Resumen Módulos Front.

Admin/users

"use client"

import { useState } from "react"

import { z } from "zod"

import { useForm } from "react-hook-form"

import { zodResolver } from "@hookform/resolvers/zod"

import { Button } from "@/components/ui/button"

import { Form, FormControl, FormDescription, FormField, FormItem, FormLabel, FormMessage } from "@/components/ui/form"

import { Input } from "@/components/ui/input"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

import {

  Dialog,

  DialogContent,

  DialogDescription,

  DialogFooter,

  DialogHeader,

  DialogTitle,

} from "@/components/ui/dialog"

import { DropdownMenu, DropdownMenuContent, DropdownMenuItem, DropdownMenuTrigger } from "@/components/ui/dropdown-menu"

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select"

import { Badge } from "@/components/ui/badge"

import { Search, UserPlus, MoreVertical, Edit, Trash } from "lucide-react"

import { useAuth } from "@/app/context/auth-context"

import { Alert, AlertDescription } from "@/components/ui/alert"

// Datos de ejemplo para usuarios

const initialUsers = [

  {

    id: "1",

    name: "Usuario Administrador",

    email: "admin@medcol.com",

    role: "admin",

    position: "Administrador del Sistema",

    status: "active",

  },

  {

    id: "2",

    name: "Usuario Regular",

    email: "user@medcol.com",

    role: "user",

    position: "Personal Médico",

    status: "active",

  },

  {

    id: "3",

    name: "Juan Pérez",

    email: "juan@medcol.com",

    role: "user",

    position: "Doctor",

    status: "active",

  },

  {

    id: "4",

    name: "María García",

    email: "maria@medcol.com",

    role: "user",

    position: "Enfermera",

    status: "inactive",

  },

  {

    id: "5",

    name: "Usuario Externo",

    email: "externo@medcol.com",

    role: "externo",

    position: "Consultor Externo",

    status: "active",

  },

]

const userFormSchema = z.object({

  name: z.string().min(2, { message: "El nombre debe tener al menos 2 caracteres" }),

  email: z.string().email({ message: "Por favor, introduce una dirección de correo electrónico válida" }),

  password: z.string().min(6, { message: "La contraseña debe tener al menos 6 caracteres" }),

  role: z.enum(["admin", "user", "externo"], {

    required\_error: "Por favor, selecciona un rol",

  }),

  position: z.string().optional(),

  status: z.enum(["active", "inactive"], {

    required\_error: "Por favor, selecciona un estado",

  }),

})

export default function AdminUsersPage() {

  const { user } = useAuth()

  const [users, setUsers] = useState(initialUsers)

  const [searchTerm, setSearchTerm] = useState("")

  const [isAddDialogOpen, setIsAddDialogOpen] = useState(false)

  const [isEditDialogOpen, setIsEditDialogOpen] = useState(false)

  const [isDeleteDialogOpen, setIsDeleteDialogOpen] = useState(false)

  const [currentUser, setCurrentUser] = useState<(typeof initialUsers)[0] | null>(null)

  const [isSubmitting, setIsSubmitting] = useState(false)

  const [error, setError] = useState<string | null>(null)

  const form = useForm<z.infer<typeof userFormSchema>>({

    resolver: zodResolver(userFormSchema),

    defaultValues: {

      name: "",

      email: "",

      password: "",

      role: "user",

      position: "",

      status: "active",

    },

  })

  const filteredUsers = users.filter(

    (user) =>

      user.name.toLowerCase().includes(searchTerm.toLowerCase()) ||

      user.email.toLowerCase().includes(searchTerm.toLowerCase()) ||

      user.position?.toLowerCase().includes(searchTerm.toLowerCase()),

  )

  function handleAddUser() {

    setCurrentUser(null)

    form.reset({

      name: "",

      email: "",

      password: "",

      role: "user",

      position: "",

      status: "active",

    })

    setIsAddDialogOpen(true)

  }

  function handleEditUser(user: (typeof initialUsers)[0]) {

    setCurrentUser(user)

    form.reset({

      name: user.name,

      email: user.email,

      password: "", // No rellenar la contraseña por seguridad

      role: user.role as "admin" | "user" | "externo",

      position: user.position || "",

      status: user.status as "active" | "inactive",

    })

    setIsEditDialogOpen(true)

  }

  function handleDeleteUser(user: (typeof initialUsers)[0]) {

    setCurrentUser(user)

    setIsDeleteDialogOpen(true)

  }

  async function onSubmit(values: z.infer<typeof userFormSchema>) {

    setIsSubmitting(true)

    setError(null)

    try {

      // Simular llamada a API

      await new Promise((resolve) => setTimeout(resolve, 1000))

      if (currentUser) {

        // Editar usuario existente

        setUsers(users.map((u) => (u.id === currentUser.id ? { ...u, ...values, id: u.id } : u)))

        setIsEditDialogOpen(false)

      } else {

        // Añadir nuevo usuario

        const newUser = {

          id: String(users.length + 1),

          ...values,

        }

        setUsers([...users, newUser])

        setIsAddDialogOpen(false)

      }

    } catch (error) {

      console.error(error)

      setError("Ocurrió un error. Por favor, inténtalo de nuevo.")

    } finally {

      setIsSubmitting(false)

    }

  }

  async function confirmDelete() {

    if (!currentUser) return

    setIsSubmitting(true)

    try {

      // Simular llamada a API

      await new Promise((resolve) => setTimeout(resolve, 1000))

      setUsers(users.filter((u) => u.id !== currentUser.id))

      setIsDeleteDialogOpen(false)

    } catch (error) {

      console.error(error)

      setError("Ocurrió un error. Por favor, inténtalo de nuevo.")

    } finally {

      setIsSubmitting(false)

    }

  }

  // Verificar si el usuario es administrador

  if (user?.role !== "admin") {

    return (

      <div className="flex h-full items-center justify-center">

        <div className="text-center">

          <h1 className="text-2xl font-bold">Acceso Denegado</h1>

          <p className="text-muted-foreground">No tienes permiso para acceder a esta página.</p>

        </div>

      </div>

    )

  }

  return (

    <div className="space-y-6">

      <div>

        <h1 className="text-3xl font-bold tracking-tight">Gestión de Usuarios</h1>

        <p className="text-muted-foreground">Administra cuentas de usuario y permisos</p>

      </div>

      <Card>

        <CardHeader>

          <CardTitle>Usuarios</CardTitle>

          <CardDescription>Ver y gestionar todos los usuarios en el sistema</CardDescription>

        </CardHeader>

        <CardContent>

          <div className="flex items-center justify-between mb-6">

            <div className="relative w-full max-w-sm">

              <Search className="absolute left-2.5 top-2.5 h-4 w-4 text-muted-foreground" />

              <Input

                type="search"

                placeholder="Buscar usuarios..."

                className="pl-8"

                value={searchTerm}

                onChange={(e) => setSearchTerm(e.target.value)}

              />

            </div>

            <Button onClick={handleAddUser}>

              <UserPlus className="mr-2 h-4 w-4" />

              Añadir Usuario

            </Button>

          </div>

          <div className="rounded-md border">

            <Table>

              <TableHeader>

                <TableRow>

                  <TableHead>Nombre</TableHead>

                  <TableHead>Correo electrónico</TableHead>

                  <TableHead>Cargo</TableHead>

                  <TableHead>Rol</TableHead>

                  <TableHead>Estado</TableHead>

                  <TableHead className="text-right">Acciones</TableHead>

                </TableRow>

              </TableHeader>

              <TableBody>

                {filteredUsers.length > 0 ? (

                  filteredUsers.map((user) => (

                    <TableRow key={user.id}>

                      <TableCell className="font-medium">{user.name}</TableCell>

                      <TableCell>{user.email}</TableCell>

                      <TableCell>{user.position}</TableCell>

                      <TableCell>

                        <Badge

                          variant={

                            user.role === "admin" ? "default" : user.role === "externo" ? "secondary" : "outline"

                          }

                        >

                          {user.role === "admin" ? "Administrador" : user.role === "externo" ? "Externo" : "Usuario"}

                        </Badge>

                      </TableCell>

                      <TableCell>

                        <Badge

                          variant={user.status === "active" ? "default" : "secondary"}

                          className={user.status === "active" ? "bg-green-500" : ""}

                        >

                          {user.status === "active" ? "Activo" : "Inactivo"}

                        </Badge>

                      </TableCell>

                      <TableCell className="text-right">

                        <DropdownMenu>

                          <DropdownMenuTrigger asChild>

                            <Button variant="ghost" size="icon">

                              <MoreVertical className="h-4 w-4" />

                              <span className="sr-only">Acciones</span>

                            </Button>

                          </DropdownMenuTrigger>

                          <DropdownMenuContent align="end">

                            <DropdownMenuItem onClick={() => handleEditUser(user)}>

                              <Edit className="mr-2 h-4 w-4" />

                              Editar

                            </DropdownMenuItem>

                            <DropdownMenuItem

                              onClick={() => handleDeleteUser(user)}

                              className="text-red-600"

                              disabled={user.id === "1"} // Prevenir eliminar al administrador principal

                            >

                              <Trash className="mr-2 h-4 w-4" />

                              Eliminar

                            </DropdownMenuItem>

                          </DropdownMenuContent>

                        </DropdownMenu>

                      </TableCell>

                    </TableRow>

                  ))

                ) : (

                  <TableRow>

                    <TableCell colSpan={6} className="h-24 text-center">

                      No se encontraron usuarios.

                    </TableCell>

                  </TableRow>

                )}

              </TableBody>

            </Table>

          </div>

        </CardContent>

      </Card>

      {/\* Diálogo Añadir Usuario \*/}

      <Dialog open={isAddDialogOpen} onOpenChange={setIsAddDialogOpen}>

        <DialogContent className="sm:max-w-[425px]">

          <DialogHeader>

            <DialogTitle>Añadir Nuevo Usuario</DialogTitle>

            <DialogDescription>Crear una nueva cuenta de usuario en el sistema</DialogDescription>

          </DialogHeader>

          {error && (

            <Alert variant="destructive">

              <AlertDescription>{error}</AlertDescription>

            </Alert>

          )}

          <Form {...form}>

            <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-4">

              <FormField

                control={form.control}

                name="name"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Nombre</FormLabel>

                    <FormControl>

                      <Input placeholder="Introduce nombre" {...field} />

                    </FormControl>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="email"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Correo electrónico</FormLabel>

                    <FormControl>

                      <Input placeholder="Introduce correo electrónico" {...field} />

                    </FormControl>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="password"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Contraseña</FormLabel>

                    <FormControl>

                      <Input type="password" placeholder="Introduce contraseña" {...field} />

                    </FormControl>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="role"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Rol</FormLabel>

                    <Select onValueChange={field.onChange} defaultValue={field.value}>

                      <FormControl>

                        <SelectTrigger>

                          <SelectValue placeholder="Selecciona rol" />

                        </SelectTrigger>

                      </FormControl>

                      <SelectContent>

                        <SelectItem value="admin">Administrador</SelectItem>

                        <SelectItem value="user">Usuario</SelectItem>

                        <SelectItem value="externo">Externo</SelectItem>

                      </SelectContent>

                    </Select>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="position"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Cargo</FormLabel>

                    <FormControl>

                      <Input placeholder="Introduce cargo" {...field} />

                    </FormControl>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="status"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Estado</FormLabel>

                    <Select onValueChange={field.onChange} defaultValue={field.value}>

                      <FormControl>

                        <SelectTrigger>

                          <SelectValue placeholder="Selecciona estado" />

                        </SelectTrigger>

                      </FormControl>

                      <SelectContent>

                        <SelectItem value="active">Activo</SelectItem>

                        <SelectItem value="inactive">Inactivo</SelectItem>

                      </SelectContent>

                    </Select>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <DialogFooter>

                <Button type="button" variant="outline" onClick={() => setIsAddDialogOpen(false)}>

                  Cancelar

                </Button>

                <Button type="submit" disabled={isSubmitting}>

                  {isSubmitting ? "Creando..." : "Crear Usuario"}

                </Button>

              </DialogFooter>

            </form>

          </Form>

        </DialogContent>

      </Dialog>

      {/\* Diálogo Editar Usuario \*/}

      <Dialog open={isEditDialogOpen} onOpenChange={setIsEditDialogOpen}>

        <DialogContent className="sm:max-w-[425px]">

          <DialogHeader>

            <DialogTitle>Editar Usuario</DialogTitle>

            <DialogDescription>Actualizar información de la cuenta de usuario</DialogDescription>

          </DialogHeader>

          {error && (

            <Alert variant="destructive">

              <AlertDescription>{error}</AlertDescription>

            </Alert>

          )}

          <Form {...form}>

            <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-4">

              <FormField

                control={form.control}

                name="name"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Nombre</FormLabel>

                    <FormControl>

                      <Input placeholder="Introduce nombre" {...field} />

                    </FormControl>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="email"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Correo electrónico</FormLabel>

                    <FormControl>

                      <Input placeholder="Introduce correo electrónico" {...field} />

                    </FormControl>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="password"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Contraseña</FormLabel>

                    <FormControl>

                      <Input

                        type="password"

                        placeholder="Introduce nueva contraseña (dejar vacío para mantener la actual)"

                        {...field}

                      />

                    </FormControl>

                    <FormDescription>Dejar vacío para mantener la contraseña actual</FormDescription>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="role"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Rol</FormLabel>

                    <Select onValueChange={field.onChange} defaultValue={field.value}>

                      <FormControl>

                        <SelectTrigger>

                          <SelectValue placeholder="Selecciona rol" />

                        </SelectTrigger>

                      </FormControl>

                      <SelectContent>

                        <SelectItem value="admin">Administrador</SelectItem>

                        <SelectItem value="user">Usuario</SelectItem>

                        <SelectItem value="externo">Externo</SelectItem>

                      </SelectContent>

                    </Select>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="position"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Cargo</FormLabel>

                    <FormControl>

                      <Input placeholder="Introduce cargo" {...field} />

                    </FormControl>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <FormField

                control={form.control}

                name="status"

                render={({ field }) => (

                  <FormItem>

                    <FormLabel>Estado</FormLabel>

                    <Select onValueChange={field.onChange} defaultValue={field.value}>

                      <FormControl>

                        <SelectTrigger>

                          <SelectValue placeholder="Selecciona estado" />

                        </SelectTrigger>

                      </FormControl>

                      <SelectContent>

                        <SelectItem value="active">Activo</SelectItem>

                        <SelectItem value="inactive">Inactivo</SelectItem>

                      </SelectContent>

                    </Select>

                    <FormMessage />

                  </FormItem>

                )}

              />

              <DialogFooter>

                <Button type="button" variant="outline" onClick={() => setIsEditDialogOpen(false)}>

                  Cancelar

                </Button>

                <Button type="submit" disabled={isSubmitting}>

                  {isSubmitting ? "Guardando..." : "Guardar Cambios"}

                </Button>

              </DialogFooter>

            </form>

          </Form>

        </DialogContent>

      </Dialog>

      {/\* Diálogo Eliminar Usuario \*/}

      <Dialog open={isDeleteDialogOpen} onOpenChange={setIsDeleteDialogOpen}>

        <DialogContent className="sm:max-w-[425px]">

          <DialogHeader>

            <DialogTitle>Eliminar Usuario</DialogTitle>

            <DialogDescription>

              ¿Estás seguro de que quieres eliminar este usuario? Esta acción no se puede deshacer.

            </DialogDescription>

          </DialogHeader>

          {error && (

            <Alert variant="destructive">

              <AlertDescription>{error}</AlertDescription>

            </Alert>

          )}

          <div className="py-4">

            <p className="mb-2">

              <strong>Nombre:</strong> {currentUser?.name}

            </p>

            <p className="mb-2">

              <strong>Correo electrónico:</strong> {currentUser?.email}

            </p>

            <p>

              <strong>Cargo:</strong> {currentUser?.position}

            </p>

          </div>

          <DialogFooter>

            <Button type="button" variant="outline" onClick={() => setIsDeleteDialogOpen(false)}>

              Cancelar

            </Button>

            <Button type="button" variant="destructive" onClick={confirmDelete} disabled={isSubmitting}>

              {isSubmitting ? "Eliminando..." : "Eliminar Usuario"}

            </Button>

          </DialogFooter>

        </DialogContent>

      </Dialog>

    </div>

  )

}

Dashboard/

"use client"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { Button } from "@/components/ui/button"

import { FileText, Upload, Clock, AlertTriangle, CheckCircle, Pill } from "lucide-react"

import Link from "next/link"

import { useAuth } from "@/app/context/auth-context"

export default function DashboardPage() {

  const { user } = useAuth()

  return (

    <div className="space-y-6">

      <div>

        <h1 className="text-3xl font-bold tracking-tight">Panel de Control</h1>

        <p className="text-muted-foreground">

          Bienvenido a Medcol Docs, {user?.name}. Aquí tienes una visión general de tu sistema.

        </p>

      </div>

      <div className="grid gap-6 md:grid-cols-2 lg:grid-cols-4">

        <Card>

          <CardHeader className="flex flex-row items-center justify-between pb-2">

            <CardTitle className="text-sm font-medium">Documentos Pendientes</CardTitle>

            <FileText className="h-4 w-4 text-muted-foreground" />

          </CardHeader>

          <CardContent>

            <div className="text-2xl font-bold">12</div>

            <p className="text-xs text-muted-foreground">+2 desde ayer</p>

          </CardContent>

        </Card>

        <Card>

          <CardHeader className="flex flex-row items-center justify-between pb-2">

            <CardTitle className="text-sm font-medium">Medicamentos Pendientes</CardTitle>

            <Pill className="h-4 w-4 text-muted-foreground" />

          </CardHeader>

          <CardContent>

            <div className="text-2xl font-bold">8</div>

            <p className="text-xs text-muted-foreground">-3 desde ayer</p>

          </CardContent>

        </Card>

        <Card>

          <CardHeader className="flex flex-row items-center justify-between pb-2">

            <CardTitle className="text-sm font-medium">Documentos Hoy</CardTitle>

            <CheckCircle className="h-4 w-4 text-muted-foreground" />

          </CardHeader>

          <CardContent>

            <div className="text-2xl font-bold">24</div>

            <p className="text-xs text-muted-foreground">+5 del promedio</p>

          </CardContent>

        </Card>

        <Card>

          <CardHeader className="flex flex-row items-center justify-between pb-2">

            <CardTitle className="text-sm font-medium">Alertas del Sistema</CardTitle>

            <AlertTriangle className="h-4 w-4 text-muted-foreground" />

          </CardHeader>

          <CardContent>

            <div className="text-2xl font-bold">3</div>

            <p className="text-xs text-muted-foreground">Requieren atención</p>

          </CardContent>

        </Card>

      </div>

      <div className="grid gap-6 md:grid-cols-2 lg:grid-cols-3">

        <Card>

          <CardHeader>

            <CardTitle>Acciones Rápidas</CardTitle>

            <CardDescription>Tareas comunes que puedes realizar</CardDescription>

          </CardHeader>

          <CardContent className="grid gap-4">

            <Button asChild className="w-full">

              <Link href="/document-capture">

                <Upload className="mr-2 h-4 w-4" />

                Subir Nuevo Documento

              </Link>

            </Button>

            <Button asChild variant="outline" className="w-full">

              <Link href="/document-consultation">

                <FileText className="mr-2 h-4 w-4" />

                Ver Documentos Recientes

              </Link>

            </Button>

            <Button asChild variant="outline" className="w-full">

              <Link href="/medication-record">

                <Pill className="mr-2 h-4 w-4" />

                Revisar Registros de Medicamentos

              </Link>

            </Button>

          </CardContent>

        </Card>

        <Card>

          <CardHeader>

            <CardTitle>Actividad Reciente</CardTitle>

            <CardDescription>Tus últimas acciones en el sistema</CardDescription>

          </CardHeader>

          <CardContent>

            <div className="space-y-4">

              {[1, 2, 3, 4].map((i) => (

                <div key={i} className="flex items-center gap-4">

                  <div className="rounded-full bg-primary/10 p-2">

                    <Clock className="h-4 w-4 text-primary" />

                  </div>

                  <div className="flex-1 space-y-1">

                    <p className="text-sm font-medium leading-none">

                      {i === 1

                        ? "Documento subido"

                        : i === 2

                          ? "Registro de medicamento actualizado"

                          : i === 3

                            ? "Registro de paciente visto"

                            : "Alerta del sistema reconocida"}

                    </p>

                    <p className="text-xs text-muted-foreground">

                      {i === 1

                        ? "Hace 10 minutos"

                        : i === 2

                          ? "Hace 2 horas"

                          : i === 3

                            ? "Ayer a las 15:30"

                            : "Hace 2 días"}

                    </p>

                  </div>

                </div>

              ))}

            </div>

          </CardContent>

        </Card>

        <Card>

          <CardHeader>

            <CardTitle>Notificaciones del Sistema</CardTitle>

            <CardDescription>Alertas importantes del sistema</CardDescription>

          </CardHeader>

          <CardContent>

            <div className="space-y-4">

              <div className="flex items-center gap-4">

                <div className="rounded-full bg-red-100 p-2">

                  <AlertTriangle className="h-4 w-4 text-red-600" />

                </div>

                <div className="flex-1 space-y-1">

                  <p className="text-sm font-medium leading-none">Mantenimiento del Sistema</p>

                  <p className="text-xs text-muted-foreground">Mantenimiento programado el sábado a las 02:00</p>

                </div>

              </div>

              <div className="flex items-center gap-4">

                <div className="rounded-full bg-yellow-100 p-2">

                  <AlertTriangle className="h-4 w-4 text-yellow-600" />

                </div>

                <div className="flex-1 space-y-1">

                  <p className="text-sm font-medium leading-none">Advertencia de Almacenamiento</p>

                  <p className="text-xs text-muted-foreground">Almacenamiento de documentos al 85% de capacidad</p>

                </div>

              </div>

              <div className="flex items-center gap-4">

                <div className="rounded-full bg-blue-100 p-2">

                  <AlertTriangle className="h-4 w-4 text-blue-600" />

                </div>

                <div className="flex-1 space-y-1">

                  <p className="text-sm font-medium leading-none">Nueva Función Disponible</p>

                  <p className="text-xs text-muted-foreground">

                    Procesamiento por lotes de documentos ahora disponible

                  </p>

                </div>

              </div>

            </div>

          </CardContent>

        </Card>

      </div>

    </div>

  )

}

Document-capture/

"use client"

import type React from "react"

import { useState, useEffect, useRef } from "react"

import { z } from "zod"

import { useForm } from "react-hook-form"

import { zodResolver } from "@hookform/resolvers/zod"

import { Button } from "@/components/ui/button"

import { Form, FormControl, FormField, FormItem, FormMessage } from "@/components/ui/form"

import { Input } from "@/components/ui/input"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogDescription } from "@/components/ui/dialog"

import { Search, Upload, Camera, FileText, Download, Eye, Check, X } from "lucide-react"

import { Alert, AlertDescription, AlertTitle } from "@/components/ui/alert"

import { useAuth } from "@/app/context/auth-context"

import { useRouter } from "next/navigation"

// Datos simulados de pacientes (en una aplicación real, esto vendría de una API)

const patientData = [

  {

    cedula: "1098765432",

    nombre: "Maria Alejandra Rodriguez Gomez",

    direccion: "Calle 123 #45-67, Bogotá, Colombia",

    eps: "Sanitas EPS",

    msd: "MSD-001",

  },

  {

    cedula: "0987654321",

    nombre: "Carlos Andrés Martínez López",

    direccion: "Carrera 45 #12-34, Medellín, Colombia",

    eps: "Nueva EPS",

    msd: "MSD-002",

  },

  {

    cedula: "5678901234",

    nombre: "Ana María Pérez Sánchez",

    direccion: "Avenida 67 #89-12, Cali, Colombia",

    eps: "Compensar EPS",

    msd: "MSD-003",

  },

]

const formSchema = z.object({

  searchType: z.enum(["msd", "cedula"]),

  searchTerm: z.string().min(1, "Por favor, introduce un término de búsqueda"),

})

export default function DocumentCapturePage() {

  const { user } = useAuth()

  const router = useRouter()

  const [isSubmitting, setIsSubmitting] = useState(false)

  const [isSuccess, setIsSuccess] = useState(false)

  const [patientFound, setPatientFound] = useState<(typeof patientData)[0] | null>(null)

  const [isCameraOpen, setIsCameraOpen] = useState(false)

  const [currentDocumentType, setCurrentDocumentType] = useState<string>("")

  const [capturedImages, setCapturedImages] = useState<Record<string, string>>({})

  const [uploadedFiles, setUploadedFiles] = useState<Record<string, File>>({})

  const [stream, setStream] = useState<MediaStream | null>(null)

  const [isPreviewOpen, setIsPreviewOpen] = useState(false)

  const [previewImage, setPreviewImage] = useState<string | null>(null)

  const videoRef = useRef<HTMLVideoElement>(null)

  const canvasRef = useRef<HTMLCanvasElement>(null)

  const form = useForm<z.infer<typeof formSchema>>({

    resolver: zodResolver(formSchema),

    defaultValues: {

      searchType: "msd",

      searchTerm: "",

    },

  })

  // Redirigir si el usuario es externo

  useEffect(() => {

    if (user?.role === "externo") {

      router.push("/dashboard")

    }

  }, [user, router])

  // Limpiar el stream de la cámara cuando se cierra

  useEffect(() => {

    return () => {

      if (stream) {

        stream.getTracks().forEach((track) => track.stop())

      }

    }

  }, [stream])

  // Si el usuario es externo, mostrar mensaje de acceso denegado

  if (user?.role === "externo") {

    return (

      <div className="flex h-full items-center justify-center">

        <div className="text-center">

          <h1 className="text-2xl font-bold">Acceso Denegado</h1>

          <p className="text-muted-foreground">No tienes permiso para acceder a esta página.</p>

        </div>

      </div>

    )

  }

  // Función para buscar paciente

  const searchPatient = (data: z.infer<typeof formSchema>) => {

    let found = null

    if (data.searchType === "msd") {

      found = patientData.find((patient) => patient.msd === data.searchTerm)

    } else {

      found = patientData.find((patient) => patient.cedula === data.searchTerm)

    }

    if (found) {

      setPatientFound(found)

    } else {

      setPatientFound(null)

      alert("No se encontró ningún paciente con los datos proporcionados.")

    }

  }

  // Función para abrir la cámara

  const openCamera = (documentType: string) => {

    setCurrentDocumentType(documentType)

    navigator.mediaDevices

      .getUserMedia({ video: true })

      .then((mediaStream) => {

        setStream(mediaStream)

        if (videoRef.current) {

          videoRef.current.srcObject = mediaStream

        }

        setIsCameraOpen(true)

      })

      .catch((error) => {

        console.error("Error al acceder a la cámara:", error)

        alert("No se pudo acceder a la cámara. Por favor, verifica los permisos.")

      })

  }

  // Función para capturar imagen

  const captureImage = () => {

    if (videoRef.current && canvasRef.current && currentDocumentType) {

      const video = videoRef.current

      const canvas = canvasRef.current

      const context = canvas.getContext("2d")

      if (context) {

        canvas.width = video.videoWidth

        canvas.height = video.videoHeight

        context.drawImage(video, 0, 0, canvas.width, canvas.height)

        const imageDataUrl = canvas.toDataURL("image/png")

        setCapturedImages((prev) => ({

          ...prev,

          [currentDocumentType]: imageDataUrl,

        }))

        // Detener la cámara

        if (stream) {

          stream.getTracks().forEach((track) => track.stop())

        }

        setIsCameraOpen(false)

      }

    }

  }

  // Función para cerrar la cámara

  const closeCamera = () => {

    if (stream) {

      stream.getTracks().forEach((track) => track.stop())

    }

    setIsCameraOpen(false)

  }

  // Función para manejar la subida de archivos

  const handleFileUpload = (documentType: string, e: React.ChangeEvent<HTMLInputElement>) => {

    if (e.target.files && e.target.files.length > 0) {

      const file = e.target.files[0]

      setUploadedFiles((prev) => ({

        ...prev,

        [documentType]: file,

      }))

      // Crear URL para previsualización

      const reader = new FileReader()

      reader.onload = (event) => {

        if (event.target && event.target.result) {

          setCapturedImages((prev) => ({

            ...prev,

            [documentType]: event.target?.result as string,

          }))

        }

      }

      reader.readAsDataURL(file)

    }

  }

  // Función para eliminar un documento

  const removeDocument = (documentType: string) => {

    setCapturedImages((prev) => {

      const newImages = { ...prev }

      delete newImages[documentType]

      return newImages

    })

    setUploadedFiles((prev) => {

      const newFiles = { ...prev }

      delete newFiles[documentType]

      return newFiles

    })

  }

  // Función para previsualizar un documento

  const previewDocument = (imageUrl: string) => {

    setPreviewImage(imageUrl)

    setIsPreviewOpen(true)

  }

  // Función para guardar todos los documentos

  const saveAllDocuments = () => {

    setIsSubmitting(true)

    // Simular envío a API

    setTimeout(() => {

      setIsSuccess(true)

      setIsSubmitting(false)

      // Resetear después de 3 segundos

      setTimeout(() => {

        setIsSuccess(false)

        setPatientFound(null)

        setCapturedImages({})

        setUploadedFiles({})

        form.reset()

      }, 3000)

    }, 1500)

  }

  // Verificar si hay documentos para guardar

  const hasDocumentsToSave = Object.keys(capturedImages).length > 0 || Object.keys(uploadedFiles).length > 0

  return (

    <div className="space-y-6">

      <div>

        <h1 className="text-3xl font-bold tracking-tight">Captura de Documentos</h1>

        <p className="text-muted-foreground">Sube documentos de dispensación de medicamentos</p>

      </div>

      {isSuccess && (

        <Alert className="bg-green-50">

          <Check className="h-4 w-4 text-green-600" />

          <AlertTitle>Éxito</AlertTitle>

          <AlertDescription>Los documentos han sido guardados correctamente.</AlertDescription>

        </Alert>

      )}

      <Card>

        <CardHeader>

          <CardTitle>Buscar Paciente</CardTitle>

          <CardDescription>Introduce el número MSD o cédula para buscar la información del paciente</CardDescription>

        </CardHeader>

        <CardContent>

          <Form {...form}>

            <form onSubmit={form.handleSubmit(searchPatient)} className="space-y-4">

              <div className="flex flex-col space-y-4">

                <div className="flex space-x-4">

                  <FormField

                    control={form.control}

                    name="searchType"

                    render={({ field }) => (

                      <FormItem className="space-y-0 flex items-center space-x-4">

                        <div className="flex items-center space-x-2">

                          <input

                            type="radio"

                            id="msd"

                            value="msd"

                            checked={field.value === "msd"}

                            onChange={() => field.onChange("msd")}

                            className="mr-1"

                          />

                          <label htmlFor="msd">MSD Number</label>

                        </div>

                        <div className="flex items-center space-x-2">

                          <input

                            type="radio"

                            id="cedula"

                            value="cedula"

                            checked={field.value === "cedula"}

                            onChange={() => field.onChange("cedula")}

                            className="mr-1"

                          />

                          <label htmlFor="cedula">ID Card (Cédula)</label>

                        </div>

                      </FormItem>

                    )}

                  />

                </div>

                <div className="flex space-x-4">

                  <FormField

                    control={form.control}

                    name="searchTerm"

                    render={({ field }) => (

                      <FormItem className="flex-1">

                        <FormControl>

                          <div className="relative">

                            <Search className="absolute left-2.5 top-2.5 h-4 w-4 text-muted-foreground" />

                            <Input

                              placeholder={`Introduce ${form.getValues().searchType === "msd" ? "MSD" : "cédula"}`}

                              className="pl-8"

                              {...field}

                            />

                          </div>

                        </FormControl>

                        <FormMessage />

                      </FormItem>

                    )}

                  />

                  <Button type="submit">

                    <Search className="mr-2 h-4 w-4" />

                    Buscar Paciente

                  </Button>

                </div>

              </div>

            </form>

          </Form>

        </CardContent>

      </Card>

      {patientFound && (

        <>

          <Card>

            <CardHeader>

              <CardTitle>Información del Paciente</CardTitle>

            </CardHeader>

            <CardContent>

              <div className="grid grid-cols-1 md:grid-cols-2 gap-6">

                <div>

                  <h3 className="text-sm font-medium text-gray-500">Nombre Completo</h3>

                  <p className="mt-1 text-lg font-medium">{patientFound.nombre}</p>

                </div>

                <div>

                  <h3 className="text-sm font-medium text-gray-500">Número de Cédula</h3>

                  <p className="mt-1 text-lg font-medium">{patientFound.cedula}</p>

                </div>

                <div>

                  <h3 className="text-sm font-medium text-gray-500">Dirección</h3>

                  <p className="mt-1 text-lg font-medium">{patientFound.direccion}</p>

                </div>

                <div>

                  <h3 className="text-sm font-medium text-gray-500">EPS</h3>

                  <p className="mt-1 text-lg font-medium">{patientFound.eps}</p>

                </div>

              </div>

            </CardContent>

          </Card>

          <Card>

            <CardHeader>

              <CardTitle>Documentos Requeridos</CardTitle>

              <CardDescription>Sube o captura los siguientes documentos requeridos</CardDescription>

            </CardHeader>

            <CardContent className="space-y-6">

              {/\* Autorización \*/}

              <div className="border-b pb-6">

                <div className="flex justify-between items-start mb-2">

                  <div>

                    <h3 className="text-lg font-medium">Autorización</h3>

                    <p className="text-sm text-gray-500">Sube o captura una imagen clara de la autorización</p>

                  </div>

                  <div className="flex space-x-2">

                    <Button variant="outline" onClick={() => document.getElementById("autorizacion-upload")?.click()}>

                      <Upload className="mr-2 h-4 w-4" />

                      Subir Archivo

                    </Button>

                    <input

                      id="autorizacion-upload"

                      type="file"

                      className="hidden"

                      accept="image/\*,.pdf"

                      onChange={(e) => handleFileUpload("autorizacion", e)}

                    />

                    <Button variant="outline" onClick={() => openCamera("autorizacion")}>

                      <Camera className="mr-2 h-4 w-4" />

                      Tomar Foto

                    </Button>

                  </div>

                </div>

                {capturedImages["autorizacion"] && (

                  <div className="mt-4 relative bg-gray-50 p-4 rounded-md">

                    <div className="flex items-center justify-between">

                      <div className="flex items-center">

                        <FileText className="h-8 w-8 text-blue-500 mr-2" />

                        <div>

                          <p className="font-medium">Autorización</p>

                          <p className="text-sm text-gray-500">Documento capturado</p>

                        </div>

                      </div>

                      <div className="flex space-x-2">

                        <Button

                          variant="ghost"

                          size="sm"

                          onClick={() => previewDocument(capturedImages["autorizacion"])}

                        >

                          <Eye className="h-4 w-4 mr-1" />

                          Ver

                        </Button>

                        <Button

                          variant="ghost"

                          size="sm"

                          className="text-red-500 hover:text-red-700"

                          onClick={() => removeDocument("autorizacion")}

                        >

                          <X className="h-4 w-4 mr-1" />

                          Eliminar

                        </Button>

                      </div>

                    </div>

                  </div>

                )}

              </div>

              {/\* Cédula \*/}

              <div className="border-b pb-6">

                <div className="flex justify-between items-start mb-2">

                  <div>

                    <h3 className="text-lg font-medium">ID Card (Cédula)</h3>

                    <p className="text-sm text-gray-500">Sube o captura una imagen clara de la cédula</p>

                  </div>

                  <div className="flex space-x-2">

                    <Button variant="outline" onClick={() => document.getElementById("cedula-upload")?.click()}>

                      <Upload className="mr-2 h-4 w-4" />

                      Subir Archivo

                    </Button>

                    <input

                      id="cedula-upload"

                      type="file"

                      className="hidden"

                      accept="image/\*,.pdf"

                      onChange={(e) => handleFileUpload("cedula", e)}

                    />

                    <Button variant="outline" onClick={() => openCamera("cedula")}>

                      <Camera className="mr-2 h-4 w-4" />

                      Tomar Foto

                    </Button>

                  </div>

                </div>

                {capturedImages["cedula"] && (

                  <div className="mt-4 relative bg-gray-50 p-4 rounded-md">

                    <div className="flex items-center justify-between">

                      <div className="flex items-center">

                        <FileText className="h-8 w-8 text-blue-500 mr-2" />

                        <div>

                          <p className="font-medium">Cédula</p>

                          <p className="text-sm text-gray-500">Documento capturado</p>

                        </div>

                      </div>

                      <div className="flex space-x-2">

                        <Button variant="ghost" size="sm" onClick={() => previewDocument(capturedImages["cedula"])}>

                          <Eye className="h-4 w-4 mr-1" />

                          Ver

                        </Button>

                        <Button

                          variant="ghost"

                          size="sm"

                          className="text-red-500 hover:text-red-700"

                          onClick={() => removeDocument("cedula")}

                        >

                          <X className="h-4 w-4 mr-1" />

                          Eliminar

                        </Button>

                      </div>

                    </div>

                  </div>

                )}

              </div>

              {/\* Fórmula Médica \*/}

              <div className="border-b pb-6">

                <div className="flex justify-between items-start mb-2">

                  <div>

                    <h3 className="text-lg font-medium">Fórmula Médica</h3>

                    <p className="text-sm text-gray-500">Sube o captura una imagen clara de la fórmula médica</p>

                  </div>

                  <div className="flex space-x-2">

                    <Button variant="outline" onClick={() => document.getElementById("formula-upload")?.click()}>

                      <Upload className="mr-2 h-4 w-4" />

                      Subir Archivo

                    </Button>

                    <input

                      id="formula-upload"

                      type="file"

                      className="hidden"

                      accept="image/\*,.pdf"

                      onChange={(e) => handleFileUpload("formula", e)}

                    />

                    <Button variant="outline" onClick={() => openCamera("formula")}>

                      <Camera className="mr-2 h-4 w-4" />

                      Tomar Foto

                    </Button>

                  </div>

                </div>

                {capturedImages["formula"] && (

                  <div className="mt-4 relative bg-gray-50 p-4 rounded-md">

                    <div className="flex items-center justify-between">

                      <div className="flex items-center">

                        <FileText className="h-8 w-8 text-blue-500 mr-2" />

                        <div>

                          <p className="font-medium">Fórmula Médica</p>

                          <p className="text-sm text-gray-500">Documento capturado</p>

                        </div>

                      </div>

                      <div className="flex space-x-2">

                        <Button variant="ghost" size="sm" onClick={() => previewDocument(capturedImages["formula"])}>

                          <Eye className="h-4 w-4 mr-1" />

                          Ver

                        </Button>

                        <Button

                          variant="ghost"

                          size="sm"

                          className="text-red-500 hover:text-red-700"

                          onClick={() => removeDocument("formula")}

                        >

                          <X className="h-4 w-4 mr-1" />

                          Eliminar

                        </Button>

                      </div>

                    </div>

                  </div>

                )}

              </div>

              {/\* MSD \*/}

              <div className="pb-6">

                <div className="flex justify-between items-start mb-2">

                  <div>

                    <h3 className="text-lg font-medium">MSD</h3>

                    <p className="text-sm text-gray-500">Sube o captura una imagen clara del MSD</p>

                  </div>

                  <div className="flex space-x-2">

                    <Button variant="outline" onClick={() => document.getElementById("msd-upload")?.click()}>

                      <Upload className="mr-2 h-4 w-4" />

                      Subir Archivo

                    </Button>

                    <input

                      id="msd-upload"

                      type="file"

                      className="hidden"

                      accept="image/\*,.pdf"

                      onChange={(e) => handleFileUpload("msd", e)}

                    />

                    <Button variant="outline" onClick={() => openCamera("msd")}>

                      <Camera className="mr-2 h-4 w-4" />

                      Tomar Foto

                    </Button>

                  </div>

                </div>

                {capturedImages["msd"] && (

                  <div className="mt-4 relative bg-gray-50 p-4 rounded-md">

                    <div className="flex items-center justify-between">

                      <div className="flex items-center">

                        <FileText className="h-8 w-8 text-blue-500 mr-2" />

                        <div>

                          <p className="font-medium">MSD</p>

                          <p className="text-sm text-gray-500">Documento capturado</p>

                        </div>

                      </div>

                      <div className="flex space-x-2">

                        <Button variant="ghost" size="sm" onClick={() => previewDocument(capturedImages["msd"])}>

                          <Eye className="h-4 w-4 mr-1" />

                          Ver

                        </Button>

                        <Button

                          variant="ghost"

                          size="sm"

                          className="text-red-500 hover:text-red-700"

                          onClick={() => removeDocument("msd")}

                        >

                          <X className="h-4 w-4 mr-1" />

                          Eliminar

                        </Button>

                      </div>

                    </div>

                  </div>

                )}

              </div>

              <div className="flex items-center space-x-2 pt-4">

                <input

                  type="checkbox"

                  id="pending-delivery"

                  className="h-4 w-4 rounded border-gray-300 text-primary focus:ring-primary"

                />

                <label htmlFor="pending-delivery" className="text-sm font-medium text-gray-700">

                  Marcar como entrega pendiente

                </label>

              </div>

              <div className="pt-4 flex justify-end">

                <Button

                  onClick={saveAllDocuments}

                  disabled={!hasDocumentsToSave || isSubmitting}

                  className="w-full md:w-auto"

                >

                  {isSubmitting ? "Guardando..." : "Guardar Documentos"}

                </Button>

              </div>

            </CardContent>

          </Card>

        </>

      )}

      {/\* Diálogo para la cámara \*/}

      <Dialog open={isCameraOpen} onOpenChange={setIsCameraOpen}>

        <DialogContent className="sm:max-w-[600px]">

          <DialogHeader>

            <DialogTitle>Capturar Documento</DialogTitle>

            <DialogDescription>Posiciona el documento frente a la cámara y haz clic en "Capturar"</DialogDescription>

          </DialogHeader>

          <div className="flex flex-col items-center space-y-4">

            <div className="relative w-full border rounded-lg overflow-hidden">

              <video ref={videoRef} autoPlay playsInline className="w-full h-auto" />

            </div>

            <div className="flex space-x-4">

              <Button onClick={captureImage}>

                <Camera className="mr-2 h-4 w-4" />

                Capturar

              </Button>

              <Button variant="outline" onClick={closeCamera}>

                Cancelar

              </Button>

            </div>

          </div>

          {/\* Canvas oculto para capturar la imagen \*/}

          <canvas ref={canvasRef} className="hidden" />

        </DialogContent>

      </Dialog>

      {/\* Diálogo para previsualización de documentos \*/}

      <Dialog open={isPreviewOpen} onOpenChange={setIsPreviewOpen}>

        <DialogContent className="sm:max-w-[700px]">

          <DialogHeader>

            <DialogTitle>Previsualización de Documento</DialogTitle>

          </DialogHeader>

          <div className="flex justify-center">

            <img

              src={previewImage || "/placeholder.svg"}

              alt="Documento"

              className="max-h-[500px] object-contain border rounded-lg"

            />

          </div>

          <div className="flex justify-end space-x-2">

            <Button variant="outline" onClick={() => setIsPreviewOpen(false)}>

              Cerrar

            </Button>

            {previewImage && (

              <Button onClick={() => window.open(previewImage, "\_blank")}>

                <Download className="mr-2 h-4 w-4" />

                Descargar

              </Button>

            )}

          </div>

        </DialogContent>

      </Dialog>

    </div>

  )

}

Document-consultation/

"use client"

import { useState } from "react"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { Button } from "@/components/ui/button"

import { Input } from "@/components/ui/input"

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

import {

  Dialog,

  DialogContent,

  DialogHeader,

  DialogTitle,

  DialogDescription,

  DialogFooter,

} from "@/components/ui/dialog"

import { Search, FileText, Download, Eye } from "lucide-react"

import { useAuth } from "@/app/context/auth-context"

// Datos de ejemplo para documentos

const documents = [

  {

    id: "DOC-001",

    patientId: "1098765432",

    patientName: "Maria Alejandra Rodriguez Gomez",

    type: "Autorización",

    date: "2023-05-15",

    status: "Completo",

    msd: "MSD-001",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-002",

    patientId: "0987654321",

    patientName: "Carlos Andrés Martínez López",

    type: "Cédula",

    date: "2023-05-14",

    status: "Pendiente",

    msd: "MSD-002",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-003",

    patientId: "5678901234",

    patientName: "Ana María Pérez Sánchez",

    type: "Fórmula Médica",

    date: "2023-05-13",

    status: "Completo",

    msd: "MSD-003",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-004",

    patientId: "1098765432",

    patientName: "Maria Alejandra Rodriguez Gomez",

    type: "MSD",

    date: "2023-05-12",

    status: "Pendiente",

    msd: "MSD-001",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

]

export default function DocumentConsultationPage() {

  const { user } = useAuth()

  const [searchTerm, setSearchTerm] = useState("")

  const [searchType, setSearchType] = useState<"cedula" | "msd">("cedula")

  const [filteredDocuments, setFilteredDocuments] = useState<typeof documents>([])

  const [isPreviewOpen, setIsPreviewOpen] = useState(false)

  const [currentDocument, setCurrentDocument] = useState<(typeof documents)[0] | null>(null)

  const [hasSearched, setHasSearched] = useState(false)

  const handleSearch = () => {

    if (!searchTerm) {

      setFilteredDocuments([])

      setHasSearched(false)

      return

    }

    const filtered = documents.filter((doc) =>

      searchType === "cedula" ? doc.patientId.includes(searchTerm) : doc.msd.includes(searchTerm),

    )

    setFilteredDocuments(filtered)

    setHasSearched(true)

  }

  const openPreview = (document: (typeof documents)[0]) => {

    setCurrentDocument(document)

    setIsPreviewOpen(true)

  }

  return (

    <div className="space-y-6">

      <div>

        <h1 className="text-3xl font-bold tracking-tight">Consultar Documentos</h1>

        <p className="text-muted-foreground">Busca y consulta documentos por cédula o número MSD</p>

      </div>

      <Card>

        <CardHeader>

          <CardTitle>Buscar Documentos</CardTitle>

          <CardDescription>Introduce la cédula del paciente o el número MSD para buscar documentos</CardDescription>

        </CardHeader>

        <CardContent>

          <div className="flex items-end gap-4">

            <div className="flex-1">

              <div className="mb-2">

                <label className="text-sm font-medium">Buscar por:</label>

                <div className="flex mt-1 space-x-4">

                  <div className="flex items-center">

                    <input

                      type="radio"

                      id="cedula-search"

                      name="searchType"

                      className="mr-2"

                      checked={searchType === "cedula"}

                      onChange={() => setSearchType("cedula")}

                    />

                    <label htmlFor="cedula-search">Cédula</label>

                  </div>

                  <div className="flex items-center">

                    <input

                      type="radio"

                      id="msd-search"

                      name="searchType"

                      className="mr-2"

                      checked={searchType === "msd"}

                      onChange={() => setSearchType("msd")}

                    />

                    <label htmlFor="msd-search">MSD</label>

                  </div>

                </div>

              </div>

              <div className="relative">

                <Search className="absolute left-2.5 top-2.5 h-4 w-4 text-muted-foreground" />

                <Input

                  type="text"

                  placeholder={`Introduce ${searchType === "cedula" ? "cédula" : "MSD"}`}

                  className="pl-8"

                  value={searchTerm}

                  onChange={(e) => setSearchTerm(e.target.value)}

                />

              </div>

            </div>

            <Button onClick={handleSearch}>Buscar</Button>

          </div>

        </CardContent>

      </Card>

      {hasSearched && (

        <Card>

          <CardHeader>

            <CardTitle>Resultados de la Búsqueda</CardTitle>

            <CardDescription>

              {filteredDocuments.length > 0

                ? `Se encontraron ${filteredDocuments.length} documentos`

                : "No se encontraron documentos con los criterios de búsqueda"}

            </CardDescription>

          </CardHeader>

          <CardContent>

            {filteredDocuments.length > 0 ? (

              <div className="rounded-md border">

                <Table>

                  <TableHeader>

                    <TableRow>

                      <TableHead>ID Documento</TableHead>

                      <TableHead>Paciente</TableHead>

                      <TableHead>Cédula</TableHead>

                      <TableHead>Tipo</TableHead>

                      <TableHead>Fecha</TableHead>

                      <TableHead>MSD</TableHead>

                      <TableHead>Estado</TableHead>

                      <TableHead className="text-right">Acciones</TableHead>

                    </TableRow>

                  </TableHeader>

                  <TableBody>

                    {filteredDocuments.map((doc) => (

                      <TableRow key={doc.id}>

                        <TableCell className="font-medium">{doc.id}</TableCell>

                        <TableCell>{doc.patientName}</TableCell>

                        <TableCell>{doc.patientId}</TableCell>

                        <TableCell>{doc.type}</TableCell>

                        <TableCell>{new Date(doc.date).toLocaleDateString()}</TableCell>

                        <TableCell>{doc.msd}</TableCell>

                        <TableCell>

                          <div

                            className={`px-2 py-1 rounded-full text-xs font-medium inline-block ${

                              doc.status === "Completo"

                                ? "bg-green-100 text-green-800"

                                : "bg-yellow-100 text-yellow-800"

                            }`}

                          >

                            {doc.status}

                          </div>

                        </TableCell>

                        <TableCell className="text-right">

                          <div className="flex justify-end space-x-2">

                            <Button variant="ghost" size="sm" onClick={() => openPreview(doc)}>

                              <Eye className="h-4 w-4" />

                            </Button>

                            <Button variant="ghost" size="sm" onClick={() => window.open(doc.fileUrl, "\_blank")}>

                              <Download className="h-4 w-4" />

                            </Button>

                          </div>

                        </TableCell>

                      </TableRow>

                    ))}

                  </TableBody>

                </Table>

              </div>

            ) : (

              <div className="flex flex-col items-center justify-center py-12">

                <FileText className="h-16 w-16 text-gray-300 mb-4" />

                <h3 className="text-lg font-medium text-gray-900">No se encontraron documentos</h3>

                <p className="text-sm text-gray-500 mt-2">Intenta con diferentes criterios de búsqueda</p>

              </div>

            )}

          </CardContent>

        </Card>

      )}

      {/\* Diálogo de previsualización \*/}

      <Dialog open={isPreviewOpen} onOpenChange={setIsPreviewOpen}>

        <DialogContent className="sm:max-w-[700px]">

          <DialogHeader>

            <DialogTitle>Previsualización de Documento</DialogTitle>

            <DialogDescription>

              {currentDocument?.type} - {currentDocument?.patientName} - {currentDocument?.msd}

            </DialogDescription>

          </DialogHeader>

          <div className="flex justify-center">

            <img

              src={currentDocument?.fileUrl || "/placeholder.svg"}

              alt="Documento"

              className="max-h-[500px] object-contain border rounded-lg"

            />

          </div>

          <DialogFooter>

            <Button variant="outline" onClick={() => setIsPreviewOpen(false)}>

              Cerrar

            </Button>

            <Button onClick={() => window.open(currentDocument?.fileUrl, "\_blank")}>

              <Download className="mr-2 h-4 w-4" />

              Descargar

            </Button>

          </DialogFooter>

        </DialogContent>

      </Dialog>

    </div>

  )

}

Medication-record/

"use client"

import { useState, useEffect } from "react"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { Button } from "@/components/ui/button"

import { Input } from "@/components/ui/input"

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

import { DropdownMenu, DropdownMenuContent, DropdownMenuItem, DropdownMenuTrigger } from "@/components/ui/dropdown-menu"

import { Badge } from "@/components/ui/badge"

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"

import { Search, MoreVertical, Download, Printer, Eye, Pill, CheckCircle } from "lucide-react"

// Importar useAuth y useRouter al principio del archivo

import { useAuth } from "@/app/context/auth-context"

import { useRouter } from "next/navigation"

// Datos de ejemplo para medicamentos

const deliveredMedications = [

  {

    id: "MED-001",

    patientId: "P-12345",

    name: "Amoxicilina",

    dosage: "500mg",

    frequency: "3x diario",

    deliveryDate: "2023-05-15",

    status: "Entregado",

  },

  {

    id: "MED-002",

    patientId: "P-67890",

    name: "Lisinopril",

    dosage: "10mg",

    frequency: "1x diario",

    deliveryDate: "2023-05-14",

    status: "Entregado",

  },

  {

    id: "MED-003",

    patientId: "P-54321",

    name: "Metformina",

    dosage: "1000mg",

    frequency: "2x diario",

    deliveryDate: "2023-05-13",

    status: "Entregado",

  },

]

const pendingMedications = [

  {

    id: "MED-004",

    patientId: "P-98765",

    name: "Atorvastatina",

    dosage: "20mg",

    frequency: "1x diario",

    requestDate: "2023-05-12",

    status: "Pendiente",

  },

  {

    id: "MED-005",

    patientId: "P-13579",

    name: "Levotiroxina",

    dosage: "50mcg",

    frequency: "1x diario",

    requestDate: "2023-05-11",

    status: "Pendiente",

  },

]

// Añadir verificación de permisos al principio de la función del componente

export default function MedicationRecordPage() {

  const { user } = useAuth()

  const router = useRouter()

  const [deliveredSearchTerm, setDeliveredSearchTerm] = useState("")

  const [pendingSearchTerm, setPendingSearchTerm] = useState("")

  // Redirigir si el usuario es externo

  useEffect(() => {

    if (user?.role === "externo") {

      router.push("/dashboard")

    }

  }, [user, router])

  // Si el usuario es externo, mostrar mensaje de acceso denegado

  if (user?.role === "externo") {

    return (

      <div className="flex h-full items-center justify-center">

        <div className="text-center">

          <h1 className="text-2xl font-bold">Acceso Denegado</h1>

          <p className="text-muted-foreground">No tienes permiso para acceder a esta página.</p>

        </div>

      </div>

    )

  }

  const filteredDeliveredMedications = deliveredMedications.filter(

    (med) =>

      med.name.toLowerCase().includes(deliveredSearchTerm.toLowerCase()) ||

      med.patientId.toLowerCase().includes(deliveredSearchTerm.toLowerCase()),

  )

  const filteredPendingMedications = pendingMedications.filter(

    (med) =>

      med.name.toLowerCase().includes(pendingSearchTerm.toLowerCase()) ||

      med.patientId.toLowerCase().includes(pendingSearchTerm.toLowerCase()),

  )

  return (

    <div className="space-y-6">

      <div>

        <h1 className="text-3xl font-bold tracking-tight">Registro de Medicamentos</h1>

        <p className="text-muted-foreground">Ver y gestionar entregas de medicamentos</p>

      </div>

      <Tabs defaultValue="delivered" className="space-y-6">

        <TabsList>

          <TabsTrigger value="delivered">Entregados</TabsTrigger>

          <TabsTrigger value="pending">Pendientes</TabsTrigger>

        </TabsList>

        <TabsContent value="delivered">

          <Card>

            <CardHeader>

              <CardTitle>Medicamentos Entregados</CardTitle>

              <CardDescription>Ver todos los medicamentos que han sido entregados a los pacientes</CardDescription>

            </CardHeader>

            <CardContent>

              <div className="flex items-center justify-between mb-6">

                <div className="relative w-full max-w-sm">

                  <Search className="absolute left-2.5 top-2.5 h-4 w-4 text-muted-foreground" />

                  <Input

                    type="search"

                    placeholder="Buscar medicamentos..."

                    className="pl-8"

                    value={deliveredSearchTerm}

                    onChange={(e) => setDeliveredSearchTerm(e.target.value)}

                  />

                </div>

                <Button>

                  <Printer className="mr-2 h-4 w-4" />

                  Imprimir Informe

                </Button>

              </div>

              <div className="rounded-md border">

                <Table>

                  <TableHeader>

                    <TableRow>

                      <TableHead>ID Medicamento</TableHead>

                      <TableHead>ID Paciente</TableHead>

                      <TableHead>Medicamento</TableHead>

                      <TableHead>Dosis</TableHead>

                      <TableHead>Frecuencia</TableHead>

                      <TableHead>Fecha de Entrega</TableHead>

                      <TableHead>Estado</TableHead>

                      <TableHead className="text-right">Acciones</TableHead>

                    </TableRow>

                  </TableHeader>

                  <TableBody>

                    {filteredDeliveredMedications.length > 0 ? (

                      filteredDeliveredMedications.map((med) => (

                        <TableRow key={med.id}>

                          <TableCell className="font-medium">{med.id}</TableCell>

                          <TableCell>{med.patientId}</TableCell>

                          <TableCell>{med.name}</TableCell>

                          <TableCell>{med.dosage}</TableCell>

                          <TableCell>{med.frequency}</TableCell>

                          <TableCell>{new Date(med.deliveryDate).toLocaleDateString()}</TableCell>

                          <TableCell>

                            <Badge className="bg-green-500">{med.status}</Badge>

                          </TableCell>

                          <TableCell className="text-right">

                            <DropdownMenu>

                              <DropdownMenuTrigger asChild>

                                <Button variant="ghost" size="icon">

                                  <MoreVertical className="h-4 w-4" />

                                  <span className="sr-only">Acciones</span>

                                </Button>

                              </DropdownMenuTrigger>

                              <DropdownMenuContent align="end">

                                <DropdownMenuItem>

                                  <Eye className="mr-2 h-4 w-4" />

                                  Ver Detalles

                                </DropdownMenuItem>

                                <DropdownMenuItem>

                                  <Download className="mr-2 h-4 w-4" />

                                  Descargar Registro

                                </DropdownMenuItem>

                                <DropdownMenuItem>

                                  <Printer className="mr-2 h-4 w-4" />

                                  Imprimir Registro

                                </DropdownMenuItem>

                              </DropdownMenuContent>

                            </DropdownMenu>

                          </TableCell>

                        </TableRow>

                      ))

                    ) : (

                      <TableRow>

                        <TableCell colSpan={8} className="h-24 text-center">

                          No se encontraron medicamentos.

                        </TableCell>

                      </TableRow>

                    )}

                  </TableBody>

                </Table>

              </div>

            </CardContent>

          </Card>

        </TabsContent>

        <TabsContent value="pending">

          <Card>

            <CardHeader>

              <CardTitle>Medicamentos Pendientes</CardTitle>

              <CardDescription>Ver todos los medicamentos que están pendientes de entrega</CardDescription>

            </CardHeader>

            <CardContent>

              <div className="flex items-center justify-between mb-6">

                <div className="relative w-full max-w-sm">

                  <Search className="absolute left-2.5 top-2.5 h-4 w-4 text-muted-foreground" />

                  <Input

                    type="search"

                    placeholder="Buscar medicamentos..."

                    className="pl-8"

                    value={pendingSearchTerm}

                    onChange={(e) => setPendingSearchTerm(e.target.value)}

                  />

                </div>

                <Button>

                  <CheckCircle className="mr-2 h-4 w-4" />

                  Marcar como Entregado

                </Button>

              </div>

              <div className="rounded-md border">

                <Table>

                  <TableHeader>

                    <TableRow>

                      <TableHead>ID Medicamento</TableHead>

                      <TableHead>ID Paciente</TableHead>

                      <TableHead>Medicamento</TableHead>

                      <TableHead>Dosis</TableHead>

                      <TableHead>Frecuencia</TableHead>

                      <TableHead>Fecha de Solicitud</TableHead>

                      <TableHead>Estado</TableHead>

                      <TableHead className="text-right">Acciones</TableHead>

                    </TableRow>

                  </TableHeader>

                  <TableBody>

                    {filteredPendingMedications.length > 0 ? (

                      filteredPendingMedications.map((med) => (

                        <TableRow key={med.id}>

                          <TableCell className="font-medium">{med.id}</TableCell>

                          <TableCell>{med.patientId}</TableCell>

                          <TableCell>{med.name}</TableCell>

                          <TableCell>{med.dosage}</TableCell>

                          <TableCell>{med.frequency}</TableCell>

                          <TableCell>{new Date(med.requestDate).toLocaleDateString()}</TableCell>

                          <TableCell>

                            <Badge variant="outline" className="bg-yellow-100 text-yellow-800 hover:bg-yellow-100">

                              {med.status}

                            </Badge>

                          </TableCell>

                          <TableCell className="text-right">

                            <DropdownMenu>

                              <DropdownMenuTrigger asChild>

                                <Button variant="ghost" size="icon">

                                  <MoreVertical className="h-4 w-4" />

                                  <span className="sr-only">Acciones</span>

                                </Button>

                              </DropdownMenuTrigger>

                              <DropdownMenuContent align="end">

                                <DropdownMenuItem>

                                  <Eye className="mr-2 h-4 w-4" />

                                  Ver Detalles

                                </DropdownMenuItem>

                                <DropdownMenuItem>

                                  <Pill className="mr-2 h-4 w-4" />

                                  Actualizar Estado

                                </DropdownMenuItem>

                                <DropdownMenuItem>

                                  <Printer className="mr-2 h-4 w-4" />

                                  Imprimir Registro

                                </DropdownMenuItem>

                              </DropdownMenuContent>

                            </DropdownMenu>

                          </TableCell>

                        </TableRow>

                      ))

                    ) : (

                      <TableRow>

                        <TableCell colSpan={8} className="h-24 text-center">

                          No se encontraron medicamentos pendientes.

                        </TableCell>

                      </TableRow>

                    )}

                  </TableBody>

                </Table>

              </div>

            </CardContent>

          </Card>

        </TabsContent>

      </Tabs>

    </div>

  )

}

Pending-documents/

"use client"

import type React from "react"

import { useState, useEffect, useRef } from "react"

import { z } from "zod"

import { useForm } from "react-hook-form"

import { zodResolver } from "@hookform/resolvers/zod"

import { Button } from "@/components/ui/button"

import { Form, FormControl, FormField, FormItem, FormMessage } from "@/components/ui/form"

import { Input } from "@/components/ui/input"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogDescription } from "@/components/ui/dialog"

import { Search, Upload, Camera, FileText, Download, Eye, Check, X } from "lucide-react"

import { Alert, AlertDescription, AlertTitle } from "@/components/ui/alert"

import { useAuth } from "@/app/context/auth-context"

import { useRouter } from "next/navigation"

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

// Datos simulados de pacientes (en una aplicación real, esto vendría de una API)

const patientData = [

  {

    cedula: "1098765432",

    nombre: "Maria Alejandra Rodriguez Gomez",

    direccion: "Calle 123 #45-67, Bogotá, Colombia",

    eps: "Sanitas EPS",

    msd: "MSD-001",

  },

  {

    cedula: "0987654321",

    nombre: "Carlos Andrés Martínez López",

    direccion: "Carrera 45 #12-34, Medellín, Colombia",

    eps: "Nueva EPS",

    msd: "MSD-002",

  },

]

// Documentos previamente almacenados

const previousDocuments = [

  {

    id: "DOC-001",

    type: "Autorización",

    uploadDate: "2023-05-15",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-002",

    type: "ID Document",

    uploadDate: "2023-05-15",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-003",

    type: "Medical Prescription",

    uploadDate: "2023-05-15",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-004",

    type: "MSD",

    uploadDate: "2023-05-15",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-005",

    type: "Patient Signature",

    uploadDate: "2023-05-15",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

]

const formSchema = z.object({

  searchType: z.enum(["msd", "cedula"]),

  searchTerm: z.string().min(1, "Por favor, introduce un término de búsqueda"),

})

export default function PendingDocumentsPage() {

  const { user } = useAuth()

  const router = useRouter()

  const [isSubmitting, setIsSubmitting] = useState(false)

  const [isSuccess, setIsSuccess] = useState(false)

  const [patientFound, setPatientFound] = useState<(typeof patientData)[0] | null>(null)

  const [isCameraOpen, setIsCameraOpen] = useState(false)

  const [currentDocumentType, setCurrentDocumentType] = useState<string>("")

  const [capturedImages, setCapturedImages] = useState<Record<string, string>>({})

  const [uploadedFiles, setUploadedFiles] = useState<Record<string, File>>({})

  const [stream, setStream] = useState<MediaStream | null>(null)

  const [isPreviewOpen, setIsPreviewOpen] = useState(false)

  const [previewImage, setPreviewImage] = useState<string | null>(null)

  const [isDocumentPreviewOpen, setIsDocumentPreviewOpen] = useState(false)

  const [currentPreviewDocument, setCurrentPreviewDocument] = useState<(typeof previousDocuments)[0] | null>(null)

  const videoRef = useRef<HTMLVideoElement>(null)

  const canvasRef = useRef<HTMLCanvasElement>(null)

  const form = useForm<z.infer<typeof formSchema>>({

    resolver: zodResolver(formSchema),

    defaultValues: {

      searchType: "msd",

      searchTerm: "",

    },

  })

  // Redirigir si el usuario es externo

  useEffect(() => {

    if (user?.role === "externo") {

      router.push("/dashboard")

    }

  }, [user, router])

  // Limpiar el stream de la cámara cuando se cierra

  useEffect(() => {

    return () => {

      if (stream) {

        stream.getTracks().forEach((track) => track.stop())

      }

    }

  }, [stream])

  // Si el usuario es externo, mostrar mensaje de acceso denegado

  if (user?.role === "externo") {

    return (

      <div className="flex h-full items-center justify-center">

        <div className="text-center">

          <h1 className="text-2xl font-bold">Acceso Denegado</h1>

          <p className="text-muted-foreground">No tienes permiso para acceder a esta página.</p>

        </div>

      </div>

    )

  }

  // Función para buscar paciente

  const searchPatient = (data: z.infer<typeof formSchema>) => {

    let found = null

    if (data.searchType === "msd") {

      found = patientData.find((patient) => patient.msd === data.searchTerm)

    } else {

      found = patientData.find((patient) => patient.cedula === data.searchTerm)

    }

    if (found) {

      setPatientFound(found)

    } else {

      setPatientFound(null)

      alert("No se encontró ningún paciente con los datos proporcionados.")

    }

  }

  // Función para abrir la cámara

  const openCamera = (documentType: string) => {

    setCurrentDocumentType(documentType)

    navigator.mediaDevices

      .getUserMedia({ video: true })

      .then((mediaStream) => {

        setStream(mediaStream)

        if (videoRef.current) {

          videoRef.current.srcObject = mediaStream

        }

        setIsCameraOpen(true)

      })

      .catch((error) => {

        console.error("Error al acceder a la cámara:", error)

        alert("No se pudo acceder a la cámara. Por favor, verifica los permisos.")

      })

  }

  // Función para capturar imagen

  const captureImage = () => {

    if (videoRef.current && canvasRef.current && currentDocumentType) {

      const video = videoRef.current

      const canvas = canvasRef.current

      const context = canvas.getContext("2d")

      if (context) {

        canvas.width = video.videoWidth

        canvas.height = video.videoHeight

        context.drawImage(video, 0, 0, canvas.width, canvas.height)

        const imageDataUrl = canvas.toDataURL("image/png")

        setCapturedImages((prev) => ({

          ...prev,

          [currentDocumentType]: imageDataUrl,

        }))

        // Detener la cámara

        if (stream) {

          stream.getTracks().forEach((track) => track.stop())

        }

        setIsCameraOpen(false)

      }

    }

  }

  // Función para cerrar la cámara

  const closeCamera = () => {

    if (stream) {

      stream.getTracks().forEach((track) => track.stop())

    }

    setIsCameraOpen(false)

  }

  // Función para manejar la subida de archivos

  const handleFileUpload = (documentType: string, e: React.ChangeEvent<HTMLInputElement>) => {

    if (e.target.files && e.target.files.length > 0) {

      const file = e.target.files[0]

      setUploadedFiles((prev) => ({

        ...prev,

        [documentType]: file,

      }))

      // Crear URL para previsualización

      const reader = new FileReader()

      reader.onload = (event) => {

        if (event.target && event.target.result) {

          setCapturedImages((prev) => ({

            ...prev,

            [documentType]: event.target?.result as string,

          }))

        }

      }

      reader.readAsDataURL(file)

    }

  }

  // Función para previsualizar un documento almacenado

  const previewStoredDocument = (document: (typeof previousDocuments)[0]) => {

    setCurrentPreviewDocument(document)

    setIsDocumentPreviewOpen(true)

  }

  // Función para previsualizar un documento capturado

  const previewCapturedDocument = (imageUrl: string) => {

    setPreviewImage(imageUrl)

    setIsPreviewOpen(true)

  }

  // Función para guardar todos los documentos

  const saveAllDocuments = () => {

    setIsSubmitting(true)

    // Simular envío a API

    setTimeout(() => {

      setIsSuccess(true)

      setIsSubmitting(false)

      // Resetear después de 3 segundos

      setTimeout(() => {

        setIsSuccess(false)

        setPatientFound(null)

        setCapturedImages({})

        setUploadedFiles({})

        form.reset()

      }, 3000)

    }, 1500)

  }

  // Verificar si hay documentos para guardar

  const hasDocumentsToSave = Object.keys(capturedImages).length > 0 || Object.keys(uploadedFiles).length > 0

  return (

    <div className="space-y-6">

      <div>

        <h1 className="text-3xl font-bold tracking-tight">Capturar Documentos Pendientes</h1>

        <p className="text-muted-foreground">

          Almacena documentos adicionales para entregas de medicamentos pendientes

        </p>

      </div>

      {isSuccess && (

        <Alert className="bg-green-50">

          <Check className="h-4 w-4 text-green-600" />

          <AlertTitle>Éxito</AlertTitle>

          <AlertDescription>Los documentos han sido guardados correctamente.</AlertDescription>

        </Alert>

      )}

      <Card>

        <CardHeader>

          <CardTitle>Buscar Paciente</CardTitle>

          <CardDescription>Introduce el número MSD o cédula para buscar la información del paciente</CardDescription>

        </CardHeader>

        <CardContent>

          <Form {...form}>

            <form onSubmit={form.handleSubmit(searchPatient)} className="space-y-4">

              <div className="flex flex-col space-y-4">

                <div className="flex space-x-4">

                  <FormField

                    control={form.control}

                    name="searchType"

                    render={({ field }) => (

                      <FormItem className="space-y-0 flex items-center space-x-4">

                        <div className="flex items-center space-x-2">

                          <input

                            type="radio"

                            id="msd"

                            value="msd"

                            checked={field.value === "msd"}

                            onChange={() => field.onChange("msd")}

                            className="mr-1"

                          />

                          <label htmlFor="msd">MSD Number</label>

                        </div>

                        <div className="flex items-center space-x-2">

                          <input

                            type="radio"

                            id="cedula"

                            value="cedula"

                            checked={field.value === "cedula"}

                            onChange={() => field.onChange("cedula")}

                            className="mr-1"

                          />

                          <label htmlFor="cedula">ID Card (Cédula)</label>

                        </div>

                      </FormItem>

                    )}

                  />

                </div>

                <div className="flex space-x-4">

                  <FormField

                    control={form.control}

                    name="searchTerm"

                    render={({ field }) => (

                      <FormItem className="flex-1">

                        <FormControl>

                          <div className="relative">

                            <Search className="absolute left-2.5 top-2.5 h-4 w-4 text-muted-foreground" />

                            <Input

                              placeholder={`Introduce ${form.getValues().searchType === "msd" ? "MSD" : "cédula"}`}

                              className="pl-8"

                              {...field}

                            />

                          </div>

                        </FormControl>

                        <FormMessage />

                      </FormItem>

                    )}

                  />

                  <Button type="submit">

                    <Search className="mr-2 h-4 w-4" />

                    Buscar Paciente

                  </Button>

                </div>

              </div>

            </form>

          </Form>

        </CardContent>

      </Card>

      {patientFound && (

        <>

          <Card>

            <CardHeader>

              <CardTitle>Información del Paciente</CardTitle>

            </CardHeader>

            <CardContent>

              <div className="grid grid-cols-3 gap-6">

                <div>

                  <h3 className="text-sm font-medium text-gray-500">Nombre Completo</h3>

                  <p className="mt-1 text-lg font-medium">{patientFound.nombre}</p>

                </div>

                <div>

                  <h3 className="text-sm font-medium text-gray-500">Número de Cédula</h3>

                  <p className="mt-1 text-lg font-medium">{patientFound.cedula}</p>

                </div>

                <div>

                  <h3 className="text-sm font-medium text-gray-500">EPS</h3>

                  <p className="mt-1 text-lg font-medium">{patientFound.eps}</p>

                </div>

              </div>

            </CardContent>

          </Card>

          <Card>

            <CardHeader>

              <CardTitle>Documentos Previamente Almacenados</CardTitle>

              <CardDescription>Estos documentos ya han sido almacenados en el sistema</CardDescription>

            </CardHeader>

            <CardContent>

              <Table>

                <TableHeader>

                  <TableRow>

                    <TableHead>Tipo de Documento</TableHead>

                    <TableHead>Fecha de Carga</TableHead>

                    <TableHead className="text-right">Acciones</TableHead>

                  </TableRow>

                </TableHeader>

                <TableBody>

                  {previousDocuments.map((doc) => (

                    <TableRow key={doc.id}>

                      <TableCell className="font-medium">{doc.type}</TableCell>

                      <TableCell>{new Date(doc.uploadDate).toLocaleDateString()}</TableCell>

                      <TableCell className="text-right">

                        <Button variant="ghost" size="sm" onClick={() => previewStoredDocument(doc)}>

                          <Eye className="h-4 w-4 mr-1" />

                          Ver

                        </Button>

                      </TableCell>

                    </TableRow>

                  ))}

                </TableBody>

              </Table>

            </CardContent>

          </Card>

          <Card>

            <CardHeader>

              <CardTitle>Documentación de Entrega de Medicamentos</CardTitle>

              <CardDescription>Sube documentos adicionales requeridos para la entrega de medicamentos</CardDescription>

            </CardHeader>

            <CardContent className="space-y-6">

              {/\* Formulario de Confirmación de Entrega \*/}

              <div className="border-b pb-6">

                <div className="flex justify-between items-start mb-2">

                  <div>

                    <h3 className="text-lg font-medium">Formulario de Confirmación de Entrega</h3>

                    <p className="text-sm text-gray-500">

                      Sube o captura una imagen clara del formulario de confirmación de entrega

                    </p>

                  </div>

                  <div className="flex space-x-2">

                    <Button variant="outline" onClick={() => document.getElementById("confirmacion-upload")?.click()}>

                      <Upload className="mr-2 h-4 w-4" />

                      Subir Archivo

                    </Button>

                    <input

                      id="confirmacion-upload"

                      type="file"

                      className="hidden"

                      accept="image/\*,.pdf"

                      onChange={(e) => handleFileUpload("confirmacion", e)}

                    />

                    <Button variant="outline" onClick={() => openCamera("confirmacion")}>

                      <Camera className="mr-2 h-4 w-4" />

                      Tomar Foto

                    </Button>

                  </div>

                </div>

                {capturedImages["confirmacion"] && (

                  <div className="mt-4 relative bg-gray-50 p-4 rounded-md">

                    <div className="flex items-center justify-between">

                      <div className="flex items-center">

                        <FileText className="h-8 w-8 text-blue-500 mr-2" />

                        <div>

                          <p className="font-medium">Formulario de Confirmación</p>

                          <p className="text-sm text-gray-500">Documento capturado</p>

                        </div>

                      </div>

                      <div className="flex space-x-2">

                        <Button

                          variant="ghost"

                          size="sm"

                          onClick={() => previewCapturedDocument(capturedImages["confirmacion"])}

                        >

                          <Eye className="h-4 w-4 mr-1" />

                          Ver

                        </Button>

                        <Button

                          variant="ghost"

                          size="sm"

                          className="text-red-500 hover:text-red-700"

                          onClick={() => {

                            setCapturedImages((prev) => {

                              const newImages = { ...prev }

                              delete newImages["confirmacion"]

                              return newImages

                            })

                          }}

                        >

                          <X className="h-4 w-4 mr-1" />

                          Eliminar

                        </Button>

                      </div>

                    </div>

                  </div>

                )}

              </div>

              {/\* Firma de Recibo de Medicamento \*/}

              <div className="border-b pb-6">

                <div className="flex justify-between items-start mb-2">

                  <div>

                    <h3 className="text-lg font-medium">Firma de Recibo de Medicamento</h3>

                    <p className="text-sm text-gray-500">

                      Sube o captura una imagen clara de la firma de recibo de medicamento

                    </p>

                  </div>

                  <div className="flex space-x-2">

                    <Button variant="outline" onClick={() => document.getElementById("firma-upload")?.click()}>

                      <Upload className="mr-2 h-4 w-4" />

                      Subir Archivo

                    </Button>

                    <input

                      id="firma-upload"

                      type="file"

                      className="hidden"

                      accept="image/\*,.pdf"

                      onChange={(e) => handleFileUpload("firma", e)}

                    />

                    <Button variant="outline" onClick={() => openCamera("firma")}>

                      <Camera className="mr-2 h-4 w-4" />

                      Tomar Foto

                    </Button>

                  </div>

                </div>

                {capturedImages["firma"] && (

                  <div className="mt-4 relative bg-gray-50 p-4 rounded-md">

                    <div className="flex items-center justify-between">

                      <div className="flex items-center">

                        <FileText className="h-8 w-8 text-blue-500 mr-2" />

                        <div>

                          <p className="font-medium">Firma de Recibo</p>

                          <p className="text-sm text-gray-500">Documento capturado</p>

                        </div>

                      </div>

                      <div className="flex space-x-2">

                        <Button

                          variant="ghost"

                          size="sm"

                          onClick={() => previewCapturedDocument(capturedImages["firma"])}

                        >

                          <Eye className="h-4 w-4 mr-1" />

                          Ver

                        </Button>

                        <Button

                          variant="ghost"

                          size="sm"

                          className="text-red-500 hover:text-red-700"

                          onClick={() => {

                            setCapturedImages((prev) => {

                              const newImages = { ...prev }

                              delete newImages["firma"]

                              return newImages

                            })

                          }}

                        >

                          <X className="h-4 w-4 mr-1" />

                          Eliminar

                        </Button>

                      </div>

                    </div>

                  </div>

                )}

              </div>

              {/\* Foto de Identificación del Paciente \*/}

              <div className="pb-6">

                <div className="flex justify-between items-start mb-2">

                  <div>

                    <h3 className="text-lg font-medium">Foto de Identificación del Paciente</h3>

                    <p className="text-sm text-gray-500">

                      Sube o captura una imagen clara de la foto de identificación del paciente

                    </p>

                  </div>

                  <div className="flex space-x-2">

                    <Button variant="outline" onClick={() => document.getElementById("foto-upload")?.click()}>

                      <Upload className="mr-2 h-4 w-4" />

                      Subir Archivo

                    </Button>

                    <input

                      id="foto-upload"

                      type="file"

                      className="hidden"

                      accept="image/\*,.pdf"

                      onChange={(e) => handleFileUpload("foto", e)}

                    />

                    <Button variant="outline" onClick={() => openCamera("foto")}>

                      <Camera className="mr-2 h-4 w-4" />

                      Tomar Foto

                    </Button>

                  </div>

                </div>

                {capturedImages["foto"] && (

                  <div className="mt-4 relative bg-gray-50 p-4 rounded-md">

                    <div className="flex items-center justify-between">

                      <div className="flex items-center">

                        <FileText className="h-8 w-8 text-blue-500 mr-2" />

                        <div>

                          <p className="font-medium">Foto de Identificación</p>

                          <p className="text-sm text-gray-500">Documento capturado</p>

                        </div>

                      </div>

                      <div className="flex space-x-2">

                        <Button

                          variant="ghost"

                          size="sm"

                          onClick={() => previewCapturedDocument(capturedImages["foto"])}

                        >

                          <Eye className="h-4 w-4 mr-1" />

                          Ver

                        </Button>

                        <Button

                          variant="ghost"

                          size="sm"

                          className="text-red-500 hover:text-red-700"

                          onClick={() => {

                            setCapturedImages((prev) => {

                              const newImages = { ...prev }

                              delete newImages["foto"]

                              return newImages

                            })

                          }}

                        >

                          <X className="h-4 w-4 mr-1" />

                          Eliminar

                        </Button>

                      </div>

                    </div>

                  </div>

                )}

              </div>

              <div className="pt-4 flex justify-end">

                <Button

                  onClick={saveAllDocuments}

                  disabled={!hasDocumentsToSave || isSubmitting}

                  className="w-full md:w-auto"

                >

                  {isSubmitting ? "Guardando..." : "Guardar Documentos"}

                </Button>

              </div>

            </CardContent>

          </Card>

        </>

      )}

      {/\* Diálogo para la cámara \*/}

      <Dialog open={isCameraOpen} onOpenChange={setIsCameraOpen}>

        <DialogContent className="sm:max-w-[600px]">

          <DialogHeader>

            <DialogTitle>Capturar Documento</DialogTitle>

            <DialogDescription>Posiciona el documento frente a la cámara y haz clic en "Capturar"</DialogDescription>

          </DialogHeader>

          <div className="flex flex-col items-center space-y-4">

            <div className="relative w-full border rounded-lg overflow-hidden">

              <video ref={videoRef} autoPlay playsInline className="w-full h-auto" />

            </div>

            <div className="flex space-x-4">

              <Button onClick={captureImage}>

                <Camera className="mr-2 h-4 w-4" />

                Capturar

              </Button>

              <Button variant="outline" onClick={closeCamera}>

                Cancelar

              </Button>

            </div>

          </div>

          {/\* Canvas oculto para capturar la imagen \*/}

          <canvas ref={canvasRef} className="hidden" />

        </DialogContent>

      </Dialog>

      {/\* Diálogo para previsualización de documentos capturados \*/}

      <Dialog open={isPreviewOpen} onOpenChange={setIsPreviewOpen}>

        <DialogContent className="sm:max-w-[700px]">

          <DialogHeader>

            <DialogTitle>Previsualización de Documento</DialogTitle>

          </DialogHeader>

          <div className="flex justify-center">

            <img

              src={previewImage || "/placeholder.svg"}

              alt="Documento"

              className="max-h-[500px] object-contain border rounded-lg"

            />

          </div>

          <div className="flex justify-end space-x-2">

            <Button variant="outline" onClick={() => setIsPreviewOpen(false)}>

              Cerrar

            </Button>

            {previewImage && (

              <Button onClick={() => window.open(previewImage, "\_blank")}>

                <Download className="mr-2 h-4 w-4" />

                Descargar

              </Button>

            )}

          </div>

        </DialogContent>

      </Dialog>

      {/\* Diálogo para previsualización de documentos almacenados \*/}

      <Dialog open={isDocumentPreviewOpen} onOpenChange={setIsDocumentPreviewOpen}>

        <DialogContent className="sm:max-w-[700px]">

          <DialogHeader>

            <DialogTitle>{currentPreviewDocument?.type}</DialogTitle>

            <DialogDescription>

              Subido el{" "}

              {currentPreviewDocument?.uploadDate && new Date(currentPreviewDocument.uploadDate).toLocaleDateString()}

            </DialogDescription>

          </DialogHeader>

          <div className="flex justify-center">

            <img

              src={currentPreviewDocument?.fileUrl || "/placeholder.svg"}

              alt="Documento"

              className="max-h-[500px] object-contain border rounded-lg"

            />

          </div>

          <div className="flex justify-end space-x-2">

            <Button variant="outline" onClick={() => setIsDocumentPreviewOpen(false)}>

              Cerrar

            </Button>

            <Button onClick={() => window.open(currentPreviewDocument?.fileUrl, "\_blank")}>

              <Download className="mr-2 h-4 w-4" />

              Descargar

            </Button>

          </div>

        </DialogContent>

      </Dialog>

    </div>

  )

}

Profile/

"use client"

import { useState } from "react"

import { z } from "zod"

import { useForm } from "react-hook-form"

import { zodResolver } from "@hookform/resolvers/zod"

import { Button } from "@/components/ui/button"

import { Form, FormControl, FormDescription, FormField, FormItem, FormLabel, FormMessage } from "@/components/ui/form"

import { Input } from "@/components/ui/input"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"

import { Avatar, AvatarFallback } from "@/components/ui/avatar"

import { useAuth } from "@/app/context/auth-context"

import { Check, AlertTriangle } from "lucide-react"

import { Alert, AlertDescription, AlertTitle } from "@/components/ui/alert"

const profileFormSchema = z.object({

  name: z.string().min(2, { message: "El nombre debe tener al menos 2 caracteres" }),

  email: z.string().email({ message: "Por favor, introduce una dirección de correo electrónico válida" }),

  position: z.string().optional(),

})

const passwordFormSchema = z

  .object({

    currentPassword: z.string().min(6, { message: "La contraseña debe tener al menos 6 caracteres" }),

    newPassword: z.string().min(6, { message: "La contraseña debe tener al menos 6 caracteres" }),

    confirmPassword: z.string().min(6, { message: "La contraseña debe tener al menos 6 caracteres" }),

  })

  .refine((data) => data.newPassword === data.confirmPassword, {

    message: "Las contraseñas no coinciden",

    path: ["confirmPassword"],

  })

export default function ProfilePage() {

  const { user } = useAuth()

  const [isProfileSubmitting, setIsProfileSubmitting] = useState(false)

  const [isPasswordSubmitting, setIsPasswordSubmitting] = useState(false)

  const [profileSuccess, setProfileSuccess] = useState(false)

  const [passwordSuccess, setPasswordSuccess] = useState(false)

  const profileForm = useForm<z.infer<typeof profileFormSchema>>({

    resolver: zodResolver(profileFormSchema),

    defaultValues: {

      name: user?.name || "",

      email: user?.email || "",

      position: user?.position || "",

    },

  })

  const passwordForm = useForm<z.infer<typeof passwordFormSchema>>({

    resolver: zodResolver(passwordFormSchema),

    defaultValues: {

      currentPassword: "",

      newPassword: "",

      confirmPassword: "",

    },

  })

  async function onProfileSubmit(values: z.infer<typeof profileFormSchema>) {

    setIsProfileSubmitting(true)

    try {

      // Simular llamada a API

      await new Promise((resolve) => setTimeout(resolve, 1000))

      console.log(values)

      setProfileSuccess(true)

      setTimeout(() => {

        setProfileSuccess(false)

      }, 3000)

    } catch (error) {

      console.error(error)

    } finally {

      setIsProfileSubmitting(false)

    }

  }

  async function onPasswordSubmit(values: z.infer<typeof passwordFormSchema>) {

    setIsPasswordSubmitting(true)

    try {

      // Simular llamada a API

      await new Promise((resolve) => setTimeout(resolve, 1000))

      console.log(values)

      setPasswordSuccess(true)

      setTimeout(() => {

        setPasswordSuccess(false)

        passwordForm.reset()

      }, 3000)

    } catch (error) {

      console.error(error)

    } finally {

      setIsPasswordSubmitting(false)

    }

  }

  // Verificar si el usuario es externo o regular (no admin)

  const isReadOnly = user?.role === "externo" || user?.role === "user"

  return (

    <div className="space-y-6">

      <div>

        <h1 className="text-3xl font-bold tracking-tight">Perfil</h1>

        <p className="text-muted-foreground">

          {isReadOnly

            ? "Visualiza la información de tu cuenta"

            : "Gestiona la configuración de tu cuenta y preferencias"}

        </p>

      </div>

      {isReadOnly && (

        <Alert className="bg-yellow-50 border-yellow-200">

          <AlertTriangle className="h-4 w-4 text-yellow-600" />

          <AlertTitle>Acceso Limitado</AlertTitle>

          <AlertDescription>

            Solo los administradores pueden modificar la información del perfil. Si necesitas realizar cambios, contacta

            a un administrador.

          </AlertDescription>

        </Alert>

      )}

      <Tabs defaultValue="general" className="space-y-6">

        <TabsList>

          <TabsTrigger value="general">General</TabsTrigger>

          {!isReadOnly && <TabsTrigger value="password">Contraseña</TabsTrigger>}

        </TabsList>

        <TabsContent value="general">

          <Card>

            <CardHeader>

              <CardTitle>Información General</CardTitle>

              <CardDescription>

                {isReadOnly ? "Visualiza tu información personal" : "Actualiza tu información personal"}

              </CardDescription>

            </CardHeader>

            <CardContent className="space-y-6">

              <div className="flex items-center gap-6">

                <Avatar className="h-20 w-20">

                  <AvatarFallback className="text-2xl">{user?.name?.charAt(0) || "U"}</AvatarFallback>

                </Avatar>

                <div>

                  <h3 className="text-xl font-medium">{user?.name}</h3>

                  <p className="text-sm text-muted-foreground">{user?.position}</p>

                </div>

              </div>

              {profileSuccess && (

                <Alert className="bg-green-50">

                  <Check className="h-4 w-4 text-green-600" />

                  <AlertTitle>Éxito</AlertTitle>

                  <AlertDescription>Tu perfil ha sido actualizado correctamente.</AlertDescription>

                </Alert>

              )}

              <Form {...profileForm}>

                <form onSubmit={profileForm.handleSubmit(onProfileSubmit)} className="space-y-6">

                  <FormField

                    control={profileForm.control}

                    name="name"

                    render={({ field }) => (

                      <FormItem>

                        <FormLabel>Nombre</FormLabel>

                        <FormControl>

                          <Input placeholder="Tu nombre" {...field} disabled={isReadOnly} readOnly={isReadOnly} />

                        </FormControl>

                        <FormMessage />

                      </FormItem>

                    )}

                  />

                  <FormField

                    control={profileForm.control}

                    name="email"

                    render={({ field }) => (

                      <FormItem>

                        <FormLabel>Correo electrónico</FormLabel>

                        <FormControl>

                          <Input

                            placeholder="Tu correo electrónico"

                            {...field}

                            disabled={isReadOnly}

                            readOnly={isReadOnly}

                          />

                        </FormControl>

                        <FormMessage />

                      </FormItem>

                    )}

                  />

                  <FormField

                    control={profileForm.control}

                    name="position"

                    render={({ field }) => (

                      <FormItem>

                        <FormLabel>Cargo</FormLabel>

                        <FormControl>

                          <Input placeholder="Tu cargo" {...field} disabled={isReadOnly} readOnly={isReadOnly} />

                        </FormControl>

                        <FormDescription>Tu rol o cargo en la organización</FormDescription>

                        <FormMessage />

                      </FormItem>

                    )}

                  />

                  {!isReadOnly && (

                    <Button type="submit" disabled={isProfileSubmitting}>

                      {isProfileSubmitting ? "Guardando..." : "Guardar Cambios"}

                    </Button>

                  )}

                </form>

              </Form>

            </CardContent>

          </Card>

        </TabsContent>

        {!isReadOnly && (

          <TabsContent value="password">

            <Card>

              <CardHeader>

                <CardTitle>Cambiar Contraseña</CardTitle>

                <CardDescription>Actualiza tu contraseña para mantener tu cuenta segura</CardDescription>

              </CardHeader>

              <CardContent>

                {passwordSuccess && (

                  <Alert className="mb-6 bg-green-50">

                    <Check className="h-4 w-4 text-green-600" />

                    <AlertTitle>Éxito</AlertTitle>

                    <AlertDescription>Tu contraseña ha sido actualizada correctamente.</AlertDescription>

                  </Alert>

                )}

                <Form {...passwordForm}>

                  <form onSubmit={passwordForm.handleSubmit(onPasswordSubmit)} className="space-y-6">

                    <FormField

                      control={passwordForm.control}

                      name="currentPassword"

                      render={({ field }) => (

                        <FormItem>

                          <FormLabel>Contraseña Actual</FormLabel>

                          <FormControl>

                            <Input type="password" placeholder="Introduce la contraseña actual" {...field} />

                          </FormControl>

                          <FormMessage />

                        </FormItem>

                      )}

                    />

                    <FormField

                      control={passwordForm.control}

                      name="newPassword"

                      render={({ field }) => (

                        <FormItem>

                          <FormLabel>Nueva Contraseña</FormLabel>

                          <FormControl>

                            <Input type="password" placeholder="Introduce la nueva contraseña" {...field} />

                          </FormControl>

                          <FormMessage />

                        </FormItem>

                      )}

                    />

                    <FormField

                      control={passwordForm.control}

                      name="confirmPassword"

                      render={({ field }) => (

                        <FormItem>

                          <FormLabel>Confirmar Nueva Contraseña</FormLabel>

                          <FormControl>

                            <Input type="password" placeholder="Confirma la nueva contraseña" {...field} />

                          </FormControl>

                          <FormMessage />

                        </FormItem>

                      )}

                    />

                    <Button type="submit" disabled={isPasswordSubmitting}>

                      {isPasswordSubmitting ? "Actualizando..." : "Actualizar Contraseña"}

                    </Button>

                  </form>

                </Form>

              </CardContent>

            </Card>

          </TabsContent>

        )}

      </Tabs>

    </div>

  )

}

Saved-documents/

"use client"

import { useState, useRef } from "react"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { Button } from "@/components/ui/button"

import { Input } from "@/components/ui/input"

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

import {

  Dialog,

  DialogContent,

  DialogHeader,

  DialogTitle,

  DialogDescription,

  DialogFooter,

} from "@/components/ui/dialog"

import { Search, Download, Trash, Eye, Upload, Camera } from "lucide-react"

import { useAuth } from "@/app/context/auth-context"

// Datos de ejemplo para documentos

const documents = [

  {

    id: "DOC-001",

    patientId: "1098765432",

    patientName: "Maria Alejandra Rodriguez Gomez",

    type: "Autorización",

    date: "2023-05-15",

    status: "Completo",

    msd: "MSD-001",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-002",

    patientId: "0987654321",

    patientName: "Carlos Andrés Martínez López",

    type: "Cédula",

    date: "2023-05-14",

    status: "Pendiente",

    msd: "MSD-002",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-003",

    patientId: "5678901234",

    patientName: "Ana María Pérez Sánchez",

    type: "Fórmula Médica",

    date: "2023-05-13",

    status: "Completo",

    msd: "MSD-003",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

  {

    id: "DOC-004",

    patientId: "1098765432",

    patientName: "Maria Alejandra Rodriguez Gomez",

    type: "MSD",

    date: "2023-05-12",

    status: "Pendiente",

    msd: "MSD-001",

    fileUrl: "/placeholder.svg?height=600&width=400",

  },

]

export default function SavedDocumentsPage() {

  const { user } = useAuth()

  const [searchTerm, setSearchTerm] = useState("")

  const [searchType, setSearchType] = useState<"cedula" | "msd">("cedula")

  const [filteredDocuments, setFilteredDocuments] = useState(documents)

  const [isPreviewOpen, setIsPreviewOpen] = useState(false)

  const [currentDocument, setCurrentDocument] = useState<(typeof documents)[0] | null>(null)

  const [isDeleteDialogOpen, setIsDeleteDialogOpen] = useState(false)

  const [isUploadDialogOpen, setIsUploadDialogOpen] = useState(false)

  const [isUploading, setIsUploading] = useState(false)

  const fileInputRef = useRef<HTMLInputElement>(null)

  const [isCameraOpen, setIsCameraOpen] = useState(false)

  const [stream, setStream] = useState<MediaStream | null>(null)

  const videoRef = useRef<HTMLVideoElement>(null)

  const canvasRef = useRef<HTMLCanvasElement>(null)

  const handleSearch = () => {

    if (!searchTerm) {

      setFilteredDocuments(documents)

      return

    }

    const filtered = documents.filter((doc) =>

      searchType === "cedula" ? doc.patientId.includes(searchTerm) : doc.msd.includes(searchTerm),

    )

    setFilteredDocuments(filtered)

  }

  const openPreview = (document: (typeof documents)[0]) => {

    setCurrentDocument(document)

    setIsPreviewOpen(true)

  }

  const openDeleteDialog = (document: (typeof documents)[0]) => {

    setCurrentDocument(document)

    setIsDeleteDialogOpen(true)

  }

  const deleteDocument = () => {

    // En una aplicación real, aquí se haría una llamada a la API

    setFilteredDocuments((prev) => prev.filter((doc) => doc.id !== currentDocument?.id))

    setIsDeleteDialogOpen(false)

  }

  const openUploadDialog = (document: (typeof documents)[0]) => {

    setCurrentDocument(document)

    setIsUploadDialogOpen(true)

  }

  const handleFileUpload = () => {

    setIsUploading(true)

    // Simular una carga de archivo

    setTimeout(() => {

      setIsUploading(false)

      setIsUploadDialogOpen(false)

      // Aquí se actualizaría el documento en una aplicación real

    }, 1500)

  }

  // Función para abrir la cámara

  const openCamera = () => {

    navigator.mediaDevices

      .getUserMedia({ video: true })

      .then((mediaStream) => {

        setStream(mediaStream)

        if (videoRef.current) {

          videoRef.current.srcObject = mediaStream

        }

        setIsCameraOpen(true)

      })

      .catch((error) => {

        console.error("Error al acceder a la cámara:", error)

        alert("No se pudo acceder a la cámara. Por favor, verifica los permisos.")

      })

  }

  // Función para capturar imagen

  const captureImage = () => {

    if (videoRef.current && canvasRef.current) {

      const video = videoRef.current

      const canvas = canvasRef.current

      const context = canvas.getContext("2d")

      if (context) {

        canvas.width = video.videoWidth

        canvas.height = video.videoHeight

        context.drawImage(video, 0, 0, canvas.width, canvas.height)

        // Detener la cámara

        if (stream) {

          stream.getTracks().forEach((track) => track.stop())

        }

        setIsCameraOpen(false)

        setIsUploading(true)

        // Simular una carga de archivo

        setTimeout(() => {

          setIsUploading(false)

          setIsUploadDialogOpen(false)

          // Aquí se actualizaría el documento en una aplicación real

        }, 1500)

      }

    }

  }

  // Función para cerrar la cámara

  const closeCamera = () => {

    if (stream) {

      stream.getTracks().forEach((track) => track.stop())

    }

    setIsCameraOpen(false)

  }

  return (

    <div className="space-y-6">

      <div>

        <h1 className="text-3xl font-bold tracking-tight">Documentos Guardados</h1>

        <p className="text-muted-foreground">Consulta y gestiona los documentos de dispensación de medicamentos</p>

      </div>

      <Card>

        <CardHeader>

          <CardTitle>Buscar Documentos</CardTitle>

          <CardDescription>Busca documentos por cédula del paciente o número MSD</CardDescription>

        </CardHeader>

        <CardContent>

          <div className="flex items-end gap-4">

            <div className="flex-1">

              <div className="mb-2">

                <label className="text-sm font-medium">Buscar por:</label>

                <div className="flex mt-1 space-x-4">

                  <div className="flex items-center">

                    <input

                      type="radio"

                      id="cedula-search"

                      name="searchType"

                      className="mr-2"

                      checked={searchType === "cedula"}

                      onChange={() => setSearchType("cedula")}

                    />

                    <label htmlFor="cedula-search">Cédula</label>

                  </div>

                  <div className="flex items-center">

                    <input

                      type="radio"

                      id="msd-search"

                      name="searchType"

                      className="mr-2"

                      checked={searchType === "msd"}

                      onChange={() => setSearchType("msd")}

                    />

                    <label htmlFor="msd-search">MSD</label>

                  </div>

                </div>

              </div>

              <div className="relative">

                <Search className="absolute left-2.5 top-2.5 h-4 w-4 text-muted-foreground" />

                <Input

                  type="text"

                  placeholder={`Introduce ${searchType === "cedula" ? "cédula" : "MSD"}`}

                  className="pl-8"

                  value={searchTerm}

                  onChange={(e) => setSearchTerm(e.target.value)}

                />

              </div>

            </div>

            <Button onClick={handleSearch}>Buscar</Button>

          </div>

        </CardContent>

      </Card>

      <Card>

        <CardHeader>

          <CardTitle>Biblioteca de Documentos</CardTitle>

          <CardDescription>Navega por todos los documentos guardados en el sistema</CardDescription>

        </CardHeader>

        <CardContent>

          <div className="rounded-md border">

            <Table>

              <TableHeader>

                <TableRow>

                  <TableHead>ID Documento</TableHead>

                  <TableHead>Paciente</TableHead>

                  <TableHead>Cédula</TableHead>

                  <TableHead>Tipo</TableHead>

                  <TableHead>Fecha</TableHead>

                  <TableHead>MSD</TableHead>

                  <TableHead>Estado</TableHead>

                  <TableHead className="text-right">Acciones</TableHead>

                </TableRow>

              </TableHeader>

              <TableBody>

                {filteredDocuments.length > 0 ? (

                  filteredDocuments.map((doc) => (

                    <TableRow key={doc.id}>

                      <TableCell className="font-medium">{doc.id}</TableCell>

                      <TableCell>{doc.patientName}</TableCell>

                      <TableCell>{doc.patientId}</TableCell>

                      <TableCell>{doc.type}</TableCell>

                      <TableCell>{new Date(doc.date).toLocaleDateString()}</TableCell>

                      <TableCell>{doc.msd}</TableCell>

                      <TableCell>

                        <div

                          className={`px-2 py-1 rounded-full text-xs font-medium inline-block ${

                            doc.status === "Completo" ? "bg-green-100 text-green-800" : "bg-yellow-100 text-yellow-800"

                          }`}

                        >

                          {doc.status}

                        </div>

                      </TableCell>

                      <TableCell className="text-right">

                        <div className="flex justify-end space-x-2">

                          <Button variant="ghost" size="sm" onClick={() => openPreview(doc)}>

                            <Eye className="h-4 w-4" />

                          </Button>

                          <Button variant="ghost" size="sm" onClick={() => window.open(doc.fileUrl, "\_blank")}>

                            <Download className="h-4 w-4" />

                          </Button>

                          {user?.role !== "externo" && (

                            <>

                              <Button

                                variant="ghost"

                                size="sm"

                                onClick={() => openDeleteDialog(doc)}

                                className="text-red-500 hover:text-red-700"

                              >

                                <Trash className="h-4 w-4" />

                              </Button>

                              {doc.status === "Pendiente" && (

                                <Button

                                  variant="ghost"

                                  size="sm"

                                  onClick={() => openUploadDialog(doc)}

                                  className="text-blue-500 hover:text-blue-700"

                                >

                                  <Upload className="h-4 w-4" />

                                </Button>

                              )}

                            </>

                          )}

                        </div>

                      </TableCell>

                    </TableRow>

                  ))

                ) : (

                  <TableRow>

                    <TableCell colSpan={8} className="h-24 text-center">

                      No se encontraron documentos.

                    </TableCell>

                  </TableRow>

                )}

              </TableBody>

            </Table>

          </div>

        </CardContent>

      </Card>

      {/\* Diálogo de previsualización \*/}

      <Dialog open={isPreviewOpen} onOpenChange={setIsPreviewOpen}>

        <DialogContent className="sm:max-w-[700px]">

          <DialogHeader>

            <DialogTitle>Previsualización de Documento</DialogTitle>

            <DialogDescription>

              {currentDocument?.type} - {currentDocument?.patientName} - {currentDocument?.msd}

            </DialogDescription>

          </DialogHeader>

          <div className="flex justify-center">

            <img

              src={currentDocument?.fileUrl || "/placeholder.svg"}

              alt="Documento"

              className="max-h-[500px] object-contain border rounded-lg"

            />

          </div>

          <DialogFooter>

            <Button variant="outline" onClick={() => setIsPreviewOpen(false)}>

              Cerrar

            </Button>

            <Button onClick={() => window.open(currentDocument?.fileUrl, "\_blank")}>

              <Download className="mr-2 h-4 w-4" />

              Descargar

            </Button>

          </DialogFooter>

        </DialogContent>

      </Dialog>

      {/\* Diálogo de eliminación \*/}

      <Dialog open={isDeleteDialogOpen} onOpenChange={setIsDeleteDialogOpen}>

        <DialogContent className="sm:max-w-[425px]">

          <DialogHeader>

            <DialogTitle>Eliminar Documento</DialogTitle>

            <DialogDescription>

              ¿Estás seguro de que quieres eliminar este documento? Esta acción no se puede deshacer.

            </DialogDescription>

          </DialogHeader>

          <div className="py-4">

            <p className="mb-2">

              <strong>Tipo:</strong> {currentDocument?.type}

            </p>

            <p className="mb-2">

              <strong>Paciente:</strong> {currentDocument?.patientName}

            </p>

            <p className="mb-2">

              <strong>Cédula:</strong> {currentDocument?.patientId}

            </p>

            <p>

              <strong>MSD:</strong> {currentDocument?.msd}

            </p>

          </div>

          <DialogFooter>

            <Button variant="outline" onClick={() => setIsDeleteDialogOpen(false)}>

              Cancelar

            </Button>

            <Button variant="destructive" onClick={deleteDocument}>

              Eliminar Documento

            </Button>

          </DialogFooter>

        </DialogContent>

      </Dialog>

      {/\* Diálogo de subida de documento pendiente \*/}

      <Dialog open={isUploadDialogOpen} onOpenChange={setIsUploadDialogOpen}>

        <DialogContent className="sm:max-w-[500px]">

          <DialogHeader>

            <DialogTitle>Subir Documento Pendiente</DialogTitle>

            <DialogDescription>

              Sube el documento pendiente para {currentDocument?.patientName} (MSD: {currentDocument?.msd})

            </DialogDescription>

          </DialogHeader>

          <div className="space-y-4 py-4">

            <div className="flex space-x-4">

              <Button

                variant="outline"

                className="flex-1"

                onClick={() => document.getElementById("upload-file")?.click()}

              >

                <Upload className="mr-2 h-4 w-4" />

                Subir Archivo

              </Button>

              <input id="upload-file" type="file" className="hidden" accept="image/\*,.pdf" ref={fileInputRef} />

              <Button variant="outline" className="flex-1" onClick={openCamera}>

                <Camera className="mr-2 h-4 w-4" />

                Tomar Foto

              </Button>

            </div>

          </div>

          <DialogFooter>

            <Button variant="outline" onClick={() => setIsUploadDialogOpen(false)}>

              Cancelar

            </Button>

            <Button onClick={handleFileUpload} disabled={isUploading}>

              {isUploading ? "Subiendo..." : "Subir Documento"}

            </Button>

          </DialogFooter>

        </DialogContent>

      </Dialog>

      {/\* Diálogo para la cámara \*/}

      <Dialog open={isCameraOpen} onOpenChange={setIsCameraOpen}>

        <DialogContent className="sm:max-w-[600px]">

          <DialogHeader>

            <DialogTitle>Capturar Documento</DialogTitle>

            <DialogDescription>Posiciona el documento frente a la cámara y haz clic en "Capturar"</DialogDescription>

          </DialogHeader>

          <div className="flex flex-col items-center space-y-4">

            <div className="relative w-full border rounded-lg overflow-hidden">

              <video ref={videoRef} autoPlay playsInline className="w-full h-auto" />

            </div>

            <div className="flex space-x-4">

              <Button onClick={captureImage}>

                <Camera className="mr-2 h-4 w-4" />

                Capturar

              </Button>

              <Button variant="outline" onClick={closeCamera}>

                Cancelar

              </Button>

            </div>

          </div>

          {/\* Canvas oculto para capturar la imagen \*/}

          <canvas ref={canvasRef} className="hidden" />

        </DialogContent>

      </Dialog>

    </div>

  )

}