```tsx
import React, { useState } from 'react';
import { useParams, useNavigate } from 'react-router-dom';
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Badge } from '@/components/ui/badge';
import { Button } from '@/components/ui/button';
import {
  User,
  Mail,
  Phone,
  Calendar,
  MapPin,
  FileText,
  ArrowLeft,
  CreditCard,
  Download,
  Send,
  RefreshCw,
  AlertCircle
} from 'lucide-react';
import { toast } from '@/hooks/use-toast';
import { candidatureService } from '@/services/candidatureService';
import { receiptService } from '@/services/receiptService';
import CandidateDocumentManager from '@/components/admin/CandidateDocumentManager';
import CandidatePhotoCard from '@/components/admin/CandidatePhotoCard';
import { useAdminState } from '@/hooks/useAdminState';
import { useAdminAuth } from '@/contexts/AdminAuthContext'; // Ajouté pour accéder au rôle
import { apiService } from '@/services/api';

const CandidateManagement = () => {
  const { nupcan } = useParams();
  const navigate = useNavigate();
  const [activeTab, setActiveTab] = useState('overview');
  const queryClient = useQueryClient();
  const { admin } = useAdminAuth(); // Récupère les informations de l'admin connecté
  const { isLoading: actionLoading, executeAction } = useAdminState();

  const { data: candidatureData, isLoading, error, refetch } = useQuery({
    queryKey: ['candidature', nupcan],
    queryFn: () => candidatureService.getCandidatureByNupcan(nupcan!),
    enabled: !!nupcan,
    retry: 3,
    retryDelay: 1000,
  });

  const validateDocumentMutation = useMutation({
    mutationFn: ({ documentId, statut, commentaire }: { documentId: number; statut: 'valide' | 'rejete'; commentaire?: string }) =>
        apiService.validateDocument(documentId.toString(), statut, commentaire),
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['admin-dossiers'] });
      toast({
        title: "Document validé",
        description: "Le candidat a été automatiquement notifié",
      });
    },
    onError: (error) => {
      console.error('Erreur validation:', error);
      toast({
        title: "Erreur",
        description: "Impossible de valider le document",
        variant: "destructive",
      });
    },
  });

  const handleDownloadReceipt = async () => {
    if (!candidatureData) return;

    await executeAction(
        async () => {
          const receiptData = {
            candidat: {
              ...candidatureData.candidat,
              ldncan: candidatureData.candidat.ldncan || 'Libreville',
              phtcan: typeof candidatureData.candidat.phtcan === 'string'
                  ? candidatureData.candidat.phtcan
                  : undefined
            },
            concours: {
              ...candidatureData.concours,
              fracnc: candidatureData.concours.fracnc || 0,
              sescnc: candidatureData.concours.sescnc || ''
            },
            filiere: candidatureData.filiere || undefined,
            paiement: candidatureData.paiement,
            documents: candidatureData.documents || []
          };
          await receiptService.downloadReceiptPDF(receiptData);
        },
        'Reçu téléchargé avec succès'
    );
  };

  const handleSendReceiptEmail = async () => {
    if (!candidatureData) return;

    await executeAction(
        async () => {
          const receiptData = {
            candidat: {
              ...candidatureData.candidat,
              ldncan: candidatureData.candidat.ldncan || 'Libreville',
              phtcan: typeof candidatureData.candidat.phtcan === 'string'
                  ? candidatureData.candidat.phtcan
                  : undefined
            },
            concours: {
              ...candidatureData.concours,
              fracnc: candidatureData.concours.fracnc || 0,
              sescnc: candidatureData.concours.sescnc || ''
            },
            filiere: candidatureData.filiere || undefined,
            paiement: candidatureData.paiement,
            documents: candidatureData.documents || []
          };
          await receiptService.generateAndSendReceiptEmail(receiptData, candidatureData.candidat.maican);
        },
        'Reçu envoyé par email avec succès'
    );
  };

  const handleRefresh = () => {
    refetch();
    toast({
      title: "Actualisation",
      description: "Données actualisées avec succès",
    });
  };

  if (isLoading) {
    return (
        <div className="flex justify-center items-center h-64">
          <div className="text-center">
            <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary mx-auto mb-4"></div>
            <p>Chargement des données de candidature...</p>
          </div>
        </div>
    );
  }

  if (error || !candidatureData) {
    return (
        <div className="text-center py-12">
          <AlertCircle className="h-16 w-16 text-red-500 mx-auto mb-4" />
          <h2 className="text-xl font-semibold mb-2 text-red-600">
            {error ? 'Erreur de chargement' : 'Candidature non trouvée'}
          </h2>
          <p className="text-muted-foreground mb-4">
            {error ? 'Impossible de charger les données' : 'Cette candidature n\'existe pas ou n\'est plus accessible.'}
          </p>
          <div className="flex justify-center space-x-4">
            <Button variant="outline" onClick={() => {
              const returnPath = admin?.role === 'admin_etablissement' ? '/admin/dashboard' : '/admin/candidats';
              navigate(returnPath);
            }}>
              <ArrowLeft className="h-4 w-4 mr-2" />
              Retour à la liste
            </Button>
            <Button onClick={handleRefresh}>
              <RefreshCw className="h-4 w-4 mr-2" />
              Réessayer
            </Button>
          </div>
        </div>
    );
  }

  const candidat = candidatureData.candidat;
  const concours = candidatureData.concours;
  const filiere = candidatureData.filiere;
  const paiement = candidatureData.paiement;

  return (
      <div className="space-y-6">
        {/* En-tête avec navigation */}
        <div className="flex items-center justify-between">
          <div className="flex items-center space-x-4">
            <Button variant="ghost" onClick={() => {
              const returnPath = admin?.role === 'admin_etablissement' ? '/admin/dashboard' : '/admin/candidats';
              navigate(returnPath);
            }}>
              <ArrowLeft className="h-4 w-4 mr-2" />
              Retour à la liste
            </Button>
            <div>
              <h1 className="text-2xl font-bold">Gestion de la Candidature</h1>
              <p className="text-muted-foreground">NUPCAN: {nupcan}</p>
            </div>
          </div>
          <div className="flex items-center space-x-2">
```

```
            <Button variant="outline" size="sm" onClick={handleRefresh}>
              <RefreshCw className="h-4 w-4 mr-2" />
              Actualiser
            </Button>
          </div>
        </div>

        {/* Onglets de navigation */}
        <div className="border-b border-gray-200">
          <nav className="-mb-px flex space-x-8">
            <button
                onClick={() => setActiveTab('overview')}
                className={`py-2 px-1 border-b-2 font-medium text-sm ${
                    activeTab === 'overview'
                        ? 'border-primary text-primary'
                        : 'border-transparent text-gray-500 hover:text-gray-700 hover:border-gray-300'
                }`}
            >
              Vue d'ensemble
            </button>
            <button
                onClick={() => setActiveTab('documents')}
                className={`py-2 px-1 border-b-2 font-medium text-sm ${
                    activeTab === 'documents'
                        ? 'border-primary text-primary'
                        : 'border-transparent text-gray-500 hover:text-gray-700 hover:border-gray-300'
                }`}
            >
              Documents
            </button>
          </nav>
        </div>

        {/* Contenu selon l'onglet actif */}
        {activeTab === 'overview' && (
            <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
              {/* Photo du candidat */}
              <CandidatePhotoCard candidat={{
                nomcan: candidat.nomcan,
                prncan: candidat.prncan,
                phtcan: typeof candidat.phtcan === 'string' ? candidat.phtcan : ''
              }} />

              {/* Informations du candidat */}
              <Card>
                <CardHeader>
                  <CardTitle className="flex items-center space-x-2">
                    <User className="h-5 w-5" />
                    <span>Informations Personnelles</span>
                  </CardTitle>
                </CardHeader>
                <CardContent className="space-y-4">
                  <div className="grid grid-cols-2 gap-4">
                    <div>
                      <label className="text-sm font-medium text-muted-foreground">Nom</label>
                      <p className="font-medium">{candidat.nomcan}</p>
                    </div>
                    <div>
                      <label className="text-sm font-medium text-muted-foreground">Prénom</label>
                      <p className="font-medium">{candidat.prncan}</p>
                    </div>
                  </div>
                  <div className="flex items-center space-x-2">
```

```
                    <Mail className="h-4 w-4 text-muted-foreground" />
                    <span>{candidat.maican}</span>
                  </div>
                  <div className="flex items-center space-x-2">
                    <Phone className="h-4 w-4 text-muted-foreground" />
                    <span>{candidat.telcan}</span>
                  </div>
                  <div className="flex items-center space-x-2">
                    <Calendar className="h-4 w-4 text-muted-foreground" />
                    <span>{new Date(candidat.dtncan).toLocaleDateString('fr-FR')}</span>
                  </div>
                  {candidat.ldncan && (
                      <div className="flex items-center space-x-2">
                        <MapPin className="h-4 w-4 text-muted-foreground" />
                        <span>{candidat.ldncan}</span>
                      </div>
                  )}
                </CardContent>
              </Card>

              {/* Informations candidature */}
              <Card>
                <CardHeader>
                  <CardTitle className="flex items-center space-x-2">
                    <FileText className="h-5 w-5" />
                    <span>Informations Candidature</span>
                  </CardTitle>
                </CardHeader>
                <CardContent className="space-y-4">
                  <div>
```

```
                    <label className="text-sm font-medium text-muted-foreground">Concours</label>
                    <p className="font-medium">{concours?.libcnc || 'Non défini'}</p>
                  </div>
                  <div>
                    <label className="text-sm font-medium text-muted-foreground">Filière</label>
                    <p className="font-medium">{filiere?.nomfil || 'Non définie'}</p>
                  </div>
                  <div>
                    <label className="text-sm font-medium text-muted-foreground">Statut</label>
                    <div className="mt-1">
                      <Badge
                          className={
                            candidat.statut === 'valide' ? 'bg-green-100 text-green-800' :
                                candidat.statut === 'rejete' ? 'bg-red-100 text-red-800' :
                                    'bg-orange-100 text-orange-800'
                          }
                      >
                        {candidat.statut || 'En attente'}
                      </Badge>
                    </div>
                  </div>
                  <div>
                    <label className="text-sm font-medium text-muted-foreground">Date de candidature</label>
                    <p>{candidat.created_at ? new Date(candidat.created_at).toLocaleDateString('fr-FR') : 'Non définie'}</p>
                  </div>
                  <div>
                    <label className="text-sm font-medium text-muted-foreground">Frais de concours</
```

```
label>
                    <p className="font-medium">
                      {!concours?.fracnc || concours.fracnc === 0 || concours.fracnc === '0'
                          ? 'GRATUIT (Programme NGORI)'
                          : `${concours.fracnc} FCFA`}
                    </p>
                  </div>
                </CardContent>
              </Card>
            </div>
        )}

        {/* Informations de paiement */}
        {activeTab === 'overview' && paiement && (
            <Card>
              <CardHeader>
                <CardTitle className="flex items-center space-x-2">
                  <CreditCard className="h-5 w-5" />
                  <span>Informations de Paiement</span>
                </CardTitle>
              </CardHeader>
              <CardContent>
                <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
                  <div>
                    <label className="text-sm font-medium text-muted-foreground">Montant</label>
                    <p className="font-medium">{paiement.montant} FCFA</p>
                  </div>
                  <div>
                    <label className="text-sm font-medium text-muted-foreground">Méthode</label>
                    <p className="font-medium">{paiement.methode}</p>
                  </div>
                  <div>
                    <label className="text-sm font-medium text-muted-foreground">Référence</label>
                    <p className="font-medium">{paiement.reference_paiement}</p>
                  </div>
                  <div>
                    <label className="text-sm font-medium text-muted-foreground">Statut</label>
                    <Badge
                        className={
                          paiement.statut === 'valide' ? 'bg-green-100 text-green-800' :
                              paiement.statut === 'rejete' ? 'bg-red-100 text-red-800' :
                                  'bg-orange-100 text-orange-800'
                        }
                    >
                      {paiement.statut}
                    </Badge>
                  </div>
                </div>
              </CardContent>
            </Card>
        )}

        {/* Actions administratives */}
        {activeTab === 'overview' && (
            <Card>
              <CardHeader>
                <CardTitle>Actions Administratives</CardTitle>
              </CardHeader>
              <CardContent>
                <div className="flex flex-wrap gap-2">
                  <Button
                      onClick={handleDownloadReceipt}
                      variant="outline"
                      disabled={actionLoading}
                  >
                    <Download className="h-4 w-4 mr-2" />
                    Télécharger le reçu
                  </Button>
                  <Button
                      onClick={handleSendReceiptEmail}
                      variant="outline"
                      disabled={actionLoading}
                  >
                    <Send className="h-4 w-4 mr-2" />
                    Envoyer le reçu par email
                  </Button>
                </div>
              </CardContent>
            </Card>
        )}

        {/* Gestion des documents */}
        {activeTab === 'documents' && (
            <CandidateDocumentManager
                candidatNupcan={nupcan!}
                candidatInfo={{
                  nom: candidat.nomcan,
                  prenom: candidat.prncan,
                  email: candidat.maican
                }}
                onDocumentValidated={() => refetch()}
            />
        )}
      </div>
  );
};

export default CandidateManagement;
```