# Résumé Final des Fonctionnalités - Riziky-Boutic

## Vue d'Ensemble Exécutive

Riziky-Boutic est une plateforme e-commerce complète qui combine une interface utilisateur moderne avec un système de gestion administratif avancé. Cette documentation présente un résumé exhaustif de toutes les fonctionnalités implémentées, leur utilisation et leur manipulation technique.

---

## Fonctionnalités Clients (Frontend)

### Page d'Accueil et Navigation

**Localisation :** `src/pages/HomePage.tsx`, `src/components/layout/Navbar.tsx`

**Fonctionnalités Principales :**

- **Hero Section** : Bannière d'accueil avec call-to-action
- **Produits Vedettes** : Carrousel automatique des produits mis en avant
- **Catégories Populaires** : Navigation rapide par catégories
- **Promotions Actives** : Affichage des ventes flash et promotions
- **Navigation Responsive** : Menu adaptatif mobile/desktop

**Comment Utiliser :**

```
// Modifier les produits vedettes
const featuredProducts = await productsService.getFeatured()

// Personnaliser la hero section
<HeroSection
  title="Nouvelle Collection 2024"
  subtitle="Découvrez nos dernières tendances"
  backgroundImage="/images/hero-bg.jpg"
/>
```

**Manipulation Technique :**

- Modifier `src/components/home/HomeHeader.tsx` pour le contenu hero
- Ajuster `src/components/home/FeaturedProductsCarousel.tsx` pour les produits
- Configurer `src/hooks/useHomePageData.ts` pour les données

### Recherche et Filtrage

**Localisation :** `src/components/search/`, `src/components/filters/`

**Fonctionnalités Avancées :**

- **Recherche Textuelle** : Recherche par nom, description, catégorie
- **Filtres Dynamiques** : Prix, stock, catégories, promotions
- **Tri Intelligent** : Par prix, popularité, nouveauté, notes
- **Recherche Instantanée** : Suggestions en temps réel
- **Historique de Recherche** : Mémorisation des recherches récentes

**Implementation :**

```
// Hook de recherche personnalisé
const {
  searchQuery,
  setSearchQuery,
  filteredProducts,
  activeFilters,
  applyFilter,
  clearFilters
} = useProductFilters()

// Composant de recherche
<AdvancedSearchBar
  onSearch={setSearchQuery}
  onFilterChange={applyFilter}
  placeholder="Rechercher un produit..."
/>
```

**Configuration Filters :**

```
// Types de filtres disponibles
interface ProductFilters {
  category?: string[]
  priceRange?: [number, number]
  inStock?: boolean
  onSale?: boolean
  rating?: number
  brand?: string[]
}
```

### Gestion du Panier

**Localisation :** `src/components/cart/`, `src/hooks/useCart.ts`

**Fonctionnalités Complètes :**

- **Panier Latéral** : Drawer responsive avec aperçu rapide
- **Gestion Quantités** : Incrémentation/décrémentation avec validation stock

- **Calculs Automatiques** : Sous-total, TVA, frais de port, total
- **Persistance** : Sauvegarde locale et synchronisation serveur
- **Livraison Gratuite** : Seuil automatique et indication progression

**API du Hook useCart :**

```
const {
  cart,              // Articles du panier
  addToCart,         // (productId, quantity) => Promise<void>
  removeFromCart,    // (itemId) => Promise<void>
  updateQuantity,    // (itemId, newQuantity) => Promise<void>
  clearCart,         // () => Promise<void>
  totalPrice,        // Prix total calculé
  itemCount,         // Nombre total d'articles
  subtotal,          // Sous-total HT
  shipping,          // Frais de livraison
  isLoading          // État des opérations
} = useCart()
```

**Utilisation Avancée :**

```
// Ajouter avec options
await addToCart(productId, quantity, {
  size: 'L',
  color: 'Rouge',
  customization: 'Texte personnalisé'
})

// Validation avant ajout
const canAddToCart = (product: Product, quantity: number) => {
  if (product.stock < quantity) {
    toast.error(`Stock insuffisant (${product.stock} disponible)`)
    return false
  }
  return true
}
```

### Processus de Commande

**Localisation :** `src/pages/CheckoutPage.tsx`, `src/components/checkout/`

**Étapes du Checkout :**

1. **Révision Panier** : Vérification finale des articles
2. **Informations Livraison** : Formulaire adresse avec validation
3. **Mode de Paiement** : Sélection et saisie informations paiement
4. **Confirmation** : Récapitulatif et validation finale

5. **Confirmation** : Page de succès avec numéro de commande

**Méthodes de Paiement Supportées :**

```
interface PaymentMethods {
  creditCard: {
    providers: ['Visa', 'Mastercard', 'American Express']
    encryption: 'PCI DSS compliant'
  }
  paypal: {
    integration: 'PayPal SDK'
    express: boolean
  }
  bankTransfer: {
    iban: string
    bic: string
    reference: string
  }
}
```

**Validation Formulaire :**

```
// Schéma de validation Zod
const checkoutSchema = z.object({
  shipping: z.object({
    firstName: z.string().min(2),
    lastName: z.string().min(2),
    address: z.string().min(5),
    city: z.string().min(2),
    postalCode: z.string().regex(/^\d{5}$/),
    country: z.string().min(2)
  }),
  payment: z.object({
    method: z.enum(['card', 'paypal', 'transfer']),
    cardNumber: z.string().optional(),
    expiryDate: z.string().optional(),
    cvv: z.string().optional()
  })
})
```

### Gestion du Profil Utilisateur

**Localisation :** `src/pages/ProfilePage.tsx`, `src/components/profile/`

**Sections du Profil :**

- **Informations Personnelles** : Nom, prénom, email, téléphone, adresse

- **Mot de Passe** : Changement sécurisé avec validation force
- **Photo de Profil** : Upload et recadrage d'image
- **Préférences** : Notifications, langue, devise, newsletter
- **Adresses** : Gestion multiple adresses de livraison

**Composants Spécialisés :**

```
// Formulaire informations personnelles
<PersonalInfoForm
  user={user}
  onUpdate={handleUpdateProfile}
  validationSchema={profileSchema}
/>

// Changement mot de passe
<PasswordForm
  onUpdate={handlePasswordChange}
  requireCurrentPassword={true}
  showStrengthIndicator={true}
/>

// Upload photo de profil
<ProfilePhotoUpload
  currentPhoto={user.avatar}
  onUpload={handlePhotoUpload}
  maxSize={2} // MB
  allowedTypes={['jpg', 'png', 'webp']}
/>
```

### Historique des Commandes

**Localisation :** `src/pages/OrdersPage.tsx`, `src/components/orders/`

**Fonctionnalités de Suivi :**

- **Liste Complète** : Toutes les commandes avec pagination
- **Détail Commande** : Produits, quantités, prix, statut
- **Suivi Livraison** : Étapes de traitement et expédition
- **Factures** : Téléchargement PDF des factures
- **Support** : Contact direct pour questions commande

**États de Commande :**

```
type OrderStatus =
  | 'pending'     // En attente de traitement
  | 'confirmed'   // Confirmée par le marchand
  | 'processing'  // En cours de préparation
```

```
  | 'shipped'     // Expédiée
  | 'delivered'   // Livrée
  | 'cancelled'   // Annulée
  | 'refunded'    // Remboursée

// Composant statut avec couleurs
<OrderStatusBadge
  status={order.status}
  variant="badge" // ou "timeline"
/>
```

### Système d'Avis et Favoris

**Localisation :** `src/components/reviews/`, `src/hooks/useFavorites.ts`

**Avis Produits :**

- **Notation 5 Étoiles** : Système de notation standard
- **Commentaires** : Avis textuels avec modération
- **Photos** : Upload d'images par les clients
- **Votes Utiles** : Like/Dislike des avis par autres clients
- **Réponses Marchands** : Réponses officielles aux avis

**Gestion Favoris :**

```
const {
  favorites,        // Liste des favoris
  isFavorite,       // (productId) => boolean
  addToFavorites,   // (productId) => Promise<void>
  removeFromFavorites, // (productId) => Promise<void>
  toggleFavorite,   // (productId) => Promise<void>
  favoriteCount     // Nombre total de favoris
} = useFavorites()

// Utilisation dans composants
<Button
  onClick={() => toggleFavorite(product.id)}
  variant={isFavorite(product.id) ? "default" : "outline"}
>
  <Heart className={cn(
    "h-4 w-4",
    isFavorite(product.id) && "fill-red-500 text-red-500"
  )} />
</Button>
```

---

## Fonctionnalités Administrateur (Backend Admin)

### Tableau de Bord Administrateur

**Localisation :** `src/pages/admin/AdminDashboardPage.tsx`

**Métriques Temps Réel :**

- **Ventes du Jour** : Chiffre d'affaires et nombre de commandes
- **Produits Populaires** : Top 10 des meilleures ventes
- **Nouveaux Clients** : Inscriptions récentes
- **Stock Faible** : Produits nécessitant réapprovisionnement
- **Avis Récents** : Derniers avis clients nécessitant attention

**Widgets Interactifs :**

```
// Composants de statistiques
<DataStatsCard
  title="Ventes du Jour"
  value="1,250€"
  change="+15.3%"
  trend="up"
  icon={TrendingUp}
/>

// Graphiques de performance
<SalesChart
  data={salesData}
  period="week" // day, week, month, year
  type="line" // bar, pie, donut
/>
```

### Gestion des Produits

**Localisation :** `src/pages/admin/AdminProductsPage.tsx`, `src/components/admin/ProductForm.tsx`

**Fonctionnalités CRUD Complètes :**

- **Création Produits** : Formulaire complet avec validation
- **Upload Multiple** : Images produits avec prévisualisation
- **Gestion Stock** : Quantités et seuils d'alerte
- **Catégorisation** : Assignment multiple catégories
- **SEO** : Meta-description, mots-clés, URL slug
- **Promotions** : Réductions temporaires et permanentes

**Formulaire Produit Avancé :**

```typescript
interface ProductFormData {
  // Informations de base
  nom: string
  description: string
  prix: number
  stock: number

  // Catégorisation
  categories: string[]
  tags: string[]
  brand: string

  // Médias
  images: File[]
  mainImageIndex: number

  // SEO
  metaTitle?: string
  metaDescription?: string
  slug?: string

  // Attributs physiques
  weight?: number
  dimensions?: {
    length: number
    width: number
    height: number
  }

  // Promotion
  promotion?: {
    type: 'percentage' | 'fixed'
    value: number
    startDate: Date
    endDate: Date
  }
}
```

**Gestion des Images :**

```
// Upload multiple avec prévisualisation
<ImageUploadZone
  maxFiles={6}
  acceptedTypes={['jpg', 'png', 'webp']}
  maxSize={5} // MB par fichier
  onUpload={handleImageUpload}
```

```
  onReorder={handleImageReorder}
  showPreview={true}
/>

// Composant de tri des images
<ImageSorter
  images={productImages}
  onSort={handleImageSort}
  onSetMain={handleSetMainImage}
  onDelete={handleDeleteImage}
/>
```

### Gestion des Commandes

**Localisation :** `src/pages/admin/AdminOrdersPage.tsx`

**Fonctionnalités de Traitement :**

- **Vue d'Ensemble** : Liste filtrable et triable des commandes
- **Détail Commande** : Informations complètes client et produits
- **Gestion Statuts** : Workflow de traitement des commandes
- **Impression** : Étiquettes de livraison et factures
- **Communication** : Messages directs avec clients
- **Rapports** : Analyses des ventes par période

**Workflow Commandes :**

```
// États et transitions possibles
const orderWorkflow = {
  'pending': ['confirmed', 'cancelled'],
  'confirmed': ['processing', 'cancelled'],
  'processing': ['shipped', 'cancelled'],
  'shipped': ['delivered', 'returned'],
  'delivered': ['returned', 'refunded'],
  'cancelled': [], // État final
  'returned': ['refunded'],
  'refunded': [] // État final
}

// Composant de gestion de statut
<OrderStatusManager
  order={order}
  currentStatus={order.status}
  availableTransitions={orderWorkflow[order.status]}
  onStatusChange={handleStatusChange}
  requireNotes={['cancelled', 'refunded']}
/>
```

### Gestion des Utilisateurs

**Localisation :** `src/pages/admin/AdminUsersPage.tsx`

**Fonctionnalités de Gestion :**

- **Liste Utilisateurs** : Vue d'ensemble avec recherche et filtres
- **Profils Détaillés** : Informations complètes et historique
- **Gestion Rôles** : Attribution et modification des permissions
- **Activation/Désactivation** : Contrôle d'accès des comptes
- **Statistiques Utilisateur** : Commandes, panier moyen, fréquence

**Système de Rôles :**

```
interface UserRole {
  id: string
  name: string
  permissions: Permission[]
  description: string
}

interface Permission {
  resource: 'products' | 'orders' | 'users' | 'settings'
  actions: ('create' | 'read' | 'update' | 'delete')[]
}

// Exemple de rôles
const roles = {
  admin: {
    name: 'Administrateur',
    permissions: ['*'], // Tous les droits
  },
  manager: {
    name: 'Gestionnaire',
    permissions: [
      { resource: 'products', actions: ['create', 'read', 'update'] },
      { resource: 'orders', actions: ['read', 'update'] }
    ]
  },
  support: {
    name: 'Support Client',
    permissions: [
      { resource: 'orders', actions: ['read', 'update'] },
      { resource: 'users', actions: ['read', 'update'] }
    ]
  }
}
```

### Chat et Support Client

**Localisation :** `src/components/chat/`, `src/pages/admin/AdminChatPage.tsx`

**Système de Chat Temps Réel :**

- **WebSocket** : Communication instantanée bidirectionnelle
- **File d'Attente** : Gestion automatique des demandes clients
- **Historique** : Conservation des conversations
- **Notifications** : Alertes pour nouveaux messages
- **Fichiers** : Envoi d'images et documents
- **Émoticônes** : Support emoji et réactions

**Implementation WebSocket :**

```
// Configuration Socket.io côté client
const socket = io(process.env.VITE_SOCKET_URL, {
  auth: {
    token: authToken
  }
})

// Gestion des événements chat
socket.on('message', (message: ChatMessage) => {
  setMessages(prev => [...prev, message])
})

socket.on('user_connected', (userId: string) => {
  setOnlineUsers(prev => [...prev, userId])
})

// Envoi de message
const sendMessage = (content: string, type: 'text' | 'image' = 'text') => {
  const message = {
    id: generateId(),
    content,
    type,
    timestamp: new Date(),
    senderId: user.id,
    conversationId: currentConversation.id
  }

  socket.emit('send_message', message)
}
```

### Paramètres du Site

**Localisation :** `src/pages/admin/AdminSettingsPage.tsx`

**Sections de Configuration :**

- **Général** : Nom du site, description, contact
- **E-commerce** : Devise, taxes, frais de port
- **Paiements** : Configuration des passerelles de paiement
- **Notifications** : Emails automatiques et SMS
- **SEO** : Meta données et optimisations
- **Sécurité** : Politique de mots de passe, tentatives de connexion
- **Apparence** : Thème, couleurs, logo
- **Maintenance** : Mode maintenance et sauvegardes

**Configuration E-commerce :**

```
interface EcommerceSettings {
  currency: {
    code: 'EUR' | 'USD' | 'GBP'
    symbol: string
    position: 'before' | 'after'
  }

  shipping: {
    freeShippingThreshold: number
    defaultShippingCost: number
    shippingZones: ShippingZone[]
  }

  tax: {
    enabled: boolean
    defaultRate: number
    includedInPrice: boolean
  }

  inventory: {
    trackStock: boolean
    allowBackorder: boolean
    lowStockThreshold: number
  }
}
```

---

## Fonctionnalités Backend (Serveur)

### API REST Complète

**Localisation :** `server/routes/`, `server/services/`

**Endpoints Principaux :**

#### Authentification (`/api/auth`)

```
POST   /api/auth/login          // Connexion utilisateur
POST   /api/auth/register       // Inscription utilisateur
POST   /api/auth/logout         // Déconnexion
POST   /api/auth/refresh        // Renouvellement token
POST   /api/auth/forgot         // Mot de passe oublié
POST   /api/auth/reset          // Reset mot de passe
```

#### Produits (`/api/products`)

```
GET    /api/products            // Liste des produits
GET    /api/products/:id        // Détail d'un produit
POST   /api/products            // Créer un produit (admin)
PUT    /api/products/:id        // Modifier un produit (admin)
DELETE /api/products/:id        // Supprimer un produit (admin)
GET    /api/products/search     // Rechercher des produits
GET    /api/products/featured   // Produits vedettes
```

#### Commandes (`/api/orders`)

```
GET    /api/orders              // Commandes utilisateur
POST   /api/orders              // Créer une commande
GET    /api/orders/:id          // Détail d'une commande
PUT    /api/orders/:id/status   // Modifier statut (admin)
POST   /api/orders/:id/cancel   // Annuler commande
GET    /api/orders/:id/invoice  // Télécharger facture
```

#### Panier (`/api/panier`)

```
GET    /api/panier              // Contenu du panier
POST   /api/panier/add          // Ajouter au panier
PUT    /api/panier/update       // Modifier quantité
DELETE /api/panier/remove       // Supprimer du panier
DELETE /api/panier/clear        // Vider le panier
```

**Services Backend Principaux :**

### Service d'Authentification

```
class AuthService {
  async authenticate(email, password) {
    // Vérification credentials
    // Génération JWT token
    // Gestion des tentatives de connexion
    // Logging des accès
  }

  async generateTokens(user) {
    const accessToken = jwt.sign(payload, secret, { expiresIn: '15m' })
    const refreshToken = jwt.sign(payload, refreshSecret, { expiresIn: '7d' })
    return { accessToken, refreshToken }
  }

  async validateToken(token) {
    // Vérification signature
    // Contrôle expiration
    // Validation utilisateur actif
  }
}
```

### Service des Produits

```
class ProductsService {
  async getAllProducts(filters = {}) {
    // Application des filtres
    // Tri et pagination
    // Calcul des promotions
    // Formatage des prix
  }

  async createProduct(productData, images) {
    // Validation des données
    // Upload et traitement des images
    // Génération du slug SEO
    // Indexation pour recherche
  }

  async updateStock(productId, quantity, operation = 'set') {
    // Mise à jour atomique du stock
    // Vérification des seuils
    // Notifications de réapprovisionnement
  }
}
```

### Système de Sécurité Avancé

**Localisation :** `server/middlewares/security.js`, `server/config/security.js`

**Middlewares de Sécurité :**

- **Rate Limiting** : Protection contre le spam et attaques par force brute
- **CORS** : Configuration Cross-Origin Resource Sharing
- **Helmet** : Headers de sécurité HTTP
- **XSS Protection** : Filtrage du contenu malveillant
- **CSRF Protection** : Protection contre les attaques Cross-Site Request Forgery
- **Input Validation** : Validation et sanitisation des données d'entrée

**Configuration Sécurité :**

```
// Rate limiting par endpoint
const rateLimits = {
  '/api/auth/login': { windowMs: 15 * 60 * 1000, max: 5 }, // 5 tentatives/15min
  '/api/products': { windowMs: 60 * 1000, max: 100 },       // 100 req/min
  '/api/orders': { windowMs: 60 * 1000, max: 20 }           // 20 req/min
}

// Headers sécurisés
app.use(helmet({
  contentSecurityPolicy: {
    directives: {
      defaultSrc: ["'self'"],
      styleSrc: ["'self'", "'unsafe-inline'"],
      scriptSrc: ["'self'"],
      imgSrc: ["'self'", "data:", "https:"]
    }
  },
  hsts: {
    maxAge: 31536000,
    includeSubDomains: true,
    preload: true
  }
}))
```

### Gestion des Fichiers

**Localisation :** `server/routes/profile-images.js`, `server/uploads/`

**Upload Sécurisé :**

- **Validation Types** : Contrôle des extensions et MIME types

- **Limitation Taille** : Limites par type de fichier
- **Scan Antivirus** : Vérification des fichiers uploadés
- **Optimisation Images** : Redimensionnement et compression automatiques
- **CDN Integration** : Stockage et livraison optimisés

**Configuration Multer :**

```
const upload = multer({
  storage: multer.diskStorage({
    destination: (req, file, cb) => {
      const uploadPath = `uploads/${file.fieldname}s/`
      ensureDirectoryExists(uploadPath)
      cb(null, uploadPath)
    },
    filename: (req, file, cb) => {
      const uniqueName = `${file.fieldname}-${Date.now()}-${Math.round(Math.random() * 1E9)}`
      cb(null, `${uniqueName}${path.extname(file.originalname)}`)
    }
  }),
  limits: {
    fileSize: 5 * 1024 * 1024, // 5MB max
    files: 6 // 6 fichiers max
  },
  fileFilter: (req, file, cb) => {
    const allowedTypes = ['image/jpeg', 'image/png', 'image/webp']
    if (allowedTypes.includes(file.mimetype)) {
      cb(null, true)
    } else {
      cb(new Error('Type de fichier non autorisé'), false)
    }
  }
})
```

### WebSocket et Temps Réel

**Localisation :** `server/socket/`, `server/services/socketHandlers.js`

**Fonctionnalités Temps Réel :**

- **Chat Support** : Messagerie instantanée client-admin
- **Notifications** : Alertes en temps réel
- **Mises à Jour Stock** : Synchronisation des quantités
- **Activité Utilisateurs** : Présence et actions en ligne
- **Commandes Live** : Notifications de nouvelles commandes

**Configuration Socket.io :**

```
// Configuration serveur Socket.io
const io = new Server(server, {
  cors: {
    origin: process.env.CLIENT_URL,
    credentials: true
  },
  transports: ['websocket', 'polling']
})

// Middleware d'authentification Socket
io.use(async (socket, next) => {
  try {
    const token = socket.handshake.auth.token
    const decoded = jwt.verify(token, process.env.JWT_SECRET)
    const user = await getUserById(decoded.userId)

    socket.userId = user.id
    socket.userRole = user.role
    next()
  } catch (err) {
    next(new Error('Authentication failed'))
  }
})

// Gestion des événements
io.on('connection', (socket) => {
  console.log(`Utilisateur connecté: ${socket.userId}`)

  // Rejoindre les salles appropriées
  socket.join(`user_${socket.userId}`)
  if (socket.userRole === 'admin') {
    socket.join('admins')
  }

  // Gestion du chat
  socket.on('send_message', async (messageData) => {
    const message = await saveMessage({
      ...messageData,
      senderId: socket.userId,
      timestamp: new Date()
    })

    // Diffuser le message
    io.to(message.conversationId).emit('message', message)
```

```
  })

  // Déconnexion
  socket.on('disconnect', () => {
    console.log(`Utilisateur déconnecté: ${socket.userId}`)
  })
})
```

---

## Interface Utilisateur et Design

### Système de Design Cohérent

**Localisation :** `src/components/ui/`, `src/index.css`, `tailwind.config.ts`

**Palette de Couleurs :**

```
:root {
  /* Couleurs principales */
  --primary: 0 84% 60%;          /* Rouge principal de la marque */
  --primary-foreground: 0 0% 98%;

  /* Couleurs secondaires */
  --secondary: 0 0% 96%;
  --secondary-foreground: 0 0% 9%;

  /* États et feedback */
  --destructive: 0 84% 60%;      /* Erreurs et suppression */
  --success: 142 76% 36%;        /* Actions réussies */
  --warning: 38 92% 50%;         /* Avertissements */

  /* Interface */
  --background: 0 0% 100%;
  --foreground: 0 0% 3.9%;
  --card: 0 0% 100%;
  --border: 0 0% 89.8%;
  --input: 0 0% 89.8%;
  --ring: 0 84% 60%;
}
```

**Composants UI Standardisés :**

- **Buttons** : 6 variantes (default, destructive, outline, secondary, ghost, link)
- **Forms** : Input, Textarea, Select, Checkbox, Radio avec validation
- **Navigation** : Navbar responsive, Breadcrumbs, Pagination

- **Feedback** : Alerts, Toasts, Loading Spinners, Progress bars
- **Layout** : Cards, Containers, Grids, Flexbox helpers
- **Overlays** : Modals, Drawers, Tooltips, Popovers

### Responsive Design

**Breakpoints Tailwind :**

```
// Configuration responsive
screens: {
  'sm': '640px',   // Mobile large
  'md': '768px',   // Tablet
  'lg': '1024px',  // Desktop
  'xl': '1280px',  // Large desktop
  '2xl': '1536px'  // Extra large
}

// Utilisation dans composants
<div className="
  grid grid-cols-1    // Mobile: 1 colonne
  sm:grid-cols-2      // Mobile large: 2 colonnes
  lg:grid-cols-4      // Desktop: 4 colonnes
  gap-4 sm:gap-6      // Espacement adaptatif
">
```

**Navigation Mobile Optimisée :**

```
// Menu hamburger avec drawer
const MobileNavigation = () => {
  return (
    <Sheet>
      <SheetTrigger asChild>
        <Button variant="ghost" size="icon" className="md:hidden">
          <Menu className="h-5 w-5" />
        </Button>
      </SheetTrigger>
      <SheetContent side="left" className="w-80">
        {/* Navigation mobile */}
      </SheetContent>
    </Sheet>
  )
}
```

### Accessibilité (A11y)

**Standards WCAG 2.1 AA :** - **Contraste** : Ratios de couleurs conformes (4.5:1 minimum) - **Navigation Clavier** : Tous les éléments accessibles au

clavier - **Screen Readers** : Labels appropriés et structure sémantique - **Focus Management** : Indicateurs de focus visibles - **ARIA Labels** : Descriptions pour éléments complexes

**Implementation Accessibilité :**

```
// Bouton accessible
<Button
  aria-label="Ajouter le produit au panier"
  aria-describedby="cart-description"
  disabled={product.stock === 0}
>
  <ShoppingCart className="mr-2 h-4 w-4" />
  Ajouter au panier
</Button>

// Formulaire accessible
<form role="form" aria-label="Formulaire de connexion">
  <Label htmlFor="email">Adresse email</Label>
  <Input
    id="email"
    type="email"
    aria-required="true"
    aria-invalid={errors.email ? 'true' : 'false'}
    aria-describedby={errors.email ? 'email-error' : undefined}
  />
  {errors.email && (
    <div id="email-error" role="alert" className="text-destructive">
      {errors.email.message}
    </div>
  )}
</form>
```

---

## Performance et Optimisations

### Optimisations Frontend

**Code Splitting et Lazy Loading :**

```
// Pages en lazy loading
const HomePage = lazy(() => import('../pages/HomePage'))
const ProductDetail = lazy(() => import('../pages/ProductDetail'))
const AdminDashboard = lazy(() => import('../pages/admin/AdminDashboard'))

// Composants conditionnels
const AdminPanel = lazy(() =>
```

```
  import('../components/admin/AdminPanel').then(module => ({
    default: module.AdminPanel
  }))
)

// Utilisation avec Suspense
<Suspense fallback={<PageLoadingSkeleton />}>
  <Routes>
    <Route path="/" element={<HomePage />} />
    <Route path="/products/:id" element={<ProductDetail />} />
  </Routes>
</Suspense>
```

**Optimisations React :**

```
// Mémorisation des calculs coûteux
const filteredProducts = useMemo(() => {
  return products.filter(product =>
    product.name.toLowerCase().includes(searchQuery.toLowerCase()) &&
    product.price >= priceRange[0] &&
    product.price <= priceRange[1]
  )
}, [products, searchQuery, priceRange])

// Mémorisation des callbacks
const handleAddToCart = useCallback((productId: string, quantity: number) => {
  addToCart(productId, quantity)
}, [addToCart])

// Composants mémorisés
const ProductCard = React.memo(({ product, onAddToCart }) => {
  return (
    <Card>
      {/* Contenu du composant */}
    </Card>
  )
}, (prevProps, nextProps) => {
  return prevProps.product.id === nextProps.product.id &&
         prevProps.product.stock === nextProps.product.stock
})
```

**Optimisations Images :**

```
// Lazy loading avec Intersection Observer
const LazyImage = ({ src, alt, className }) => {
  const [isInView, setIsInView] = useState(false)
```

```
  const [isLoaded, setIsLoaded] = useState(false)
  const imgRef = useRef(null)

  useEffect(() => {
    const observer = new IntersectionObserver(
      ([entry]) => {
        if (entry.isIntersecting) {
          setIsInView(true)
          observer.disconnect()
        }
      },
      { threshold: 0.1 }
    )

    if (imgRef.current) {
      observer.observe(imgRef.current)
    }

    return () => observer.disconnect()
  }, [])

  return (
    <div ref={imgRef} className={className}>
      {isInView && (
        <img
          src={src}
          alt={alt}
          loading="lazy"
          onLoad={() => setIsLoaded(true)}
          className={cn(
            "transition-opacity duration-300",
            isLoaded ? "opacity-100" : "opacity-0"
          )}
        />
      )}
    </div>
  )
}
```

### Optimisations Backend

**Cache et Performance Base de Données :**

```
// Cache en mémoire pour données fréquentes
const cache = new Map()
const CACHE_TTL = 5 * 60 * 1000 // 5 minutes
```

```
const getCachedData = (key) => {
  const cached = cache.get(key)
  if (cached && Date.now() - cached.timestamp < CACHE_TTL) {
    return cached.data
  }
  cache.delete(key)
  return null
}

const setCachedData = (key, data) => {
  cache.set(key, {
    data,
    timestamp: Date.now()
  })
}

// Service avec cache
class ProductsService {
  async getFeaturedProducts() {
    const cacheKey = 'featured_products'
    let products = getCachedData(cacheKey)

    if (!products) {
      products = await this.loadFeaturedProducts()
      setCachedData(cacheKey, products)
    }

    return products
  }
}
```

**Compression et Optimisation Réponses :**

```
// Compression gzip
app.use(compression({
  level: 6,
  threshold: 1024,
  filter: (req, res) => {
    if (req.headers['x-no-compression']) {
      return false
    }
    return compression.filter(req, res)
  }
}))
```

```
// Headers de cache optimisés
app.use('/static', express.static('public', {
  maxAge: '1y',
  etag: true,
  lastModified: true,
  cacheControl: true
}))

// Réponses JSON optimisées
app.use((req, res, next) => {
  const originalSend = res.send
  res.send = function(data) {
    if (typeof data === 'object') {
      // Minification JSON
      data = JSON.stringify(data)
    }
    originalSend.call(this, data)
  }
  next()
})
```

---

## Tests et Qualité

### Tests Frontend

**Tests Unitaires Composants :**

```
// ProductCard.test.tsx
import { render, screen, fireEvent } from '@testing-library/react'
import { ProductCard } from '../ProductCard'

describe('ProductCard', () => {
  const mockProduct = {
    id: '1',
    nom: 'Test Product',
    prix: 29.99,
    stock: 10,
    images: ['test-image.jpg']
  }

  test('affiche les informations du produit correctement', () => {
    render(<ProductCard product={mockProduct} />)

    expect(screen.getByText('Test Product')).toBeInTheDocument()
    expect(screen.getByText('29.99€')).toBeInTheDocument()
```

```
    expect(screen.getByAltText('Test Product')).toBeInTheDocument()
  })

  test('ajoute au panier quand on clique sur le bouton', async () => {
    const mockAddToCart = jest.fn()
    render(<ProductCard product={mockProduct} onAddToCart={mockAddToCart} />)

    const addButton = screen.getByText('Ajouter au panier')
    fireEvent.click(addButton)

    expect(mockAddToCart).toHaveBeenCalledWith(mockProduct.id, 1)
  })

  test('affiche "Rupture" quand stock est à 0', () => {
    const outOfStockProduct = { ...mockProduct, stock: 0 }
    render(<ProductCard product={outOfStockProduct} />)

    expect(screen.getByText('Rupture')).toBeInTheDocument()
    expect(screen.getByText('Indisponible')).toBeInTheDocument()
  })
})
```

**Tests d'Intégration :**

```
// CartIntegration.test.tsx
describe('Cart Integration', () => {
  test('workflow complet ajout au panier', async () => {
    render(<App />)

    // Navigation vers produits
    fireEvent.click(screen.getByText('Produits'))
    await waitFor(() => screen.getByTestId('products-grid'))

    // Sélection d'un produit
    const productCard = screen.getByTestId('product-1')
    fireEvent.click(within(productCard).getByText('Ajouter au panier'))

    // Vérification ajout
    await waitFor(() => {
      expect(screen.getByTestId('cart-count')).toHaveTextContent('1')
    })

    // Ouverture du panier
    fireEvent.click(screen.getByTestId('cart-button'))

    // Vérification contenu
```

```
    await waitFor(() => {
      expect(screen.getByTestId('cart-drawer')).toBeInTheDocument()
      expect(screen.getByText('Test Product')).toBeInTheDocument()
    })
  })
})
```

### Tests Backend

**Tests API :**

```
// products.test.js
const request = require('supertest')
const app = require('../server')

describe('Products API', () => {
  describe('GET /api/products', () => {
    test('retourne la liste des produits', async () => {
      const response = await request(app)
        .get('/api/products')
        .expect(200)

      expect(Array.isArray(response.body)).toBe(true)
      expect(response.body.length).toBeGreaterThan(0)
    })

    test('filtre par catégorie', async () => {
      const response = await request(app)
        .get('/api/products?category=electronique')
        .expect(200)

      response.body.forEach(product => {
        expect(product.categories).toContain('electronique')
      })
    })
  })

  describe('POST /api/products', () => {
    test('crée un nouveau produit avec authentification admin', async () => {
      const adminToken = await getAdminToken()
      const productData = {
        nom: 'Nouveau Produit',
        description: 'Description test',
        prix: 49.99,
        stock: 100
      }
```

```
      const response = await request(app)
        .post('/api/products')
        .set('Authorization', `Bearer ${adminToken}`)
        .send(productData)
        .expect(201)

      expect(response.body.nom).toBe(productData.nom)
      expect(response.body.id).toBeDefined()
    })

    test('refuse la création sans authentification', async () => {
      const productData = { nom: 'Test' }

      await request(app)
        .post('/api/products')
        .send(productData)
        .expect(401)
    })
  })
})
```

### Couverture de Tests

```
# Commandes de test
npm run test              # Tests unitaires
npm run test:watch        # Mode watch
npm run test:coverage     # Avec couverture
npm run test:e2e          # Tests end-to-end

# Configuration Jest (jest.config.js)
module.exports = {
  testEnvironment: 'jsdom',
  setupFilesAfterEnv: ['<rootDir>/src/setupTests.ts'],
  collectCoverageFrom: [
    'src/**/*.{ts,tsx}',
    '!src/**/*.d.ts',
    '!src/main.tsx',
    '!src/vite-env.d.ts'
  ],
  coverageThreshold: {
    global: {
      branches: 70,
      functions: 70,
      lines: 70,
      statements: 70
```

```
    }
  }
}
```

---

## Déploiement et Production

### Build et Déploiement

**Configuration de Build :**

```
// vite.config.ts
export default defineConfig({
  plugins: [react()],
  build: {
    outDir: 'dist',
    minify: 'terser',
    sourcemap: process.env.NODE_ENV !== 'production',
    rollupOptions: {
      output: {
        manualChunks: {
          vendor: ['react', 'react-dom'],
          router: ['react-router-dom'],
          ui: ['@radix-ui/react-dialog', '@radix-ui/react-dropdown-menu']
        }
      }
    },
    chunkSizeWarningLimit: 1000
  },
  server: {
    proxy: {
      '/api': 'http://localhost:10000'
    }
  }
})
```

**Variables d'Environnement Production :**

```
# .env.production
NODE_ENV=production
VITE_API_BASE_URL=https://api.riziky-boutic.com
VITE_SOCKET_URL=https://api.riziky-boutic.com
VITE_STRIPE_PUBLIC_KEY=pk_live_...
VITE_PAYPAL_CLIENT_ID=...

# Backend (.env serveur)
NODE_ENV=production
```

```
JWT_SECRET=your-super-secret-jwt-key-256-bits
JWT_REFRESH_SECRET=your-refresh-secret-key
DATABASE_URL=postgresql://user:pass@localhost:5432/riziky_boutic
REDIS_URL=redis://localhost:6379
SMTP_HOST=smtp.sendgrid.net
SMTP_USER=apikey
SMTP_PASS=your-sendgrid-api-key
```

**Scripts de Déploiement :**

```
#!/bin/bash
# deploy.sh

echo "  Début du déploiement..."

# Build frontend
echo "  Build du frontend..."
npm run build

# Tests avant déploiement
echo "  Exécution des tests..."
npm run test:prod

# Upload vers serveur
echo "  Upload des fichiers..."
rsync -avz --delete dist/ user@server:/var/www/riziky-boutic/

# Redémarrage des services
echo "  Redémarrage des services..."
ssh user@server "sudo systemctl restart nginx && sudo systemctl restart riziky-boutic"

echo "  Déploiement terminé!"
```

### Configuration Serveur Production

**Nginx Configuration :**

```
# /etc/nginx/sites-available/riziky-boutic
server {
    listen 80;
    listen [::]:80;
    server_name riziky-boutic.com www.riziky-boutic.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
```

```
    listen [::]:443 ssl http2;
    server_name riziky-boutic.com www.riziky-boutic.com;

    ssl_certificate /etc/letsencrypt/live/riziky-boutic.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/riziky-boutic.com/privkey.pem;

    # Frontend statique
    location / {
        root /var/www/riziky-boutic;
        try_files $uri $uri/ /index.html;

        # Cache statique
        location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2)$ {
            expires 1y;
            add_header Cache-Control "public, immutable";
        }
    }

    # API Backend
    location /api/ {
        proxy_pass http://localhost:10000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }

    # WebSocket
    location /socket.io/ {
        proxy_pass http://localhost:10000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

**Service Systemd :**

```
# /etc/systemd/system/riziky-boutic.service
[Unit]
Description=Riziky-Boutic API Server
After=network.target
```

```
[Service]
Type=simple
User=www-data
WorkingDirectory=/opt/riziky-boutic
ExecStart=/usr/bin/node server/server.js
Restart=on-failure
RestartSec=10
Environment=NODE_ENV=production
EnvironmentFile=/opt/riziky-boutic/.env

# Sécurité
NoNewPrivileges=yes
ProtectSystem=strict
ProtectHome=yes
ReadWritePaths=/opt/riziky-boutic/server/uploads
ReadWritePaths=/opt/riziky-boutic/server/data

[Install]
WantedBy=multi-user.target
```

---

## Maintenance et Monitoring

### Monitoring et Logs

**Logging Structuré :**

```
// logger.js
const winston = require('winston')

const logger = winston.createLogger({
  level: process.env.LOG_LEVEL || 'info',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.errors({ stack: true }),
    winston.format.json()
  ),
  defaultMeta: { service: 'riziky-boutic' },
  transports: [
    new winston.transports.File({ filename: 'logs/error.log', level: 'error' }),
    new winston.transports.File({ filename: 'logs/combined.log' })
  ]
})

if (process.env.NODE_ENV !== 'production') {
```

```
  logger.add(new winston.transports.Console({
    format: winston.format.simple()
  }))
}

// Utilisation dans l'application
logger.info('Utilisateur connecté', {
  userId: user.id,
  email: user.email,
  ip: req.ip
})

logger.error('Erreur lors de la création de produit', {
  error: error.message,
  stack: error.stack,
  productData: productData
})
```

**Health Checks :**

```
// routes/health.js
router.get('/health', async (req, res) => {
  const checks = {
    timestamp: new Date().toISOString(),
    status: 'ok',
    checks: {
      database: await checkDatabase(),
      redis: await checkRedis(),
      disk: await checkDiskSpace(),
      memory: await checkMemoryUsage()
    }
  }

  const hasErrors = Object.values(checks.checks).some(check => !check.status)

  res.status(hasErrors ? 503 : 200).json(checks)
})

const checkDatabase = async () => {
  try {
    const result = await db.query('SELECT 1')
    return { status: true, latency: Date.now() - start }
  } catch (error) {
    return { status: false, error: error.message }
  }
}
```

### Sauvegarde et Restauration

**Scripts de Sauvegarde :**

```
#!/bin/bash
# backup.sh

BACKUP_DIR="/var/backups/riziky-boutic"
DATE=$(date +%Y%m%d_%H%M%S)

# Sauvegarde données JSON
echo "  Sauvegarde des données..."
tar -czf "$BACKUP_DIR/data_$DATE.tar.gz" server/data/

# Sauvegarde uploads
echo "  Sauvegarde des fichiers uploadés..."
tar -czf "$BACKUP_DIR/uploads_$DATE.tar.gz" server/uploads/

# Sauvegarde base de données (si PostgreSQL)
if [ "$USE_DATABASE" == "true" ]; then
  echo "  Sauvegarde de la base de données..."
  pg_dump $DATABASE_URL | gzip > "$BACKUP_DIR/db_$DATE.sql.gz"
fi

# Nettoyage des anciennes sauvegardes (> 30 jours)
find $BACKUP_DIR -name "*.tar.gz" -mtime +30 -delete
find $BACKUP_DIR -name "*.sql.gz" -mtime +30 -delete

echo "  Sauvegarde terminée: $DATE"
```

**Restauration :**

```
#!/bin/bash
# restore.sh

BACKUP_FILE=$1
RESTORE_DIR="/opt/riziky-boutic"

if [ -z "$BACKUP_FILE" ]; then
  echo "  Usage: $0 <backup_file>"
  exit 1
fi

echo "  Arrêt des services..."
sudo systemctl stop riziky-boutic

echo "  Restauration des données..."
```

```
tar -xzf "$BACKUP_FILE" -C "$RESTORE_DIR"

echo "  Restauration des permissions..."
chown -R www-data:www-data "$RESTORE_DIR/server/data"
chown -R www-data:www-data "$RESTORE_DIR/server/uploads"

echo "  Redémarrage des services..."
sudo systemctl start riziky-boutic

echo "  Restauration terminée!"
```

---

## Documentation de Référence

### APIs et Intégrations

**Documentation API Complete :**

```yaml
# openapi.yml
openapi: 3.0.0
info:
  title: Riziky-Boutic API
  version: 1.0.0
  description: API complète pour la plateforme e-commerce

paths:
  /api/products:
    get:
      summary: Liste des produits
      parameters:
        - name: category
          in: query
          schema:
            type: string
        - name: search
          in: query
          schema:
            type: string
        - name: limit
          in: query
          schema:
            type: integer
            default: 20
        - name: offset
          in: query
          schema:
```

```yaml
            type: integer
            default: 0
      responses:
        200:
          description: Liste des produits
          content:
            application/json:
              schema:
                type: object
                properties:
                  products:
                    type: array
                    items:
                      $ref: '#/components/schemas/Product'
                  total:
                    type: integer
                  hasMore:
                    type: boolean

components:
  schemas:
    Product:
      type: object
      properties:
        id:
          type: string
        nom:
          type: string
        description:
          type: string
        prix:
          type: number
        stock:
          type: integer
        categories:
          type: array
          items:
            type: string
        images:
          type: array
          items:
            type: string
```

### Guides d'Utilisation Avancée

Ce résumé complet des fonctionnalités de Riziky-Boutic couvre tous les aspects techniques et fonctionnels de la plateforme. Il sert de référence pour comprendre, utiliser et maintenir l'ensemble du système e-commerce.

**Points Clés :** - Interface client complète et moderne - Administration avancée avec tous les outils nécessaires
- Backend robuste avec API REST et WebSocket - Sécurité de niveau production - Performance et optimisations - Tests et qualité de code - Déploiement et monitoring - Documentation complète pour développeurs

Cette plateforme est prête pour la production et peut évoluer selon les besoins business futurs.