Application de Facturation Complète

Une application web moderne et complète pour la gestion de factures, développée avec React, Node.js, et stockage JSON. Interface entièrement en français avec génération de PDF professionnels.

🌟 Fonctionnalités principales

✅ Interface utilisateur complète

- **Page d'accueil** avec navigation intuitive
- **Gestion complète des factures** (CRUD)
- **Recherche et filtrage** par client, période, montant
- **Pagination** pour une navigation fluide
- **Interface responsive** adaptée à tous les écrans

✅ Gestion de factures

- **Création de factures** avec formulaire interactif
- **Modification en temps réel** de factures existantes
- **Calculs automatiques** des montants et totaux
- **Validation** des données côté client et serveur
- **Numérotation automatique** des factures

✅ Fonctionnalités avancées

- **Export PDF** avec mise en forme professionnelle
- **Formatage français** des dates (DD/MM/YYYY) et devises (€)
- **Stockage persistant** avec fichiers JSON

- **API RESTful** complète avec gestion d'erreurs
- **Interface entièrement en français**

✅ Gestion des clients

- Informations client complètes (nom, entreprise, téléphone, adresse)
- Recherche rapide par nom de client ou entreprise
- Historique des factures par client

🚀 Installation et démarrage

Prérequis

- Node.js 18+
- pnpm (ou npm)

Installation

1. **Cloner ou télécharger le projet**
   `bash cd billing-app`
2. **Installer les dépendances du backend**
   `bash cd backend pnpm install`
3. **Installer les dépendances du frontend**
   `bash cd ../frontend pnpm install`

Démarrage

1. **Démarrer le serveur backend**
   `bash cd backend pnpm start # ou pour le développement: pnpm run dev`
   Le serveur démarre sur http://localhost:3001

2. **Démarrer l'interface frontend**
   `bash cd frontend pnpm run dev`
   L'application est accessible sur http://localhost:5173

📁 Structure du projet

```
billing-app/
├── backend/                    # API Node.js/Express
│   ├── database/               # Système de stockage JSON
│   │   ├── storage.js          # Gestionnaire de base de données
JSON
│   │   └── data/               # Fichiers de données (auto-créés)
│   │       ├── factures.json   # Données des factures
│   │       └── lignes.json     # Lignes de facturation
│   ├── server.js               # Serveur Express principal
│   └── package.json            # Dépendances backend
├── frontend/                   # Application React
│   ├── src/
│   │   ├── pages/              # Pages principales
│   │   │   ├── Accueil.tsx     # Page d'accueil
│   │   │   ├── ListeFactures.tsx    # Liste et recherche
│   │   │   ├── CreerFacture.tsx     # Création de facture
│   │   │   ├── DetailFacture.tsx    # Détails d'une facture
│   │   │   └── ModifierFacture.tsx  # Modification
│   │   ├── components/         # Composants réutilisables
│   │   └── App.tsx            # Application principale
│   └── package.json            # Dépendances frontend
└── README.md                   # Cette documentation
```

🔧 API REST - Endpoints

Factures

- `GET /api/factures` - Liste des factures avec pagination et filtres
- `GET /api/factures/:id` - Détails d'une facture
- `POST /api/factures` - Créer une nouvelle facture
- `PUT /api/factures/:id` - Modifier une facture
- `DELETE /api/factures/:id` - Supprimer une facture
- `GET /api/factures/:id/pdf` - Télécharger le PDF d'une facture

Utilitaires

- `GET /api/health` - État de santé de l'API
- `GET /api/stats` - Statistiques générales

Paramètres de recherche (GET /api/factures)

- `page` : Numéro de page (défaut: 1)
- `limit` : Nombre d'éléments par page (défaut: 10)
- `search` : Recherche par nom client, entreprise ou numéro de facture
- `dateDebut` : Filtrer à partir de cette date (format: YYYY-MM-DD)
- `dateFin` : Filtrer jusqu'à cette date (format: YYYY-MM-DD)

📊 Utilisation

Créer une nouvelle facture

1. Cliquer sur "Créer une nouvelle facture" depuis l'accueil
2. Remplir les informations client (nom requis)

3. Ajouter les lignes d'articles/prestations
4. Le montant total se calcule automatiquement
5. Cliquer sur "Créer la facture"

Gérer les factures

1. Accéder à la liste des factures
2. Utiliser les filtres pour rechercher
3. Cliquer sur les actions : Voir, Modifier, Télécharger PDF, Supprimer

Exporter en PDF

- Cliquer sur l'icône de téléchargement dans la liste
- Ou utiliser le bouton "Télécharger PDF" dans les détails
- Le PDF est généré avec une mise en forme professionnelle

🎨 Fonctionnalités interface

Formatage français

- **Dates** : Format DD/MM/YYYY partout
- **Devises** : Format français avec € (ex: 1 234,56 €)
- **Nombres** : Séparateurs français (virgule pour les décimales)

Validation et sécurité

- **Validation côté client** : Formulaires avec contrôles en temps réel
- **Validation côté serveur** : API avec validation des données
- **Gestion d'erreurs** : Messages d'erreur explicites en français
- **Calculs automatiques** : Évite les erreurs de saisie

Expérience utilisateur

- **Interface intuitive** : Navigation claire et logique
- **Feedback visuel** : Confirmations et messages d'état
- **Responsive design** : Fonctionne sur mobile et desktop
- **Performance** : Chargement rapide avec pagination

🛠️ Développement

Données d'exemple

L'application inclut des données d'exemple pour la démonstration :
- 3 factures pré-créées avec différents clients
- Lignes de facturation variées
- Montants et dates réalistes

Personnalisation

- **Informations entreprise** : Modifiables dans `server.js` (section PDF)
- **Styles** : Interface basée sur Tailwind CSS
- **Stockage** : Facilement extensible vers une base de données

Tests

- Tester l'API avec un client REST (Postman, curl)
- Interface testable directement dans le navigateur
- Génération PDF testable via l'interface

⚡ Production

Déploiement

1. **Backend** : Déployer le serveur Node.js sur votre hébergeur
2. **Frontend** : Construire avec `pnpm run build` et déployer les fichiers statiques
3. **Configuration** : Ajuster les URLs dans le frontend pour pointer vers votre API

Sauvegardes

- Les données sont stockées dans `backend/database/data/`
- Sauvegarder ces fichiers JSON pour préserver les données
- Simple restauration par copie des fichiers

🆘 Support

Résolution de problèmes

- **Port déjà utilisé** : Modifier le PORT dans server.js
- **CORS errors** : Vérifier que le backend est démarré
- **Données perdues** : Vérifier les fichiers JSON dans database/data/

Logs et debugging

- Logs serveur affichés dans la console backend
- Erreurs frontend visibles dans la console du navigateur
- API de santé : GET /api/health pour vérifier l'état

📄 Licence

Ce projet est fourni à des fins éducatives et de démonstration. Libre d'utilisation et modification.

Développé avec ❤️ en français pour une utilisation professionnelle