

DOCUMENTATION DU CODE POUR NOTRE PROJET WEB

**Groupe N\*8**

**Membres du Groupe**

**•** MBOCK Eliza Lolita (chef de graoupe)

• NTSAM Alain

• NASSER Ismael

• GUEMTUE Étienne Varnel

• MAVOUNGOU Smith

• ESSOMBA Aloys

**TABLE DE MATIERE**

[Structure de la documentation 2](#_Toc180738771)

[I. Introduction 2](#_Toc180738772)

[II. Mode d’installation 3](#_Toc180738773)

[III. Fichier Principaux 3](#_Toc180738774)

[IV. Configuration de vite 3](#_Toc180738775)

[V. Scripts 3](#_Toc180738776)

[VI. Dépendances 3](#_Toc180738777)

[VII. Composants 4](#_Toc180738778)

[**1.** **AccueilFinal** 4](#_Toc180738779)

[**2.** **Accueil** 5](#_Toc180738780)

[**3.** **Admin** 7](#_Toc180738781)

[**4.** **Contact** 11](#_Toc180738782)

[**5.** **Login** 11](#_Toc180738783)

[**6.** **Sign** 12](#_Toc180738784)

[**7.** **Employe** 12](#_Toc180738785)

[**8.** **Evènement** 14](#_Toc180738786)

[**9.** **Gérant** 15](#_Toc180738787)

[**10.** **Historique** 17](#_Toc180738788)

[**11.** **Menu** 18](#_Toc180738789)

[**12.** **Panier Etudiant** 19](#_Toc180738790)

[**13.** **Produit** 21](#_Toc180738791)

[**14.** **Parrainage** 21](#_Toc180738792)

[**15.** **Main** 22](#_Toc180738793)

[VIII. Linting 24](#_Toc180738794)

[IX. Conclusion 24](#_Toc180738795)

# Structure de la documentation

# Introduction

Cette documentation vise à fournir une vue d'ensemble complète du code du site web développé en React. Elle est organisée par dossiers, chaque dossier contenant des fichiers spécifiques qui jouent un rôle crucial dans le fonctionnement de l'application. L'objectif est de faciliter la compréhension du code, de ses composants et de son architecture pour les futurs développeurs et pour vous-même lors de la maintenance ou des mises à jour.

# Mode d’installation

Pour exécuter le code, nous devons aller en ligne de commande du terminal et taper : « npm run dev », cela nous donnera le local sur lequel nous verrons notre page soit : http://localhost:5174/.

# Fichier Principaux

* **vite.config.js** : Configuration de Vite pour le projet React, incluant le plugin React pour le support du JSX et du Fast Refresh.
* **README.md** : Fournit une introduction au projet, expliquant qu'il s'agit d'un modèle minimal pour utiliser React avec Vite, incluant des liens vers les plugins officiels.
* **package.json** : Contient les informations sur le projet, les scripts de démarrage, ainsi que les dépendances nécessaires pour le projet, comme React, React Router, et Vite.
* **package-lock.json** : Gère les versions précises des dépendances installées pour assurer la cohérence entre les installations.
* **index.html** : Le fichier HTML principal qui charge l'application React, incluant un élément div avec l'ID root, où l'application sera rendue.

# Configuration de vite

Vite est un bundler moderne qui optimise le développement et le déploiement d'applications web. La configuration par défaut est suffisante pour commencer, mais peut être personnalisée selon les besoins.

# Scripts

* **dev :** Lance le serveur de développement avec Vite.
* **build :** Compile le projet pour la production.
* **lint :** Exécute ESLint pour vérifier le code.
* **preview** : Permet de prévisualiser la version construite de l'application.

# Dépendances

* **React** : La bibliothèque principale pour construire l'interface utilisateur.
* **React Router** : Pour gérer le routage dans l'application.
* **Bootstrap :** Pour le style et la mise en page.
* **ESLint** : Pour le linting du code JavaScript.

# Composants

## **AccueilFinal**

* **Documentation du Composant ContactF**

Le composant ContactF est conçu pour permettre aux utilisateurs de soumettre des commentaires. Il commence par importer les dépendances nécessaires, notamment React, useState pour gérer l'état local, et les styles Bootstrap.

État Local : Trois états sont définis pour stocker le nom, le prénom et le commentaire de l'utilisateur. Ces états sont initialisés à des chaînes vides.

Gestion de la Soumission : La fonction handleSubmit est déclenchée lors de l'envoi du formulaire. Elle empêche le comportement par défaut du formulaire et récupère les messages existants du local storage, ou initialise un tableau vide si aucun message n'est trouvé.

Création d'un Nouveau Message : Un nouvel objet de message est créé avec un identifiant unique, le nom, le prénom, le commentaire et un indicateur d'approbation initialisé à false.

Mise à Jour du Local Storage : Le nouveau message est ajouté à la liste des messages existants, et le tableau mis à jour est enregistré dans le local storage.

Réinitialisation du Formulaire : Les champs du formulaire sont réinitialisés après la soumission, et une alerte informe l'utilisateur que son message a été soumis et est en attente d'approbation.

Rendu du Composant : Le composant retourne un JSX contenant un en-tête, un formulaire avec des champs pour le nom, le prénom et le commentaire, ainsi qu'un bouton d'envoi. Il inclut également le composant NavigatorF pour la navigation et Address pour les informations de contact.

* **Documentation du Composant HomeF**

Le composant HomeF agit comme la page d'accueil de l'application. Il importe plusieurs sous-composants pour construire la structure de la page.

Importation des Composants : Il importe NavigatorF pour la navigation, Hero, Main, Main1, et Home1 pour le contenu principal de la page, ainsi que Address pour les informations de contact.

Rendu du Composant : Le composant retourne un fragment JSX qui inclut tous les sous-composants dans un ordre spécifique. Cela permet de structurer la page d'accueil en sections distinctes, chacune ayant sa propre fonction visuelle et interactive.

* **Documentation du Composant NavigatorF**

Le composant NavigatorF gère la navigation dans l'application à l'aide de react-router-dom.

Importation des Dépendances : Il importe useNavigate pour gérer le routage et des icônes pour le panier.

Fonctions de Navigation : Plusieurs fonctions sont définies pour naviguer vers différentes pages de l'application, comme la page d'accueil, le menu, le panier, etc. Chaque fonction utilise navigate pour changer de route.

Rendu de la Navigation : Le composant retourne un JSX qui représente une barre de navigation. Cette barre inclut un logo, des liens vers différentes sections de l'application, et un bouton pour accéder au panier. Les éléments de la barre sont stylisés à l'aide de classes Bootstrap et de styles personnalisés.

Déconnexion : Un bouton de déconnexion est également inclus, permettant à l'utilisateur de revenir à la page d'accueil.

## **Accueil**

* **Documentation du Composant Hero**

Le composant Hero est conçu pour accueillir les utilisateurs sur la page d'accueil avec une présentation visuelle attrayante.

Importation des Images : Trois images sont importées pour illustrer le contenu de la section.

Structure du Composant : Le composant utilise une grille Bootstrap pour organiser le contenu. Il est divisé en deux colonnes.

Colonne Principale : Contient un titre de bienvenue et une description motivante sur l'importance de la nourriture pour les étudiants. Un élément img affiche une image principale.

Colonne Secondaire : Comprend une image supplémentaire positionnée avec un style spécifique, créant un effet visuel distinct.

Styles : Des styles en ligne et des classes Bootstrap sont utilisés pour assurer une mise en page cohérente et esthétique.

* **Documentation du Composant Home**

Le composant Home agit comme le conteneur principal de la page d'accueil, intégrant plusieurs sous-composants.

Importation des Composants : Il importe NavigatorA, Hero, Main, Main1, et Address pour construire la page.

Rendu du Composant : Le composant retourne un fragment JSX qui inclut tous les sous-composants dans un ordre logique. Cela permet de structurer le contenu de la page d'accueil de manière fluide et cohérente.

* **Documentation du Composant Home1**

Le composant Home1 affiche des promotions et permet la navigation vers d'autres pages.

Contexte des Promotions : Il utilise PromotionsContext pour accéder aux données des promotions.

Navigation : La fonction VersEven utilise useNavigate pour diriger l'utilisateur vers la page des détails des promotions.

Affichage des Promotions : Le composant utilise slice pour afficher uniquement les trois premières promotions. Chaque promotion est présentée dans une carte avec une image, un titre, une description, et un lien pour en savoir plus.

Bouton de Retour : Un bouton permet de revenir à la page précédente, avec une image intégrée pour l'icône.

Connexion : Un bouton de connexion est également inclus, permettant à l'utilisateur de se connecter facilement.

* **Documentation du Composant Main**

Le composant Main présente une image et une description d'un plat.

Importation de l'Image : Une image est importée pour illustrer le contenu.

Structure : Le composant utilise une grille Bootstrap pour disposer l'image et le texte dans deux colonnes.

Colonne de l'Image : Affiche une image centrée.

Colonne du Texte : Contient un titre, une description du plat, et un bouton pour en savoir plus.

Styles : Des styles personnalisés sont appliqués pour améliorer l'aspect visuel, notamment pour le texte et le fond.

* **Documentation du Composant Main1**

Le composant Main1 affiche une liste d'articles avec des options de commande.

Liste d'Articles : Un tableau d'objets items est défini pour représenter les articles à afficher, bien que les images soient pour l'instant vides.

Navigation : La fonction VersA permet de naviguer vers la page des produits.

Affichage des Articles : Chaque article est affiché dans une carte avec une image, une description, et un bouton pour commander.

Mise en Page : Utilisation de Bootstrap pour structurer les cartes en colonnes, assurant une présentation claire et accessible.

* **Documentation du Composant NavigatorA**

Le composant NavigatorA gère la navigation dans l'application.

Importation des Dépendances : Il importe useNavigate pour gérer le routage et une image pour le logo.

Fonctions de Navigation : Plusieurs fonctions sont définies pour naviguer vers différentes pages de l'application, comme l'accueil, le contact, la connexion, et l'administration.

Structure de la Navigation : Le composant retourne un élément de navigation Bootstrap. Il inclut un logo, des liens vers différentes sections, et des boutons pour se connecter et accéder à l'administration.

Styles : Des styles personnalisés et des classes Bootstrap sont utilisés pour assurer une mise en page et une esthétique cohérentes.

## **Admin**

* **Documentation du Composant AccueilA**

Le composant AccueilA sert de tableau de bord pour l'administrateur, affichant des statistiques et des graphiques.

Importation de Bibliothèques : Utilise recharts pour les graphiques et NavigaAdmin pour la navigation.

Données : Contient un tableau de données pour le graphique, représentant des valeurs mensuelles.

Structure du Composant :

Navigation : Intègre le composant NavigaAdmin.

Titre : Affiche un titre de bienvenue.

Cartes d'Informations : Trois cartes affichent des informations statistiques, avec des titres et des valeurs.

Graphiques : Utilise LineChart pour représenter les données de manière visuelle.

Adresse : Inclut le composant Address à la fin.

* **Documentation du Composant AcceuilEmploye**

Le composant AcceuilEmploye est une page simple dédiée aux employés.

Structure : Affiche un message statique "AcceuilEmploye".

Utilisation : Peut être étendu pour afficher des informations sur les employés ou d'autres fonctionnalités.

* **Documentation du Composant Client**

Le composant Client gère l'affichage des clients et leurs informations.

Navigation : Intègre le composant NavigaAdmin.

Recherche : Inclut un champ de recherche pour filtrer les clients.

Cartes d'Informations : Affiche des cartes pour deux clients avec des titres et des valeurs.

Tableau : Présente une liste de clients avec des colonnes pour le nom, le nombre de commandes, la photo, et des actions (comme supprimer).

Adresse : Inclut le composant Address à la fin.

* **Documentation du Composant Employe**

Le composant Employe permet de gérer les employés avec des fonctionnalités d'ajout, de modification et de suppression.

État Local : Utilise useState pour gérer les employés, les détails de l'employé sélectionné, et l'affichage du modal.

Fonctions :

handleEditClick : Prépare les données de l'employé sélectionné pour édition.

handleSave : Met à jour les informations de l'employé.

handleDelete : Supprime l'employé de la liste.

Tableau : Affiche une liste d'employés avec des boutons pour éditer ou supprimer.

Modal : Utilisé pour modifier les détails d'un employé, y compris la possibilité de télécharger une photo.

Adresse : Inclut le composant Address à la fin.

* **Documentation du Composant MenuA**

Le composant MenuA gère l'ajout et la gestion des produits.

Contexte des Produits : Utilise ProductContext pour gérer les produits.

État Local : Gère l'état des nouveaux produits et des produits actuels.

Fonctions :

handleInputChange : Met à jour les détails d'un nouveau produit.

handleAddProduct : Ajoute un produit au contexte.

handleShowModal : Ouvre un modal pour modifier un produit.

handleUpdateProduct : Met à jour un produit existant.

handleDeleteProduct : Supprime un produit.

Tabs : Utilise Tab pour basculer entre l'ajout de produits et la gestion des produits.

Modal : Permet de modifier les détails d'un produit.

Adresse : Inclut le composant Address à la fin.

* **Documentation du Composant NavigaAdmin**

Le composant NavigaAdmin gère la navigation dans l'interface d'administration.

Navigation : Utilise useNavigate pour naviguer vers différentes sections de l'application.

Structure : Comprend une barre de navigation avec des liens vers l'accueil, les employés, les clients, les promotions, et le menu.

Déconnexion : Inclut un bouton pour déconnecter l'utilisateur.

* **Documentation du Contexte ProductContext**

Le contexte ProductContext gère l'état des produits dans l'application.

État Local : Utilise useState pour stocker une liste de produits.

Fonctionnalités :

addProduct : Ajoute un nouveau produit à la liste.

deleteProduct : Supprime un produit de la liste.

Provider : Fournit l'état des produits et les fonctions associées aux composants enfants.

* **Documentation du Composant Promotion**

Le composant Promotion permet de gérer les promotions.

Contexte des Promotions : Utilise PromotionsContext pour accéder et modifier les promotions.

État Local : Gère les détails de la promotion à ajouter ou modifier.

Fonctions :

handleAddPromotion : Ajoute une nouvelle promotion.

handleUpdatePromotion : Met à jour une promotion existante.

handleDeletePromotion : Supprime une promotion.

handleImageChange : Gère le téléchargement d'images pour les promotions.

Tabs : Permet de basculer entre l'ajout et la gestion des promotions.

Modal : Utilisé pour modifier les détails d'une promotion.

Adresse : Inclut le composant Address à la fin.

* **Documentation du Contexte PromotionContext**

Le contexte PromotionContext permet de gérer les promotions dans l'application.

Création du Contexte : Utilise createContext pour créer un contexte pour les promotions.

État Local : Utilise useState pour stocker une liste de promotions avec des détails comme la date, l'image, le titre, la description et le lien.

Fonctions :

addPromotion : Ajoute une nouvelle promotion à la liste.

updatePromotion : Met à jour une promotion existante en fonction de son index.

deletePromotion : Supprime une promotion en filtrant par index.

Provider : Fournit les promotions et les fonctions associées aux composants enfants via PromotionsContext.Provider.

* **Documentation du Composant Signadmin**

Le composant Signadmin gère la connexion des utilisateurs en fonction de leur rôle.

État Local : Utilise useState pour gérer le rôle sélectionné par l'utilisateur.

Navigation : Utilise useNavigate pour rediriger l'utilisateur vers la page appropriée en fonction de son rôle.

Fonctions :

handleRoleChange : Met à jour l'état du rôle sélectionné.

handleLogin : Redirige vers la page correspondante selon le rôle choisi (gérant, employé, administrateur).

Interface Utilisateur :

Logo : Affiche un logo en haut de la carte de connexion.

Sélecteur de Rôle : Permet à l'utilisateur de choisir son rôle.

Champs de Formulaire : Inclut des champs pour l'email et le mot de passe.

Bouton de Connexion : Active ou désactive en fonction de la sélection du rôle.

## **Contact**

* **Documentation du Composant Contact**

Le composant Contact permet aux utilisateurs de soumettre des commentaires et des informations personnelles.

État Local : Utilise useState pour gérer la quantité, initialisée à 1. Cela pourrait être utilisé pour d'autres fonctionnalités liées à des quantités, même si ce n'est pas visible dans le code actuel.

Calcul du Prix Total : Calcule le prix total basé sur la quantité et un prix unitaire fixe.

Gestion des Événements :

handleQuantityChange : Met à jour l'état de la quantité en fonction de l'entrée de l'utilisateur.

Interface Utilisateur :

Navigation : Inclut le composant NavigatorA pour la navigation.

Formulaire de Commentaire : Permet à l'utilisateur d'entrer son nom, prénom et commentaire.

Bouton de Soumission : Un bouton pour soumettre le commentaire.

* **Documentation du Composant NavigatorC**

Le composant NavigatorC représente la barre de navigation de l'application.

Navigation : Utilise une barre de navigation Bootstrap pour permettre aux utilisateurs de naviguer entre différentes sections de l'application.

Logo : Affiche un logo à gauche de la barre de navigation.

Liens de Navigation : Comprend des liens vers les sections Accueil, Menu et Contact.

Bouton de Connexion : Inclut un bouton pour se connecter, positionné à droite.

## **Login**

* **Composant Connexion**

Le composant Connexion gère le processus d'authentification des utilisateurs.

État Local : Utilise useState pour gérer l'email, le mot de passe et les messages d'erreur.

Navigation : Utilise useNavigate de react-router-dom pour rediriger les utilisateurs après une connexion réussie.

Validation des Identifiants : Vérifie les identifiants par rapport à des valeurs par défaut ou via une API.

Gestion des Événements :

handleLogin : Gère la soumission du formulaire et l'authentification.

Interface Utilisateur :

Formulaire de Connexion : Comprend des champs pour l'email et le mot de passe, avec des messages d'erreur conditionnels.

## **Sign**

* **Composant CreerCompte**

Le composant CreerCompte gère la création d'un nouveau compte utilisateur.

État Local : Utilise useState pour gérer le nom, l'email et le mot de passe.

Navigation : Utilise useNavigate pour rediriger les utilisateurs après la création du compte.

Gestion des Événements :

handleSignUp : Gère la soumission du formulaire.

Interface Utilisateur :

Formulaire de Création de Compte : Comprend des champs pour le nom, l'email et le mot de passe.

## **Employe**

* **AccueilE.js**

Fonctionnalité Principale : Affiche le tableau de bord de l'employé.

Composants Inclus :

NavEmploye : Barre de navigation pour accéder aux différentes sections de l'application.

Graphique : Utilise LineChart pour afficher des statistiques mensuelles.

Cartes d'Information : Affiche des informations clés sous forme de cartes.

Données : Utilise un tableau de données statiques pour le graphique.

* **CommandeContext.js**

Fonctionnalité Principale : Gère l'état des commandes dans l'application.

Fonctions Incluses :

addCommande : Ajoute une nouvelle commande à l'état.

Utilisation : Fournit un contexte pour accéder aux commandes dans d'autres composants.

* **CommandeE.js**

Fonctionnalité Principale : Affiche la liste des commandes et permet de modifier leur statut.

Composants Inclus :

NavEmploye : Barre de navigation.

Tableau : Affiche les détails des commandes, y compris ID, articles, quantité, total et statut.

Fonctionnalités :

Changement de Statut : Permet de basculer entre "En cours" et "Prêt" pour chaque commande.

Affichage des Articles : Montre les articles dans chaque commande.

* **Employe2.js**

Fonctionnalité Principale : Composant simple affichant le nom de l'employé.

Composants Inclus :

NavEmploye : Barre de navigation.

* **MenuEmploye.js**

Fonctionnalité Principale : Affiche la liste des produits disponibles à la commande.

Composants Inclus :

NavEmploye : Barre de navigation.

Liste de Produits : Affiche chaque produit avec image, titre, description et bouton de commande.

Données : Utilise le contexte ProductContext pour récupérer les produits.

* **NavEmploye.js**

Fonctionnalité Principale : Fournit une navigation entre les différentes sections de l'application.

Liens Inclus :

Accès à "Accueil", "Commande", "Menu", "Réclamations".

Bouton de déconnexion pour quitter l'application.

* **PanierContext.js**

Fonctionnalité Principale : Gère l'état du panier d'achats.

Fonctions Incluses :

ajouterAuPanier : Ajoute un article au panier.

supprimerDuPanier : Supprime un article du panier.

viderPanier : Vide le panier.

Utilisation : Fournit un contexte pour accéder aux informations du panier dans d'autres composants.

* **ReclamationsEmploye.js**

Fonctionnalité Principale : Affiche la liste des réclamations et permet de répondre à celles-ci.

Composants Inclus :

NavEmploye : Barre de navigation.

Tableau : Affiche les réclamations avec les détails du nom, prénom, commentaire, réponse et action.

Fonctionnalités :

Modal de Réponse : Permet à l'employé de répondre à une réclamation via un modal.

Gestion des Réponses : Stocke les réponses dans localStorage et indique si elles sont approuvées ou non.

## **Evènement**

* **Even.js**

Fonctionnalité Principale : Affiche une liste de promotions ou d'événements.

Composants Inclus :

PromotionsContext : Utilise le contexte pour accéder à la liste des promotions.

Structure :

Affichage des Promotions :

Pour chaque promotion, un élément de carte est créé avec :

Image : Affiche l'image de la promotion.

Titre : Affiche le titre de la promotion.

Description : Affiche une brève description de la promotion.

Date : Affiche la date de la promotion sous forme de badge.

Lien : Un lien pour "Lire la suite" vers plus d'informations.

Style : Utilise des classes Bootstrap pour le style et une couleur de fond personnalisée.

* **Final.js**

Fonctionnalité Principale : Composant principal qui regroupe plusieurs sous-composants pour l'affichage final.

Composants Inclus :

NavigatorF : Barre de navigation pour l'application.

TitreE : Affiche le titre et la barre de recherche.

Even : Affiche la liste des promotions.

Address : Affiche l'adresse ou des informations supplémentaires.

Structure : Organisé en un seul composant fonctionnel qui inclut tous les sous-composants dans un format de fragment.

* **TitreE.js**

Fonctionnalité Principale : Affiche le titre de la section "Événements" et une barre de recherche.

Structure :

Titre : Un titre stylisé "Événement" avec une police personnalisée.

Barre de Recherche : Un champ de texte pour rechercher des événements, avec un style personnalisé.

Séparateur : Une ligne horizontale pour séparer la section du contenu principal.

Style : Utilise des classes Bootstrap et des styles en ligne pour personnaliser l'apparence.

## **Gérant**

* **AccueilG.js**

Fonctionnalité Principale : Affiche le tableau de bord pour le gérant avec des statistiques et des informations.

Composants Inclus :

NavGerant : Barre de navigation pour accéder aux différentes sections.

Address : Affiche des informations supplémentaires.

Structure :

Titre : "Bienvenue to Dashboard" avec un style personnalisé.

Cartes d'Information : Trois cartes affichant des données (nom et nombres) dans un style sombre.

Graphique : Utilise recharts pour afficher des statistiques sous forme de graphique linéaire.

Style : Utilisation de Bootstrap pour la mise en page et des styles personnalisés pour le texte.

* **CommandesGerant.js**

Fonctionnalité Principale : Composant placeholder pour gérer les commandes.

Structure :

Un simple div qui affiche "CommandesGerant".

Utilisation : À développer pour gérer les commandes spécifiques au gérant.

* **EmployeGerant.js**

Fonctionnalité Principale : Gère les employés, permettant d'ajouter, modifier et supprimer des employés.

Composants Inclus :

NavGerant : Barre de navigation.

Address : Affiche des informations supplémentaires.

Structure :

Ajout d'Employé : Formulaire pour ajouter un nouvel employé avec validation.

Liste des Employés : Affiche une table avec les employés existants, permettant la modification et la suppression.

Modal : Utilisé pour modifier les informations d'un employé.

Stockage : Les employés sont stockés dans localStorage.

* **NavGerant.js**

Fonctionnalité Principale : Barre de navigation pour le gérant.

Structure :

Liens vers les différentes sections : Accueil, Commandes, Employés, Réclamations.

Bouton de déconnexion.

Navigation : Utilise useNavigate de react-router-dom pour changer de route.

* **Reclamations.js**

Fonctionnalité Principale : Gère les réclamations des utilisateurs, permettant d'approuver ou de rejeter des réponses.

Composants Inclus :

NavGerant : Barre de navigation.

Address : Affiche des informations supplémentaires.

Structure :

Liste des Réclamations : Affiche les réclamations en attente de validation dans une table.

Actions : Boutons pour approuver ou rejeter les réponses aux réclamations.

Stockage : Les réclamations sont stockées dans localStorage, et les mises à jour sont synchronisées.

## **Historique**

* **CommandeContext.js**

Fonctionnalité Principale : Fournit un contexte pour gérer l'état des commandes dans l'application.

Composants Inclus :

createContext : Crée un contexte pour les commandes.

useState : Utilisé pour suivre les commandes ajoutées.

Fonctions :

addCommande : Ajoute une nouvelle commande à l'état.

Structure :

Le contexte CommandeContext est exporté pour être utilisé dans d'autres composants.

* **HistF.js**

Fonctionnalité Principale : Composant principal pour l'historique des commandes, regroupant plusieurs sous-composants.

Composants Inclus :

NavigatorF : Barre de navigation.

TitreH : Titre et barre de recherche pour l'historique.

Historique : Affiche l'historique des commandes.

Address : Affiche des informations supplémentaires.

Structure : Utilise une structure simple avec des composants imbriqués pour créer une page cohérente.

* **Historique.jsx**

Fonctionnalité Principale : Affiche l'historique des commandes à partir du contexte.

Composants Inclus :

CommandeContext : Utilisé pour récupérer les commandes.

Structure :

Tableau des Commandes : Affiche les commandes sous forme de tableau avec des colonnes pour le nom, le montant, les points et la date.

Gestion des Commandes : Affiche un message si aucune commande n'est trouvée.

Style : Utilisation de Bootstrap pour la mise en page et des styles personnalisés pour le texte.

* **TitreH.js**

Fonctionnalité Principale : Affiche le titre de la page et un champ de recherche.

Structure :

Titre : "Historique" avec un style personnalisé.

Champ de Recherche : Permet à l'utilisateur de rechercher dans l'historique des commandes.

Séparation Visuelle : Une ligne horizontale pour séparer le titre du contenu.

Style : Utilisation de classes Bootstrap pour la mise en page et des styles personnalisés.

## **Menu**

* **Menu.js**

Fonctionnalité Principale : Affiche le menu des produits disponibles et permet la navigation vers la page de commande.

Composants Inclus :

ProductContext : Utilisé pour récupérer la liste des produits.

NavigatorF : Barre de navigation supérieure.

Address : Affiche des informations supplémentaires en bas du menu.

Fonctions :

VersA : Fonction qui utilise useNavigate pour rediriger l'utilisateur vers la page des produits lors du clic sur le bouton "Commander".

Structure :

Titre du Menu : Un titre stylisé pour le menu.

Barre de Recherche : Permet à l'utilisateur de rechercher des produits.

Liste des Produits : Affiche les produits sous forme de cartes avec image, titre, description et un bouton pour commander.

Style : Utilisation de Bootstrap pour la mise en page et des styles personnalisés pour le texte et les cartes.

* **NavigatorM.js**

Fonctionnalité Principale : Barre de navigation pour l'application.

Composants Inclus :

Logo : Affiche le logo de l'application.

Structure :

Liens de Navigation : Liens vers les pages Accueil, Menu et Contact.

Bouton de Connexion : Un bouton stylisé pour la connexion.

Style : Utilisation de classes Bootstrap pour la mise en page et des styles personnalisés pour les éléments de navigation.

## **Panier Etudiant**

* **Achat.js**

Fonctionnalité Principale : Fournit un contexte pour gérer les articles du panier.

Composants Inclus :

CartContext : Crée un contexte pour stocker les articles du panier.

Fonctions :

addToCart : Ajoute un article au panier en mettant à jour l'état cartItems.

Structure :

Provider : Enveloppe les enfants avec le contexte du panier, permettant l'accès aux données du panier dans d'autres composants.

* **AchatsFinal.js**

Fonctionnalité Principale : Affiche les détails du panier et permet le paiement.

Composants Inclus :

CartContext : Utilisé pour récupérer les articles du panier.

CommandeContext : Utilisé pour ajouter une commande au contexte.

NavigatorF : Barre de navigation supérieure.

Address : Affiche des informations supplémentaires en bas de la page.

Fonctions :

handleApplyCode : Applique un code promotionnel et calcule les points.

handlePay : Prépare les détails de la commande et les ajoute au contexte des commandes.

Structure :

Liste des Articles : Affiche les articles du panier avec leurs détails (image, titre, quantité, total).

Total du Panier : Affiche le montant total et permet d'appliquer un code promotionnel.

Bouton de Paiement : Permet à l'utilisateur de procéder au paiement.

* **Address.js**

Fonctionnalité Principale : Affiche des informations de contact et des liens utiles en bas de page.

Structure :

À propos de nous : Présente l'entreprise.

Liens rapides : Fournit des liens vers d'autres sections du site.

Réseaux sociaux : Liens vers les profils de réseaux sociaux de l'entreprise.

Bas de page : Mention des droits d'auteur.

* **NavigatorP.js**

Fonctionnalité Principale : Barre de navigation pour la section du panier.

Composants Inclus :

Logo : Affiche le logo de l'application.

Structure :

Liens de Navigation : Liens vers les pages Accueil, Menu, Promotion, Événement, Mon Panier, et Parrainage.

Style : Utilisation de classes Bootstrap pour la mise en page et des styles personnalisés pour les éléments de navigation

## **Produit**

* **Produit.js**

Fonctionnalité Principale : Affiche les détails d'un produit et permet à l'utilisateur d'ajouter ce produit à son panier.

## **Parrainage**

* **Composant Parrains**

Fonctionnalité Principale : Affiche la page de parrainage avec les sections pour les points et les informations d'adresse.

Composants Inclus :

NavigatorF : Barre de navigation pour la navigation dans l'application.

TitreP : Titre de la page de parrainage.

PointsF : Affiche les points et les informations relatives aux parrains.

Address : Section pour afficher ou gérer l'adresse de l'utilisateur.

* **Composant Points**

Fonctionnalité Principale : Affiche les points accumulés par l'utilisateur.

Structure :

Titre : "Vos Points" avec un style personnalisé.

Affichage des Points : Montre le nombre total de points et une valeur associée (ex. 30.000 et 30 pts).

* **Composant Points1**

Fonctionnalité Principale : Affiche les informations concernant les parrains de l'utilisateur.

Structure :

Titre : "Vos Parrains" avec un style personnalisé.

Affichage des Commandes : Montre le nombre de commandes et le nombre de parrains.

* **Composant PointsF**

Fonctionnalité Principale : Regroupe les informations sur les points et les parrains.

Structure :

Barre de Recherche : Permet à l'utilisateur de rechercher des informations.

Cartes d'Informations : Affiche des cartes avec des informations sur des parrains spécifiques.

Tableau : Liste des parrains avec leurs commandes et une image associée.

* **Composant TitreP**

Fonctionnalité Principale : Affiche le titre de la section "Vos Parrains".

Structure :

Titre : "Vos Parrains" avec un style personnalisé.

Barre de Recherche : Permet à l'utilisateur de rechercher des parrains.

Styles et Mise en Page

Utilisation de classes Bootstrap pour la mise en page et des styles personnalisés pour le texte.

Les couleurs et polices sont spécifiées pour correspondre à l'esthétique de l'application.

## **Main**

* **Main.js**

**Fonctionnalité Principale :**

Initialise et rend l'application React en utilisant le mode strict pour détecter les problèmes potentiels.

Importations :

React et ReactDOM :

StrictMode : Utilisé pour activer des vérifications supplémentaires sur le code React.

createRoot : Méthode pour créer un point d'entrée pour le rendu de l'application.

**Composants :**

**App** : Composant principal de l'application, qui contient toute la logique et les routes.

Styles :

**index.css** : Fichier CSS pour les styles globaux de l'application.

**Bootstrap** :

CSS et JS de Bootstrap pour utiliser les composants et styles pré-définis.

**Contextes :**

ProductProvider : Fournisseur de contexte pour gérer l'état des produits dans l'application.

CartProvider : Fournisseur de contexte pour gérer le panier d'achats.

CommandeProvider : Fournisseur de contexte pour l'historique des commandes.

PanierProvider : Fournisseur de contexte pour la gestion du panier par les employés.

Structure de Rendu :

Utilisation de createRoot pour rendre l'application dans l'élément HTML avec l'ID root.

Les différents fournisseurs de contexte sont imbriqués pour permettre un accès facile à l'état global dans toute l'application.

Le composant App est rendu à l'intérieur de ces fournisseurs, ce qui lui permet d'accéder à tous les contextes définis.

**Composants Inclus :**

NavigatorF : Barre de navigation supérieure pour la navigation dans l'application.

CartContext : Utilisé pour ajouter des articles au panier.

**État et Variables :**

quantity : État pour gérer la quantité sélectionnée par l'utilisateur (initialisé à 1).

pricePerUnit : Prix unitaire du produit fixé à 10.

totalPrice : Calculé comme le produit de quantity et pricePerUnit.

**Fonctions :**

handleQuantityChange : Met à jour l'état quantity lorsque l'utilisateur change la quantité dans l'input.

handleAddToCart : Crée un objet item contenant les détails du produit (titre, quantité, prix total, image) et utilise addToCart pour l'ajouter au panier. Ensuite, redirige l'utilisateur vers la page du panier.

**Structure :**

Image du Produit : Affiche une image du produit ().

Détails du Produit : Titre et description du produit.

Formulaire de Quantité : Permet à l'utilisateur de sélectionner la quantité souhaitée.

Prix Total : Affiche le prix total basé sur la quantité sélectionnée ().

Bouton Ajouter au Panier : Permet à l'utilisateur d'ajouter le produit au panier.

**Styles :**

Utilisation de classes Bootstrap pour la mise en page et des styles personnalisés pour le texte.

# Linting

ESLint est configuré pour aider à maintenir la qualité du code. Nous pouvons l'exécuter avec la commande :

« npm run lint »

# Conclusion

Cette documentation fournit un aperçu des composants principaux du dossier, détaillant leur rôle et leur structure. Cela facilitera la compréhension et la maintenance du code.