
import { useAuth } from "@/hooks/useAuth";
import Bouton from "@/composants/communs/Bouton/Bouton";
import "./Connexion.css";

export default function Connexion() {
  const { connexion } = useAuth();
  const [email, setEmail] = useState("");
  const [motDePasse, setMotDePasse] = useState("");
  const [erreur, setErreur] = useState("");
  const [chargement, setChargement] = useState(false);

  const gererSoumission = async (e) => {
    e.preventDefault();
    setErreur("");
    setChargement(true);

    const resultat = await connexion(email, motDePasse);

    setChargement(false);

    if (resultat.success) {
      alert("Connexion réussie !");
    } else {
      setErreur(resultat.erreur);
    }
  };

  return (
    <div className="connexion">
      <form className="connexion__formulaire" onSubmit=
{gererSoumission}>
        <h1>Connexion</h1>

        {erreur && <div className="erreur">{erreur}</div>}

        <input
          type="email"
          placeholder="Email"
          value={email}
```

```
            onChange={(e) => setEmail(e.target.value)}
            required
          />

          <input
            type="password"
            placeholder="Mot de passe"
            value={motDePasse}
            onChange={(e) => setMotDePasse(e.target.value)}
            required
          />

          <Bouton
            type="submit"
            texte={chargement ? "Connexion..." : "Se connecter"}
            desactive={chargement}
            pleineLargeur
          />
        </form>
      </div>
    );
}
```

Créez `src/pages/Authentification/Connexion.css` :

```
.connexion {
  min-height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  background-color: var(--couleur-fond-gris);
  padding: var(--espace-xl);
}

.connexion__formulaire {
  background: var(--couleur-fond);
  padding: var(--espace-xxl);
  border-radius: var(--rayon-bordure-lg);
  box-shadow: var(--ombre-lg);
  width: 100%;
  max-width: 400px;
  display: flex;
  flex-direction: column;
  gap: var(--espace-lg);
}

.connexion__formulaire h1 {
  text-align: center;
  margin: 0 0 var(--espace-lg);
}
```

```css
.connexion__formulaire input {
  padding: var(--espace-md);
  border: 1px solid #ddd;
  border-radius: var(--rayon-bordure-sm);
  font-size: var(--font-taille-base);
}

.erreur {
  background-color: #fee;
  color: #c00;
  padding: var(--espace-md);
  border-radius: var(--rayon-bordure-sm);
  border: 1px solid #fcc;
}
```

Étape 3 : Ajouter la route dans `App.jsx`

```jsx
import { BrowserRouter, Routes, Route } from "react-router-dom";
import { AuthProvider } from "./contextes/AuthContexte";
import Connexion from "./pages/Authentification/Connexion";
import "./App.css";

function App() {
  return (
    <BrowserRouter>
      <AuthProvider>
        <Routes>
          <Route
            path="/"
            element={
              <div>
                <h1>Accueil</h1>
              </div>
            }
          />
          <Route path="/connexion" element={<Connexion />} />
        </Routes>
      </AuthProvider>
    </BrowserRouter>
  );
}

export default App;
```

Étape 4 : Tester

1. Démarrez le backend et le frontend
2. Accédez à http://localhost:5173/connexion
3. Testez la connexion avec un compte existant

🐛 Résolution de problèmes

Erreur : `Cannot find module '@/...'`

Solution : Assurez-vous d'avoir configuré les alias dans `vite.config.js` et redémarré le serveur Vite.

Erreur CORS

Solution : Vérifiez que le backend autorise l'origine `http://localhost:5173` dans `Node/server.js`.

Erreur : `useAuth must be used within AuthProvider`

Solution : Assurez-vous que votre composant est enveloppé dans `<AuthProvider>` dans `App.jsx`.

Le backend ne démarre pas

Solution :

```
cd Node
npm install
# Vérifiez le fichier .env
npm run dev
```

📚 Ressources utiles

Documentation officielle

- **React** : https://react.dev
- **Vite** : https://vitejs.dev
- **React Router** : https://reactrouter.com
- **Axios** : https://axios-http.com

Tutoriels recommandés

- React pour débutants : https://react.dev/learn
- JavaScript moderne (ES6+) : https://javascript.info
- CSS Flexbox : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

✅ Checklist de démarrage

- ☐ Node.js et npm installés
- ☐ Dépendances installées (`npm install`)
- ☐ Structure de dossiers créée
- ☐ Fichiers d'exemple copiés

- ☐ Variables d'environnement configurées (`.env.development`)
- ☐ `vite.config.js` configuré avec les alias
- ☐ Styles globaux créés (`variables.css`)
- ☐ Backend démarré (`Node/` sur port 3000)
- ☐ Frontend démarré (`React/` sur port 5173)
- ☐ Page de connexion testée

🎓 Conseils pour bien démarrer

1. **Commencez petit** : Ne créez pas tous les composants d'un coup
2. **Testez souvent** : Après chaque nouveau composant, testez dans le navigateur
3. **Console.log()** : Utilisez-le pour déboguer
4. **React DevTools** : Installez l'extension Chrome/Firefox
5. **Git** : Faites des commits réguliers
6. **Documentation** : Consultez la documentation officielle en cas de doute

---

Bon développement ! 🚀

Si vous avez des questions, consultez les fichiers :

- `ARCHITECTURE_REACT.md` - Architecture complète
- `STRUCTURE_EXEMPLE.md` - Exemples détaillés
- `EXEMPLES_CODE/` - Fichiers d'exemple prêts à l'emploi