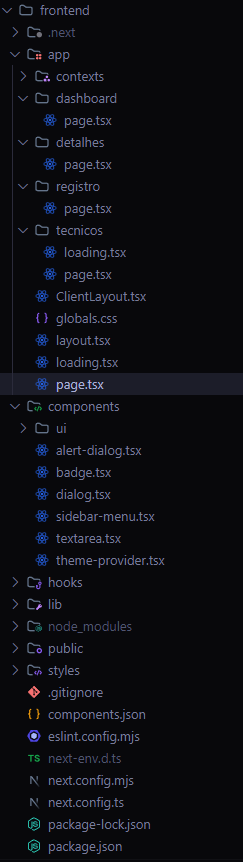
Web APP Agnus

Frontend estrutura:

frontend/app/contexts/ThemeContext.tsx :

"use client"

import type React from "react"

import { createContext, useContext, useEffect, useState } from "react"

type Theme = "dark" | "light"

interface ThemeContextType {

  theme: Theme

  toggleTheme: () => void

  isDark: boolean

  isLight: boolean

}

*const* ThemeContext = createContext<ThemeContextType | undefined>(undefined)

export function ThemeProvider({ children }: { children: React.ReactNode }) {

*const* [theme, setTheme] = useState<Theme>("dark")

  useEffect(() => {

*// Load theme from localStorage on mount*

*const* savedTheme = localStorage.getItem("agnus-theme") as Theme

    if (savedTheme) {

      setTheme(savedTheme)

    }

  }, [])

  useEffect(() => {

*// Apply theme to document*

    document.documentElement.setAttribute("data-theme", theme)

    localStorage.setItem("agnus-theme", theme)

  }, [theme])

*const* toggleTheme = () => {

    setTheme((prev) => (prev === "dark" ? "light" : "dark"))

  }

*return* (

    <ThemeContext.Provider

      value={{

        theme,

        toggleTheme,

        isDark: theme === "dark",

        isLight: theme === "light",

      }}

    >

      {children}

    </ThemeContext.Provider>

  )

}

export function useTheme() {

*const* context = useContext(ThemeContext)

  if (context === undefined) {

    throw new Error("useTheme must be used within a ThemeProvider")

  }

*return* context

}

frontend/app/dashboard/page.tsx :

"use client"

import { useState, useEffect } from "react"

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

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

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

import {

  Menu,

  Monitor,

  Clock,

  CheckCircle,

  AlertTriangle,

  Users,

  Building,

  TrendingUp,

  Activity,

  Zap,

  Home,

  ChevronRight,

  Filter,

  Calendar,

  RefreshCw,

  Download,

  Eye,

  User,

  Timer,

  Award,

  Target,

  Bell,

  Check,

  Trash2,

  ChevronLeft,

} from "lucide-react"

import { SidebarMenu } from "@/components/sidebar-menu"

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

import { toast, Toaster } from "sonner"

interface DashboardPageProps {

  onNavigate: (page: string, data?: any) => void

}

export default function DashboardPage({ onNavigate }: DashboardPageProps) {

*const* [sidebarOpen, setSidebarOpen] = useState(false)

*const* [currentTime, setCurrentTime] = useState(new Date())

*const* [dateFilter, setDateFilter] = useState("hoje")

*const* [statusFilter, setStatusFilter] = useState("todos")

*const* [tecnicoFilter, setTecnicoFilter] = useState("todos")

*const* [autoRefresh, setAutoRefresh] = useState(true)

*const* [lastRefresh, setLastRefresh] = useState(new Date())

*const* [currentTecnicoIndex, setCurrentTecnicoIndex] = useState(0)

*const* [isTransitioning, setIsTransitioning] = useState(false)

*// Notifications state*

*const* [notifications, setNotifications] = useState([

    {

      id: 1,

      title: "Novo computador adicionado",

      time: "Agora mesmo",

      read: false,

      type: "info",

    },

    {

      id: 2,

      title: "Manutenção concluída",

      time: "5 minutos atrás",

      read: false,

      type: "success",

    },

    {

      id: 3,

      title: "Alerta de sistema",

      time: "1 hora atrás",

      read: true,

      type: "warning",

    },

    {

      id: 4,

      title: "Técnico João Silva finalizou reparo",

      time: "2 horas atrás",

      read: false,

      type: "success",

    },

    {

      id: 5,

      title: "USF Centro com alta demanda",

      time: "3 horas atrás",

      read: true,

      type: "warning",

    },

  ])

*// Update time every second*

  useEffect(() => {

*const* timer = setInterval(() => {

      setCurrentTime(new Date())

    }, 1000)

*return* () => clearInterval(timer)

  }, [])

*// Auto refresh data every 30 seconds*

  useEffect(() => {

    if (autoRefresh) {

*const* refreshTimer = setInterval(() => {

        setLastRefresh(new Date())

        console.log("Refreshing dashboard data...")

      }, 30000)

*return* () => clearInterval(refreshTimer)

    }

  }, [autoRefresh])

*// Mock data*

*const* stats = {

    totalComputadores: 156,

    entrada: 12,

    prontos: 8,

    saida: 4,

    tecnicos: 5,

    unidades: 8,

    tempoMedioReparo: "2.5h",

    eficiencia: 87,

    satisfacao: 94,

    resolucaoRapida: 76,

  }

*const* recentActivityData = [

    {

      id: 1,

      tipo: "ENTRADA",

      tombamento: "202259131",

      unidade: "USF RIO DAS VELHAS II",

      tempo: "2 min",

      tecnico: "João Silva",

      ambiente: "Recepção",

    },

    {

      id: 2,

      tipo: "PRONTO",

      tombamento: "202259128",

      unidade: "USF QUADROS III",

      tempo: "5 min",

      tecnico: "Maria Santos",

      ambiente: "Consultório",

    },

    {

      id: 3,

      tipo: "SAÍDA",

      tombamento: "202259125",

      unidade: "USF VIETNÃ",

      tempo: "8 min",

      tecnico: "Pedro Costa",

      ambiente: "Enfermaria",

    },

    {

      id: 4,

      tipo: "ENTRADA",

      tombamento: "202259132",

      unidade: "USF CENTRO",

      tempo: "12 min",

      tecnico: "Ana Lima",

      ambiente: "Administração",

    },

    {

      id: 5,

      tipo: "PRONTO",

      tombamento: "202259133",

      unidade: "USF NORTE",

      tempo: "15 min",

      tecnico: "Carlos Oliveira",

      ambiente: "Laboratório",

    },

    {

      id: 6,

      tipo: "SAÍDA",

      tombamento: "202259134",

      unidade: "USF SUL",

      tempo: "18 min",

      tecnico: "João Silva",

      ambiente: "Farmácia",

    },

  ]

*// Filter for PRONTO items*

*const* prontosItems = recentActivityData.filter((item) => item.tipo === "PRONTO")

*const* [recentActivity, setRecentActivity] = useState(recentActivityData)

  useEffect(() => {

*let* filteredActivity = recentActivityData

    if (statusFilter !== "todos") {

      filteredActivity = filteredActivity.filter((item) => item.tipo.toLowerCase() === statusFilter)

    }

    if (tecnicoFilter !== "todos") {

      filteredActivity = filteredActivity.filter((item) => item.tecnico === tecnicoFilter)

    }

    setRecentActivity(filteredActivity)

  }, [statusFilter, tecnicoFilter])

*const* tecnicoStats = [

    { nome: "João Silva", tickets: 23, eficiencia: 91 },

    { nome: "Maria Santos", tickets: 19, eficiencia: 95 },

    { nome: "Pedro Costa", tickets: 17, eficiencia: 88 },

    { nome: "Ana Lima", tickets: 21, eficiencia: 90 },

    { nome: "Carlos Oliveira", tickets: 15, eficiencia: 93 },

    { nome: "Fernanda Costa", tickets: 18, eficiencia: 87 },

  ]

*const* unidadeStats = [

    { nome: "USF RIO DAS VELHAS II", entrada: 4, prontos: 2, saida: 1 },

    { nome: "USF QUADROS III", entrada: 2, prontos: 3, saida: 1 },

    { nome: "USF VIETNÃ", entrada: 3, prontos: 1, saida: 0 },

    { nome: "USF CENTRO", entrada: 2, prontos: 2, saida: 2 },

  ]

*// Carousel navigation with smooth transitions*

*const* nextTecnico = () => {

    if (isTransitioning) *return*

    setIsTransitioning(true)

    setCurrentTecnicoIndex((prev) => {

*const* maxIndex = Math.max(0, tecnicoStats.length - 4)

*return* prev >= maxIndex ? 0 : prev + 1

    })

    setTimeout(() => setIsTransitioning(false), 300)

  }

*const* prevTecnico = () => {

    if (isTransitioning) *return*

    setIsTransitioning(true)

    setCurrentTecnicoIndex((prev) => {

*const* maxIndex = Math.max(0, tecnicoStats.length - 4)

*return* prev <= 0 ? maxIndex : prev - 1

    })

    setTimeout(() => setIsTransitioning(false), 300)

  }

*const* visibleTecnicos = tecnicoStats.slice(currentTecnicoIndex, currentTecnicoIndex + 3)

*// Notification functions*

*const* markAllAsRead = () => {

    setNotifications((prev) => prev.map((notification) => ({ **...**notification, read: true })))

    toast.success("Todas as notificações foram marcadas como lidas", {

      duration: 3000,

      position: "top-right",

    })

  }

*const* clearAllNotifications = () => {

    setNotifications([])

    toast.success("Todas as notificações foram removidas", {

      duration: 3000,

      position: "top-right",

    })

  }

*const* markAsRead = (id: number) => {

    setNotifications((prev) =>

      prev.map((notification) => (notification.id === id ? { **...**notification, read: true } : notification)),

    )

  }

*// Get unread notifications count*

*const* unreadCount = notifications.filter((n) => !n.read).length

*const* getStatusIcon = (status: string) => {

    switch (status) {

      case "ENTRADA":

*return* <Clock className="h-3 w-3 text-red-400" />

      case "SAÍDA":

*return* <CheckCircle className="h-3 w-3 text-green-400" />

      case "PRONTO":

*return* <AlertTriangle className="h-3 w-3 text-blue-400" />

      default:

*return* <Clock className="h-3 w-3 text-gray-400" />

    }

  }

*const* getStatusColor = (status: string) => {

    switch (status) {

      case "ENTRADA":

*return* "text-red-400"

      case "SAÍDA":

*return* "text-green-400"

      case "PRONTO":

*return* "text-blue-400"

      default:

*return* "text-gray-400"

    }

  }

*const* getNotificationIcon = (type: string) => {

    switch (type) {

      case "success":

*return* <CheckCircle className="h-4 w-4 text-green-400" />

      case "warning":

*return* <AlertTriangle className="h-4 w-4 text-yellow-400" />

      case "info":

*return* <Bell className="h-4 w-4 text-blue-400" />

      default:

*return* <Bell className="h-4 w-4 text-gray-400" />

    }

  }

*return* (

    <div className="min-h-screen bg-[var(--agnus-bg-primary)] text-[var(--agnus-text-primary)] font-abel tv-display">

      {*/\* Toaster for notifications \*/*}

      <Toaster

        theme="dark"

        position="top-right"

        toastOptions={{

          style: {

            background: "#1C1815",

            border: "1px solid #E9A870",

            color: "#E0CAA5",

          },

        }}

      />

      {*/\* Sidebar Menu \*/*}

      <SidebarMenu

        isOpen={sidebarOpen}

        onClose={() => setSidebarOpen(false)}

        onNavigate={onNavigate}

        currentPage="dashboard"

      />

      {*/\* Header - Fixed/Sticky \*/*}

      <header className="sticky top-0 z-30 flex items-center justify-between px-6 py-4 bg-[#0C0C0C] border-b border-[#E9A870] shadow-lg backdrop-blur-sm">

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

          <Button

            variant="ghost"

            size="icon"

            className="text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

            onClick={() => setSidebarOpen(true)}

          >

            <Menu className="h-6 w-6" />

          </Button>

          <h1 className="text-[42px] font-abel text-[#E0CAA5] tracking-wider drop-shadow-sm">AGNUS</h1>

          {*/\* Breadcrumbs \*/*}

          <div className="flex items-center text-sm text-gray-400 ml-6">

            <Button

              variant="link"

              className="p-0 h-auto text-[#E0CAA5] hover:text-[#E9A870] transition-colors font-arial"

              onClick={() => onNavigate("home")}

            >

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

              <span>Início</span>

            </Button>

            <ChevronRight className="h-3 w-3 mx-2 text-gray-500" />

            <span className="text-gray-400 font-arial">Dashboard</span>

          </div>

        </div>

        {*/\* Header Controls \*/*}

        <div className="flex items-center gap-3">

          <Button

            variant="ghost"

            size="sm"

            onClick={() => setAutoRefresh(!autoRefresh)}

            className={`text-sm ${

              autoRefresh ? "text-green-400" : "text-gray-400"

            } hover:bg-[#E9A870]/20 transition-all duration-200`}

          >

            <RefreshCw className={`h-4 w-4 mr-1 ${autoRefresh ? "animate-spin" : ""}`} />

            Auto

          </Button>

          <Button

            variant="ghost"

            size="sm"

            className="text-[#E0CAA5] hover:bg-[#E9A870]/20 transition-all duration-200"

          >

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

            Export

          </Button>

          {*/\* Notifications Dropdown \*/*}

          <DropdownMenu>

            <DropdownMenuTrigger asChild>

              <Button

                variant="ghost"

                size="icon"

                className="relative text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

              >

                <Bell className="h-5 w-5" />

                {unreadCount > 0 && (

                  <span className="absolute -top-1 -right-1 w-5 h-5 bg-gradient-to-r from-red-500 to-red-600 rounded-full text-sm font-bold flex items-center justify-center animate-pulse">

                    {unreadCount}

                  </span>

                )}

              </Button>

            </DropdownMenuTrigger>

            <DropdownMenuContent

              align="end"

              className="w-80 bg-gradient-to-b from-[#1C1815] to-[#1C1815] border-[#3D3024] text-white shadow-xl backdrop-blur-sm animate-in slide-in-from-top-2 duration-200 max-h-96 overflow-y-auto custom-scroll"

            >

              <div className="flex items-center justify-between px-4 py-3 border-b border-[#3D3024]">

                <span className="font-medium font-arial">Notificações</span>

                {unreadCount > 0 && (

                  <span className="text-sm bg-gradient-to-r from-[#E9A870] to-[#A8784F] text-black px-2 py-1 rounded-full font-arial">

                    {unreadCount} novas

                  </span>

                )}

              </div>

              {*/\* Action Buttons - Icons Only \*/*}

              {notifications.length > 0 && (

                <div className="flex justify-center gap-4 px-4 py-3 border-b border-[#3D3024]">

                  <Button

                    variant="ghost"

                    size="icon"

                    onClick={markAllAsRead}

                    className="h-8 w-8 text-[#E9A870] hover:bg-[#E9A870]/10 rounded-full"

                    disabled={unreadCount === 0}

                    title="Marcar todas como lidas"

                  >

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

                  </Button>

                  <Button

                    variant="ghost"

                    size="icon"

                    onClick={clearAllNotifications}

                    className="h-8 w-8 text-red-400 hover:bg-red-400/10 rounded-full"

                    title="Limpar notificações"

                  >

                    <Trash2 className="h-4 w-4" />

                  </Button>

                </div>

              )}

              {notifications.length > 0 ? (

                notifications.map((notification) => (

                  <DropdownMenuItem

                    key={notification.id}

                    className={`px-4 py-3 focus:bg-gradient-to-r focus:from-[#3D3024] focus:to-[#2a2a2a] focus:text-white transition-all duration-200 hover:translate-x-1 cursor-pointer ${

                      !notification.read ? "bg-[#E9A870]/5 border-l-2 border-l-[#E9A870]" : ""

                    }`}

                    onClick={() => markAsRead(notification.id)}

                  >

                    <div className="flex items-start gap-3 w-full">

                      {getNotificationIcon(notification.type)}

                      <div className="flex-1">

                        <div className="flex items-center justify-between mb-1">

                          <span

                            className={`font-medium font-arial text-sm ${!notification.read ? "text-white" : "text-gray-300"}`}

                          >

                            {notification.title}

                          </span>

                          {!notification.read && (

                            <div className="w-2 h-2 bg-[#E9A870] rounded-full flex-shrink-0"></div>

                          )}

                        </div>

                        <div className="text-sm text-gray-400 font-arial">{notification.time}</div>

                      </div>

                    </div>

                  </DropdownMenuItem>

                ))

              ) : (

                <div className="px-4 py-8 text-center text-gray-400">

                  <Bell className="h-8 w-8 mx-auto mb-2 opacity-50" />

                  <p className="text-sm font-arial">Nenhuma notificação</p>

                </div>

              )}

              {notifications.length > 0 && (

                <div className="p-2 text-center border-t border-[#3D3024]">

                  <Button

                    variant="link"

                    className="text-[#E9A870] hover:text-[#E0CAA5] text-sm w-full transition-colors font-arial"

                  >

                    Ver todas as notificações

                  </Button>

                </div>

              )}

            </DropdownMenuContent>

          </DropdownMenu>

          <div className="text-right">

            <div className="text-lg font-bold text-[#E9A870]">

              {currentTime.toLocaleTimeString("pt-BR", {

                hour: "2-digit",

                minute: "2-digit",

              })}

            </div>

            <div className="text-sm text-gray-400">

              {currentTime.toLocaleDateString("pt-BR", {

                day: "2-digit",

                month: "short",

              })}

            </div>

          </div>

        </div>

      </header>

      <div className="p-6 space-y-6">

        {*/\* Title and Filters \*/*}

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

          <div>

            <h2 className="text-4xl font-afacad bg-gradient-to-r from-[#E9A870] via-[#E0CAA5] to-[#A8784F] bg-clip-text text-transparent font-semibold">

              DASHBOARD

            </h2>

            <p className="text-lg font-afacad text-[#E0CAA5] opacity-80">Monitoramento em tempo real</p>

          </div>

          <div className="flex items-center gap-3">

            <Select value={dateFilter} onValueChange={setDateFilter}>

              <SelectTrigger className="w-28 h-8 bg-[#3D3024] border-[#E9A870]/20 text-white text-sm">

                <Calendar className="h-3 w-3 mr-1" />

                <SelectValue />

              </SelectTrigger>

              <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white">

                <SelectItem value="hoje">Hoje</SelectItem>

                <SelectItem value="semana">Semana</SelectItem>

                <SelectItem value="mes">Mês</SelectItem>

              </SelectContent>

            </Select>

            <Select value={statusFilter} onValueChange={setStatusFilter}>

              <SelectTrigger className="w-32 h-8 bg-[#3D3024] border-[#E9A870]/20 text-white text-sm">

                <Filter className="h-3 w-3 mr-1" />

                <SelectValue />

              </SelectTrigger>

              <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white">

                <SelectItem value="todos">Todos</SelectItem>

                <SelectItem value="entrada">Entrada</SelectItem>

                <SelectItem value="pronto">Pronto</SelectItem>

                <SelectItem value="saida">Saída</SelectItem>

              </SelectContent>

            </Select>

            <Select value={tecnicoFilter} onValueChange={setTecnicoFilter}>

              <SelectTrigger className="w-32 h-8 bg-[#3D3024] border-[#E9A870]/20 text-white text-sm">

                <Filter className="h-3 w-3 mr-1" />

                <SelectValue />

              </SelectTrigger>

              <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white">

                <SelectItem value="todos">Todos</SelectItem>

                {tecnicoStats.map((tecnico) => (

                  <SelectItem key={tecnico.nome} value={tecnico.nome}>

                    {tecnico.nome}

                  </SelectItem>

                ))}

              </SelectContent>

            </Select>

          </div>

        </div>

        {*/\* Main Stats - Simplified \*/*}

        <div className="grid grid-cols-4 gap-4">

          <Card className="bg-[#1C1815] border-[#E9A870]/30 hover:border-[#E9A870]/50 transition-all duration-200">

            <CardContent className="p-4">

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

                <div>

                  <p className="text-lg text-[var(--agnus-text-muted)] uppercase tracking-wide font-bold">Total</p>

                  <p className="text-4xl font-bold text-[var(--agnus-text-primary)]">{stats.totalComputadores}</p>

                </div>

                <Monitor className="h-6 w-6 text-[#E9A870]" />

              </div>

            </CardContent>

          </Card>

          <Card className="bg-[#1C1815] border-red-500/30 hover:border-red-500/50 transition-all duration-200">

            <CardContent className="p-4">

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

                <div>

                  <p className="text-sm text-gray-400 uppercase tracking-wide">Entrada</p>

                  <p className="text-2xl font-bold text-red-400">{stats.entrada}</p>

                </div>

                <Clock className="h-6 w-6 text-red-400" />

              </div>

            </CardContent>

          </Card>

          <Card className="bg-[#1C1815] border-blue-500/30 hover:border-blue-500/50 transition-all duration-200">

            <CardContent className="p-4">

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

                <div>

                  <p className="text-sm text-gray-400 uppercase tracking-wide">Prontos</p>

                  <p className="text-2xl font-bold text-blue-400">{stats.prontos}</p>

                </div>

                <AlertTriangle className="h-6 w-6 text-blue-400" />

              </div>

            </CardContent>

          </Card>

          <Card className="bg-[#1C1815] border-green-500/30 hover:border-green-500/50 transition-all duration-200">

            <CardContent className="p-4">

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

                <div>

                  <p className="text-sm text-gray-400 uppercase tracking-wide">Entregues</p>

                  <p className="text-2xl font-bold text-green-400">{stats.saida}</p>

                </div>

                <CheckCircle className="h-6 w-6 text-green-400" />

              </div>

            </CardContent>

          </Card>

        </div>

        {*/\* Secondary Stats - Condensed \*/*}

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

          {[

            { icon: Users, label: "Técnicos", value: stats.tecnicos, color: "text-[#E9A870]" },

            { icon: Building, label: "Unidades", value: stats.unidades, color: "text-[#E9A870]" },

            { icon: Timer, label: "Tempo Médio", value: stats.tempoMedioReparo, color: "text-[#E9A870]" },

            { icon: Zap, label: "Eficiência", value: `${stats.eficiencia}%`, color: "text-[#E9A870]" },

            { icon: Award, label: "Satisfação", value: `${stats.satisfacao}%`, color: "text-[#E9A870]" },

            { icon: Target, label: "Resolução", value: `${stats.resolucaoRapida}%`, color: "text-[#E9A870]" },

          ].map((stat, index) => (

            <Card

              key={index}

              className="bg-[#1C1815] border-[#E9A870]/20 hover:bg-[#1C1815]/80 transition-all duration-200"

            >

              <CardContent className="p-3 text-center">

                <stat.icon className={`h-4 w-4 ${stat.color} mx-auto mb-1`} />

                <p className="text-sm font-bold text-white">{stat.value}</p>

                <p className="text-sm text-gray-400">{stat.label}</p>

              </CardContent>

            </Card>

          ))}

        </div>

        {*/\* Main Content Grid \*/*}

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

          {*/\* Recent Activity - Left Side \*/*}

          <div className="col-span-2">

            <Card className="bg-[#1C1815] border-[#E9A870]/30 h-[350px]">

              <CardHeader className="pb-2">

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

                  <h3 className="text-lg font-bold text-white flex items-center gap-2">

                    <Activity className="h-4 w-4 text-[#E9A870]" />

                    Atividade Recente

                  </h3>

                  <Button

                    variant="ghost"

                    size="sm"

                    className="text-[#E9A870] hover:bg-[#E9A870]/10 h-7 px-2 text-sm"

                    onClick={() => onNavigate("home")}

                  >

                    <Eye className="h-3 w-3 mr-1" />

                    Ver Todos

                  </Button>

                </div>

              </CardHeader>

              <CardContent className="pt-0 h-[280px] overflow-y-auto custom-scroll">

                <div className="space-y-1.5">

                  {recentActivity.map((activity) => (

                    <div

                      key={activity.id}

                      className="flex items-center justify-between p-2.5 bg-[#121212] rounded-lg hover:bg-[#1a1a1a] transition-colors cursor-pointer"

                      onClick={() => onNavigate("detalhes", { tombamento: activity.tombamento })}

                    >

                      <div className="flex items-center gap-2.5">

                        <div className="flex items-center gap-1.5">

                          {getStatusIcon(activity.tipo)}

                          <span className={`text-xs font-medium ${getStatusColor(activity.tipo)}`}>

                            {activity.tipo}

                          </span>

                        </div>

                        <div>

                          <p className="text-base font-medium text-[var(--agnus-text-primary)]">

                            {activity.tombamento}

                          </p>

                          <p className="text-sm text-[var(--agnus-text-muted)]">{activity.unidade}</p>

                        </div>

                      </div>

                      <div>

                        <p className="text-xs text-gray-500">{activity.ambiente}</p>

                      </div>

                      <div className="text-right">

                        <p className="text-xs text-[#E9A870]">{activity.tecnico}</p>

                        <p className="text-xs text-gray-500">{activity.tempo}</p>

                      </div>

                    </div>

                  ))}

                </div>

              </CardContent>

            </Card>

          </div>

          {*/\* PRONTOS Card - Right Side, matching height \*/*}

          <div>

            <Card className="bg-[#1C1815] border-[#E9A870]/30 h-[350px]">

              <CardHeader className="pb-2">

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

                  <h3 className="text-lg font-bold text-white flex items-center gap-2">

                    <AlertTriangle className="h-4 w-4 text-blue-400" />

                    Prontos

                  </h3>

                  <Button

                    variant="ghost"

                    size="sm"

                    className="text-[#E9A870] hover:bg-[#E9A870]/10 h-7 px-2 text-sm"

                    onClick={() => onNavigate("home")}

                  >

                    <Eye className="h-3 w-3" />

                  </Button>

                </div>

              </CardHeader>

              <CardContent className="pt-0 h-[280px] overflow-y-auto custom-scroll">

                <div className="space-y-2">

                  {prontosItems.length > 0 ? (

                    prontosItems.map((item) => (

                      <div

                        key={item.id}

                        className="p-2.5 bg-[#121212] rounded-lg hover:bg-[#1a1a1a] transition-colors cursor-pointer border-l-2 border-l-blue-400"

                        onClick={() => onNavigate("detalhes", { tombamento: item.tombamento })}

                      >

                        <div className="flex items-center justify-between mb-1.5">

                          <p className="text-base font-bold text-white">{item.tombamento}</p>

                          <span className="text-xs text-blue-400 font-medium">{item.ambiente}</span>

                        </div>

                        <p className="text-xs text-gray-400 mb-1">{item.unidade}</p>

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

                          <p className="text-xs text-[#E9A870]">{item.tecnico}</p>

                          <p className="text-xs text-gray-500">{item.tempo}</p>

                        </div>

                      </div>

                    ))

                  ) : (

                    <div className="text-center py-6 text-gray-400">

                      <AlertTriangle className="h-10 w-10 mx-auto mb-2 opacity-50" />

                      <p className="text-base">Nenhum item pronto</p>

                      <p className="text-xs">Aguardando conclusão de reparos</p>

                    </div>

                  )}

                </div>

              </CardContent>

            </Card>

          </div>

        </div>

        {*/\* Técnicos Carousel \*/*}

        <Card className="bg-[#1C1815] border-[#E9A870]/30">

          <CardHeader className="pb-2">

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

              <h3 className="text-lg font-bold text-white flex items-center gap-2">

                <User className="h-4 w-4 text-[#E9A870]" />

                Técnicos

              </h3>

              <div className="flex items-center gap-2">

                <Button

                  variant="ghost"

                  size="icon"

                  onClick={prevTecnico}

                  className="h-8 w-8 text-[#E9A870] hover:bg-[#E9A870]/10 rounded-full transition-all duration-200"

                  disabled={isTransitioning || currentTecnicoIndex === 0}

                >

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

                </Button>

                <Button

                  variant="ghost"

                  size="icon"

                  onClick={nextTecnico}

                  className="h-8 w-8 text-[#E9A870] hover:bg-[#E9A870]/10 rounded-full transition-all duration-200"

                  disabled={isTransitioning || currentTecnicoIndex >= Math.max(0, tecnicoStats.length - 4)}

                >

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

                </Button>

                <Button

                  variant="ghost"

                  size="sm"

                  className="text-[#E9A870] hover:bg-[#E9A870]/10 h-7 px-2 text-sm ml-2"

                  onClick={() => onNavigate("tecnicos")}

                >

                  <Eye className="h-3 w-3" />

                </Button>

              </div>

            </div>

          </CardHeader>

          <CardContent className="pt-0">

            <div className="relative overflow-hidden">

              <div

                className="flex transition-transform duration-300 ease-in-out gap-2"

                style={{

                  transform: `translateX(-${currentTecnicoIndex \* (100 / 4)}%)`,

                  width: `${(tecnicoStats.length / 4) \* 100}%`,

                }}

              >

                {tecnicoStats.map((tecnico, index) => (

                  <div

                    key={tecnico.nome}

                    className="flex-shrink-0 w-1/4 p-2.5 bg-[#121212] rounded-lg hover:bg-[#1a1a1a] transition-colors cursor-pointer border border-[#E9A870]/20 hover:border-[#E9A870]/40"

                    onClick={() => onNavigate("tecnicos", { selectedTecnico: tecnico.nome })}

                  >

                    <div className="flex items-center gap-2 mb-2">

                      <div className="w-7 h-7 bg-gradient-to-r from-[#E9A870] to-[#A8784F] rounded-full flex items-center justify-center">

                        <User className="h-3.5 w-3.5 text-black" />

                      </div>

                      <div>

                        <p className="text-xs font-bold text-white">{tecnico.nome}</p>

                        <p className="text-xs text-gray-400">{tecnico.tickets} tickets</p>

                      </div>

                    </div>

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

                      <span className="text-base font-bold text-[#E9A870]">{tecnico.eficiencia}%</span>

                      <div className="w-12 bg-gray-700 rounded-full h-1.5">

                        <div

                          className="bg-[#E9A870] h-1.5 rounded-full transition-all duration-500"

                          style={{ width: `${tecnico.eficiencia}%` }}

                        ></div>

                      </div>

                    </div>

                  </div>

                ))}

              </div>

              {*/\* Scroll Indicators \*/*}

              <div className="flex justify-center mt-3 gap-1">

                {Array.from({ length: Math.ceil(tecnicoStats.length / 4) }).map((\_, index) => (

                  <button

                    key={index}

                    onClick={() => {

                      if (!isTransitioning) {

                        setIsTransitioning(true)

                        setCurrentTecnicoIndex(index)

                        setTimeout(() => setIsTransitioning(false), 300)

                      }

                    }}

                    className={`w-2 h-2 rounded-full transition-all duration-200 ${

                      Math.floor(currentTecnicoIndex / 1) === index ? "bg-[#E9A870]" : "bg-gray-600 hover:bg-gray-500"

                    }`}

                  />

                ))}

              </div>

            </div>

          </CardContent>

        </Card>

        {*/\* Units Overview - Simplified \*/*}

        <Card className="bg-[#1C1815] border-[#E9A870]/30">

          <CardHeader className="pb-3">

            <h3 className="text-lg font-bold text-white flex items-center gap-2">

              <Building className="h-4 w-4 text-[#E9A870]" />

              Unidades de Saúde

            </h3>

          </CardHeader>

          <CardContent className="pt-0">

            <div className="grid grid-cols-4 gap-4">

              {unidadeStats.map((unidade) => (

                <div key={unidade.nome} className="p-3 bg-[#121212] rounded-lg hover:bg-[#1a1a1a] transition-colors">

                  <h4 className="text-sm font-medium text-white mb-2">{unidade.nome}</h4>

                  <div className="flex justify-between text-sm">

                    <div className="text-center">

                      <div className="flex items-center gap-1 mb-1">

                        <div className="w-2 h-2 bg-red-500 rounded-full"></div>

                        <span className="text-red-400">{unidade.entrada}</span>

                      </div>

                      <p className="text-gray-500">Entrada</p>

                    </div>

                    <div className="text-center">

                      <div className="flex items-center gap-1 mb-1">

                        <div className="w-2 h-2 bg-blue-500 rounded-full"></div>

                        <span className="text-blue-400">{unidade.prontos}</span>

                      </div>

                      <p className="text-gray-500">Prontos</p>

                    </div>

                    <div className="text-center">

                      <div className="flex items-center gap-1 mb-1">

                        <div className="w-2 h-2 bg-green-500 rounded-full"></div>

                        <span className="text-green-400">{unidade.saida}</span>

                      </div>

                      <p className="text-gray-500">Saída</p>

                    </div>

                  </div>

                </div>

              ))}

            </div>

          </CardContent>

        </Card>

        {*/\* Performance Indicators - Simplified \*/*}

        <Card className="bg-[#1C1815] border-[#E9A870]/30">

          <CardHeader className="pb-3">

            <h3 className="text-lg font-bold text-white flex items-center gap-2">

              <TrendingUp className="h-4 w-4 text-[#E9A870]" />

              Indicadores de Performance

            </h3>

          </CardHeader>

          <CardContent className="pt-0">

            <div className="grid grid-cols-4 gap-6">

              {[

                { label: "Taxa de Resolução", value: "87%", target: "85%", progress: 87, color: "green" },

                { label: "Tempo Médio", value: "2.5h", target: "3h", progress: 75, color: "blue" },

                { label: "Satisfação", value: "94%", target: "90%", progress: 94, color: "orange" },

                { label: "Tempo de Resposta", value: "1.2h", target: "1.5h", progress: 80, color: "purple" },

              ].map((metric, index) => (

                <div key={index} className="text-center">

                  <div className={`text-2xl font-bold text-${metric.color}-400 mb-1`}>{metric.value}</div>

                  <p className="text-sm text-gray-400 mb-2">{metric.label}</p>

                  <div className="w-full bg-gray-700 rounded-full h-2">

                    <div

                      className={`bg-${metric.color}-400 h-2 rounded-full transition-all duration-500`}

                      style={{ width: `${metric.progress}%` }}

                    ></div>

                  </div>

                  <p className={`text-sm text-${metric.color}-400 mt-1`}>Meta: {metric.target}</p>

                </div>

              ))}

            </div>

          </CardContent>

        </Card>

      </div>

    </div>

  )

}

frontend/app/detalhes/page.tsx:

"use client"

import type React from "react"

import { useState } from "react"

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

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

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

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

import { Textarea } from "@/components/ui/textarea"

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

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

import {

  AlertDialog,

  AlertDialogAction,

  AlertDialogCancel,

  AlertDialogContent,

  AlertDialogDescription,

  AlertDialogFooter,

  AlertDialogHeader,

  AlertDialogTitle,

} from "@/components/ui/alert-dialog"

import {

  Menu,

  ChevronLeft,

  Clock,

  CheckCircle,

  AlertTriangle,

  User,

  MapPin,

  Monitor,

  FileText,

  Calendar,

  History,

  Home,

  ChevronRight,

  Printer,

  Trash2,

  Edit,

  Check,

} from "lucide-react"

import { SidebarMenu } from "@/components/sidebar-menu"

import { toast, Toaster } from "sonner"

interface DetalhesRegistroProps {

  onNavigate: (page: string, data?: any) => void

  data?: any

}

export default function DetalhesRegistro({ onNavigate, data }: DetalhesRegistroProps) {

*const* [sidebarOpen, setSidebarOpen] = useState(false)

*const* [printDialogOpen, setPrintDialogOpen] = useState(false)

*const* [deleteDialogOpen, setDeleteDialogOpen] = useState(false)

*const* [editDialogOpen, setEditDialogOpen] = useState(false)

*const* [markReadyDialogOpen, setMarkReadyDialogOpen] = useState(false)

*const* [markDeliveredDialogOpen, setMarkDeliveredDialogOpen] = useState(false)

*// Mock detailed data - in real app this would come from API*

*const* ticketDetails = data || {

    id: "O1",

    tombamento: "202259123",

    status: "ENTRADA",

    unidade: "USF RIO DAS VELHAS II",

    tecnico: "João Silva",

    ambiente: "MÉDICO",

    tipo: "DESKTOP",

    descricao: "Computador não liga após queda de energia. Verificar fonte e componentes internos.",

    dataAbertura: "2025-01-15 14:30",

    dataUltimaAtualizacao: "2025-01-15 16:45",

    dataSaida: "2025-01-18 10:15",

    observacoes: "Cliente relatou que o problema começou após tempestade na região.",

  }

*// Edit form state*

*const* [editFormData, setEditFormData] = useState({

    tombamento: ticketDetails.tombamento,

    ambiente: ticketDetails.ambiente,

    tipo: ticketDetails.tipo,

    unidade: ticketDetails.unidade,

    descricao: ticketDetails.descricao,

    tecnico: ticketDetails.tecnico,

  })

*const* unidades = ["USF RIO DAS VELHAS II", "USF QUADROS III", "USF VIETNÃ", "USF CENTRO", "USF NORTE"]

*const* tecnicos = ["João Silva", "Maria Santos", "Pedro Costa", "Ana Lima", "Carlos Oliveira"]

*// Mock related tickets*

*const* relatedTickets = [

    {

      id: "O5",

      tombamento: "202259127",

      status: "PRONTO",

      unidade: "USF RIO DAS VELHAS II",

      descricao: "Problema similar resolvido",

      dataAbertura: "2025-01-10",

    },

    {

      id: "O8",

      tombamento: "202259130",

      status: "ENTRADA",

      unidade: "USF RIO DAS VELHAS II",

      descricao: "Mesmo setor - verificar rede elétrica",

      dataAbertura: "2025-01-14",

    },

  ]

*// Mock history*

*const* ticketHistory = [

    {

      id: 1,

      acao: "Ticket criado",

      usuario: "João Silva",

      data: "2025-01-15 14:30",

      detalhes: "Abertura do chamado",

    },

    {

      id: 2,

      acao: "Status atualizado",

      usuario: "Sistema",

      data: "2025-01-15 14:31",

      detalhes: "Status alterado para ENTRADA",

    },

    {

      id: 3,

      acao: "Observação adicionada",

      usuario: "João Silva",

      data: "2025-01-15 16:45",

      detalhes: "Adicionada observação sobre tempestade",

    },

  ]

*const* getStatusIcon = (status: string) => {

    switch (status) {

      case "ENTRADA":

*return* <Clock className="h-4 w-4 text-red-400" />

      case "SAÍDA":

*return* <CheckCircle className="h-4 w-4 text-green-400" />

      case "PRONTO":

*return* <AlertTriangle className="h-4 w-4 text-blue-400" />

      default:

*return* <Clock className="h-4 w-4 text-gray-400" />

    }

  }

*const* getStatusColor = (status: string) => {

    switch (status) {

      case "ENTRADA":

*return* "bg-red-500/20 text-red-300 border-red-500/30"

      case "SAÍDA":

*return* "bg-green-500/20 text-green-300 border-green-500/30"

      case "PRONTO":

*return* "bg-blue-500/20 text-blue-300 border-blue-500/30"

      default:

*return* "bg-gray-500/20 text-gray-300 border-gray-500/30"

    }

  }

*const* handlePrintLabel = () => {

    toast.success("Etiqueta enviada para impressão", {

      duration: 3000,

      position: "top-right",

    })

    setPrintDialogOpen(false)

  }

*const* handleDeleteRecord = () => {

    toast.success(`Registro ${ticketDetails.tombamento} foi excluído com sucesso`, {

      duration: 3000,

      position: "top-right",

    })

    setDeleteDialogOpen(false)

    onNavigate("home")

  }

*const* handleMarkAsReady = () => {

    toast.success(`Registro ${ticketDetails.tombamento} marcado como PRONTO`, {

      duration: 3000,

      position: "top-right",

    })

    setMarkReadyDialogOpen(false)

  }

*const* handleMarkAsDelivered = () => {

    toast.success(`Registro ${ticketDetails.tombamento} marcado como SAÍDA`, {

      duration: 3000,

      position: "top-right",

    })

    setMarkDeliveredDialogOpen(false)

  }

*const* handleEditSubmit = (e: React.FormEvent) => {

    e.preventDefault()

    toast.success(`Registro ${ticketDetails.tombamento} foi atualizado com sucesso`, {

      duration: 3000,

      position: "top-right",

    })

    setEditDialogOpen(false)

  }

*const* isEditFormValid =

    editFormData.tombamento &&

    editFormData.ambiente &&

    editFormData.unidade &&

    editFormData.descricao &&

    editFormData.tecnico

*return* (

    <div className="min-h-screen bg-[#191818] text-white">

      {*/\* Toaster for notifications \*/*}

      <Toaster

        theme="dark"

        position="top-right"

        toastOptions={{

          style: {

            background: "#1C1815",

            border: "1px solid #E9A870",

            color: "#E0CAA5",

          },

        }}

      />

      {*/\* Sidebar Menu \*/*}

      <SidebarMenu

        isOpen={sidebarOpen}

        onClose={() => setSidebarOpen(false)}

        onNavigate={onNavigate}

        currentPage="detalhes"

      />

      {*/\* Header - Fixed/Sticky \*/*}

      <header className="sticky top-0 z-30 flex items-center justify-between px-6 py-4 bg-[#0C0C0C] border-b border-[#E9A870] shadow-lg backdrop-blur-sm">

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

          <Button

            variant="ghost"

            size="icon"

            className="text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

            onClick={() => setSidebarOpen(true)}

          >

            <Menu className="h-6 w-6" />

          </Button>

          <h1 className="text-[42px] font-abel text-[#E0CAA5] tracking-wider drop-shadow-sm">AGNUS</h1>

          {*/\* Breadcrumbs \*/*}

          <div className="flex items-center text-sm text-gray-400 ml-6">

            <Button

              variant="link"

              className="p-0 h-auto text-[#E0CAA5] hover:text-[#E9A870] transition-colors font-arial"

              onClick={() => onNavigate("home")}

            >

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

              <span>Início</span>

            </Button>

            <ChevronRight className="h-3 w-3 mx-2 text-gray-500" />

            <Button

              variant="link"

              className="p-0 h-auto text-[#E0CAA5] hover:text-[#E9A870] transition-colors font-arial"

              onClick={() => onNavigate("home")}

            >

              <span>Computadores</span>

            </Button>

            <ChevronRight className="h-3 w-3 mx-2 text-gray-500" />

            <span className="text-gray-400 font-arial">Detalhes</span>

          </div>

          <Button

            variant="ghost"

            size="icon"

            onClick={() => onNavigate("home")}

            className="text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

          >

            <ChevronLeft className="h-6 w-6" />

          </Button>

        </div>

      </header>

      <div className="p-6 custom-scroll">

        <div className="mb-6">

          <h2 className="text-[58px] font-afacad bg-gradient-to-r from-[#E9A870] via-[#E0CAA5] to-[#A8784F] bg-clip-text text-transparent font-semibold leading-tight drop-shadow-sm">

            DETALHES DO REGISTRO

          </h2>

          <p className="text-[32px] font-afacad text-[#E0CAA5] leading-tight opacity-90">

            TOMBAMENTO: {ticketDetails.tombamento}

          </p>

        </div>

        <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">

          {*/\* Main Details \*/*}

          <div className="lg:col-span-2 space-y-6">

            {*/\* Basic Information \*/*}

            <Card className="bg-[#1C1815] border-[#E9A870] border-2 rounded-xl overflow-hidden shadow-2xl backdrop-blur-sm">

              <CardHeader className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] p-4">

                <h3 className="text-xl font-bold text-black flex items-center gap-2 font-arial">

                  <FileText className="h-5 w-5" />

                  INFORMAÇÕES BÁSICAS

                </h3>

              </CardHeader>

              <CardContent className="p-6 bg-[#1C1815] space-y-4 custom-scroll">

                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">ID</label>

                    <p className="text-white font-bold text-lg font-abel">{ticketDetails.id}</p>

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">TOMBAMENTO</label>

                    <p className="text-white font-bold text-lg font-abel">{ticketDetails.tombamento}</p>

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">STATUS</label>

                    <div className="flex items-center gap-2">

                      {getStatusIcon(ticketDetails.status)}

                      <Badge className={`${getStatusColor(ticketDetails.status)} font-bold font-arial`}>

                        {ticketDetails.status}

                      </Badge>

                    </div>

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] flex items-center gap-1 font-arial">

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

                      DATA SAÍDA

                    </label>

                    <p className="text-white text-lg font-abel">{ticketDetails.dataSaida || "Não definida"}</p>

                  </div>

                </div>

                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] flex items-center gap-1 font-arial">

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

                      TÉCNICO

                    </label>

                    <button

                      onClick={() => onNavigate("tecnicos", { selectedTecnico: ticketDetails.tecnico })}

                      className="text-white font-bold hover:text-[#E9A870] transition-colors cursor-pointer text-lg font-abel"

                    >

                      {ticketDetails.tecnico}

                    </button>

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] flex items-center gap-1 font-arial">

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

                      UNIDADE

                    </label>

                    <p className="text-white font-bold text-lg font-abel">{ticketDetails.unidade}</p>

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">AMBIENTE</label>

                    <p className="text-white font-bold text-lg font-abel">{ticketDetails.ambiente}</p>

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] flex items-center gap-1 font-arial">

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

                      TIPO

                    </label>

                    <p className="text-white font-bold text-lg font-abel">{ticketDetails.tipo}</p>

                  </div>

                </div>

                <div className="space-y-2">

                  <label className="text-sm font-medium text-[#E9A870] font-arial">DESCRIÇÃO DO PROBLEMA</label>

                  <p className="text-white bg-[#121212] p-4 rounded-lg border border-[#E9A870]/20 text-base font-arial leading-relaxed">

                    {ticketDetails.descricao}

                  </p>

                </div>

                {ticketDetails.observacoes && (

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">OBSERVAÇÕES</label>

                    <p className="text-white bg-[#121212] p-4 rounded-lg border border-[#E9A870]/20 text-base font-arial leading-relaxed">

                      {ticketDetails.observacoes}

                    </p>

                  </div>

                )}

                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] flex items-center gap-1 font-arial">

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

                      DATA DE ABERTURA

                    </label>

                    <p className="text-white text-base font-arial">{ticketDetails.dataAbertura}</p>

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] flex items-center gap-1 font-arial">

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

                      ÚLTIMA ATUALIZAÇÃO

                    </label>

                    <p className="text-white text-base font-arial">{ticketDetails.dataUltimaAtualizacao}</p>

                  </div>

                </div>

                {*/\* Action Buttons \*/*}

                <div className="flex justify-center gap-4 pt-4 border-t border-[#E9A870]/20">

                  {*/\* MARCAR COMO PRONTO \*/*}

                  <div className="relative">

                    <button

                      onClick={() => setMarkReadyDialogOpen(true)}

                      className="w-16 h-16 flex items-center justify-center rounded-xl bg-[#1C1815] hover:bg-[#2A231E] transition-all duration-200 hover:scale-105 cursor-pointer"

                      title="MARCAR COMO PRONTO"

                    >

                      <div className="w-14 h-14 rounded-xl flex items-center justify-center border-2 border-[#E9A870]">

                        <svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">

                          <path

                            d="M12 22C17.5228 22 22 17.5228 22 12C22 6.47715 17.5228 2 12 2C6.47715 2 2 6.47715 2 12C2 17.5228 6.47715 22 12 22Z"

                            stroke="#E9A870"

                            strokeWidth="2"

                            strokeLinecap="round"

                            strokeLinejoin="round"

                          />

                          <path

                            d="M9 12L11 14L15 10"

                            stroke="#E9A870"

                            strokeWidth="2"

                            strokeLinecap="round"

                            strokeLinejoin="round"

                          />

                        </svg>

                      </div>

                    </button>

                    <AlertDialog open={markReadyDialogOpen} onOpenChange={setMarkReadyDialogOpen}>

                      <AlertDialogContent className="bg-[#1C1815] border-[#E9A870] text-white">

                        <AlertDialogHeader>

                          <AlertDialogTitle className="text-[#E9A870] text-xl font-abel">

                            Marcar como Pronto

                          </AlertDialogTitle>

                          <AlertDialogDescription className="text-gray-300 font-arial">

                            Tem certeza que deseja marcar o registro{" "}

                            <span className="font-bold text-white">{ticketDetails.tombamento}</span> como PRONTO?

                            <br />

                            <span className="text-blue-400 text-sm">

                              O status será alterado para "PRONTO" e ficará disponível para entrega.

                            </span>

                          </AlertDialogDescription>

                        </AlertDialogHeader>

                        <AlertDialogFooter>

                          <AlertDialogCancel className="bg-gray-600 hover:bg-gray-700 text-white border-gray-500">

                            Cancelar

                          </AlertDialogCancel>

                          <AlertDialogAction

                            onClick={handleMarkAsReady}

                            className="bg-gradient-to-r from-blue-600 to-blue-700 hover:from-blue-700 hover:to-blue-800 text-white"

                          >

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

                            Marcar como Pronto

                          </AlertDialogAction>

                        </AlertDialogFooter>

                      </AlertDialogContent>

                    </AlertDialog>

                  </div>

                  {*/\* MARCAR COMO SAÍDA \*/*}

                  <div className="relative">

                    <button

                      onClick={() => setMarkDeliveredDialogOpen(true)}

                      className="w-16 h-16 flex items-center justify-center rounded-xl bg-[#1C1815] hover:bg-[#2A231E] transition-all duration-200 hover:scale-105 cursor-pointer"

                      title="MARCAR COMO SAÍDA"

                    >

                      <div className="w-14 h-14 rounded-xl flex items-center justify-center border-2 border-[#E9A870]">

                        <svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">

                          <path

                            d="M9 21H5C4.46957 21 3.96086 20.7893 3.58579 20.4142C3.21071 20.0391 3 19.5304 3 19V5C3 4.46957 3.21071 3.96086 3.58579 3.58579C3.96086 3.21071 4.46957 3 5 3H9"

                            stroke="#E9A870"

                            strokeWidth="2"

                            strokeLinecap="round"

                            strokeLinejoin="round"

                          />

                          <path

                            d="M16 17L21 12L16 7"

                            stroke="#E9A870"

                            strokeWidth="2"

                            strokeLinecap="round"

                            strokeLinejoin="round"

                          />

                          <path

                            d="M21 12H9"

                            stroke="#E9A870"

                            strokeWidth="2"

                            strokeLinecap="round"

                            strokeLinejoin="round"

                          />

                        </svg>

                      </div>

                    </button>

                    <AlertDialog open={markDeliveredDialogOpen} onOpenChange={setMarkDeliveredDialogOpen}>

                      <AlertDialogContent className="bg-[#1C1815] border-[#E9A870] text-white">

                        <AlertDialogHeader>

                          <AlertDialogTitle className="text-[#E9A870] text-xl font-abel">

                            Marcar como Saída

                          </AlertDialogTitle>

                          <AlertDialogDescription className="text-gray-300 font-arial">

                            Tem certeza que deseja marcar o registro{" "}

                            <span className="font-bold text-white">{ticketDetails.tombamento}</span> como SAÍDA?

                            <br />

                            <span className="text-green-400 text-sm">

                              O equipamento será marcado como entregue e o chamado será finalizado.

                            </span>

                          </AlertDialogDescription>

                        </AlertDialogHeader>

                        <AlertDialogFooter>

                          <AlertDialogCancel className="bg-gray-600 hover:bg-gray-700 text-white border-gray-500">

                            Cancelar

                          </AlertDialogCancel>

                          <AlertDialogAction

                            onClick={handleMarkAsDelivered}

                            className="bg-gradient-to-r from-green-600 to-green-700 hover:from-green-700 hover:to-green-800 text-white"

                          >

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

                            Marcar como Saída

                          </AlertDialogAction>

                        </AlertDialogFooter>

                      </AlertDialogContent>

                    </AlertDialog>

                  </div>

                  {*/\* IMPRIMIR ETIQUETA \*/*}

                  <div className="relative">

                    <button

                      onClick={() => setPrintDialogOpen(true)}

                      className="w-16 h-16 flex items-center justify-center rounded-xl bg-[#1C1815] hover:bg-[#2A231E] transition-all duration-200 hover:scale-105 cursor-pointer"

                      title="IMPRIMIR ETIQUETA"

                    >

                      <div className="w-14 h-14 rounded-xl flex items-center justify-center border-2 border-[#E9A870]">

                        <Printer className="h-6 w-6 text-[#E9A870]" />

                      </div>

                    </button>

                    <Dialog open={printDialogOpen} onOpenChange={setPrintDialogOpen}>

                      <DialogContent className="bg-gradient-to-b from-[#1C1815] to-[#0C0C0C] border-2 border-[#E9A870] text-white max-w-md shadow-2xl backdrop-blur-sm">

                        <DialogHeader>

                          <DialogTitle className="text-[#E9A870] text-xl font-abel bg-gradient-to-r from-[#E9A870] to-[#A8784F] bg-clip-text text-transparent">

                            Imprimir Etiqueta

                          </DialogTitle>

                          <div className="w-full h-px bg-gradient-to-r from-transparent via-[#E9A870] to-transparent mt-4"></div>

                        </DialogHeader>

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

                          <div className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] p-4 rounded-lg border border-[#E9A870]/20">

                            <h4 className="font-bold text-white mb-2 font-arial">Prévia da Etiqueta:</h4>

                            <div className="bg-white text-black p-3 rounded text-sm font-mono">

                              <div className="text-center border-b border-gray-300 pb-2 mb-2">

                                <strong>AGNUS - GETI</strong>

                              </div>

                              <div>ID: {ticketDetails.id}</div>

                              <div>TOMB: {ticketDetails.tombamento}</div>

                              <div>UNIDADE: {ticketDetails.unidade}</div>

                              <div>TÉCNICO: {ticketDetails.tecnico}</div>

                              <div>DATA: {new Date().toLocaleDateString("pt-BR")}</div>

                            </div>

                          </div>

                          <div className="flex justify-end gap-3">

                            <Button

                              variant="outline"

                              onClick={() => setPrintDialogOpen(false)}

                              className="bg-gradient-to-r from-[#2A231E] to-[#1C1815] border-[#E9A870]/40 text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#3D3024] hover:to-[#2A231E] hover:border-[#E9A870] transition-all duration-200"

                            >

                              Cancelar

                            </Button>

                            <Button

                              onClick={handlePrintLabel}

                              className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] hover:from-[#E0CAA5] hover:to-[#E9A870] text-black font-bold shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105"

                            >

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

                              Imprimir

                            </Button>

                          </div>

                        </div>

                      </DialogContent>

                    </Dialog>

                  </div>

                  {*/\* EXCLUIR REGISTRO \*/*}

                  <div className="relative">

                    <button

                      onClick={() => setDeleteDialogOpen(true)}

                      className="w-16 h-16 flex items-center justify-center rounded-xl bg-[#1C1815] hover:bg-[#2A231E] transition-all duration-200 hover:scale-105 cursor-pointer"

                      title="EXCLUIR REGISTRO"

                    >

                      <div className="w-14 h-14 rounded-xl flex items-center justify-center border-2 border-[#E9A870]">

                        <Trash2 className="h-6 w-6 text-[#E9A870]" />

                      </div>

                    </button>

                    <AlertDialog open={deleteDialogOpen} onOpenChange={setDeleteDialogOpen}>

                      <AlertDialogContent className="bg-[#1C1815] border-[#E9A870] text-white">

                        <AlertDialogHeader>

                          <AlertDialogTitle className="text-[#E9A870] text-xl font-abel">

                            Confirmar Exclusão

                          </AlertDialogTitle>

                          <AlertDialogDescription className="text-gray-300 font-arial">

                            Tem certeza que deseja excluir o registro do computador{" "}

                            <span className="font-bold text-white">{ticketDetails.tombamento}</span>?

                            <br />

                            <span className="text-red-400 text-sm">Esta ação não pode ser desfeita.</span>

                          </AlertDialogDescription>

                        </AlertDialogHeader>

                        <AlertDialogFooter>

                          <AlertDialogCancel className="bg-gray-600 hover:bg-gray-700 text-white border-gray-500">

                            Cancelar

                          </AlertDialogCancel>

                          <AlertDialogAction

                            onClick={handleDeleteRecord}

                            className="bg-gradient-to-r from-red-600 to-red-700 hover:from-red-700 hover:to-red-800 text-white"

                          >

                            <Trash2 className="h-4 w-4 mr-2" />

                            Excluir

                          </AlertDialogAction>

                        </AlertDialogFooter>

                      </AlertDialogContent>

                    </AlertDialog>

                  </div>

                  {*/\* EDITAR REGISTRO \*/*}

                  <div className="relative">

                    <button

                      onClick={() => setEditDialogOpen(true)}

                      className="w-16 h-16 flex items-center justify-center rounded-xl bg-[#1C1815] hover:bg-[#2A231E] transition-all duration-200 hover:scale-105 cursor-pointer"

                      title="EDITAR REGISTRO"

                    >

                      <div className="w-14 h-14 rounded-xl flex items-center justify-center border-2 border-[#E9A870]">

                        <Edit className="h-6 w-6 text-[#E9A870]" />

                      </div>

                    </button>

                    <Dialog open={editDialogOpen} onOpenChange={setEditDialogOpen}>

                      <DialogContent className="bg-gradient-to-b from-[#1C1815] to-[#0C0C0C] border-2 border-[#E9A870] text-white max-w-4xl font-abel shadow-2xl backdrop-blur-sm">

                        <DialogHeader>

                          <DialogTitle className="text-[#E9A870] text-2xl font-bold bg-gradient-to-r from-[#E9A870] to-[#A8784F] bg-clip-text text-transparent">

                            EDITAR REGISTRO

                          </DialogTitle>

                          <div className="w-full h-px bg-gradient-to-r from-transparent via-[#E9A870] to-transparent mt-4"></div>

                        </DialogHeader>

                        <form

                          onSubmit={handleEditSubmit}

                          className="space-y-6 custom-scroll max-h-[70vh] overflow-y-auto p-1"

                        >

                          {*/\* First Row \*/*}

                          <div className="grid grid-cols-1 md:grid-cols-3 gap-4">

                            <div className="space-y-2">

                              <label className="text-sm font-medium text-[#E9A870]">

                                TOMBAMENTO <span className="text-red-400">\*</span>

                              </label>

                              <Input

                                placeholder="Ex: 202212345"

                                value={editFormData.tombamento}

                                onChange={(e) => setEditFormData({ **...**editFormData, tombamento: e.target.value })}

                                className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                              />

                            </div>

                            <div className="space-y-2">

                              <label className="text-sm font-medium text-[#E9A870]">

                                AMBIENTE <span className="text-red-400">\*</span>

                              </label>

                              <Input

                                placeholder="Ex: Médico"

                                value={editFormData.ambiente}

                                onChange={(e) => setEditFormData({ **...**editFormData, ambiente: e.target.value })}

                                className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                              />

                            </div>

                            <div className="space-y-2">

                              <label className="text-sm font-medium text-[#E9A870]">TIPO</label>

                              <Input

                                placeholder="Ex: TCORP"

                                value={editFormData.tipo}

                                onChange={(e) => setEditFormData({ **...**editFormData, tipo: e.target.value })}

                                className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                              />

                            </div>

                          </div>

                          {*/\* Second Row \*/*}

                          <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                            <div className="space-y-2">

                              <label className="text-sm font-medium text-[#E9A870]">

                                UNIDADE <span className="text-red-400">\*</span>

                              </label>

                              <Select

                                value={editFormData.unidade}

                                onValueChange={(value) => setEditFormData({ **...**editFormData, unidade: value })}

                              >

                                <SelectTrigger className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200">

                                  <SelectValue placeholder="SELECIONE A UNIDADE" />

                                </SelectTrigger>

                                <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white custom-scroll">

                                  {unidades.map((unidade) => (

                                    <SelectItem

                                      key={unidade}

                                      value={unidade}

                                      className="focus:bg-[#E9A870]/20 font-arial"

                                    >

                                      {unidade}

                                    </SelectItem>

                                  ))}

                                </SelectContent>

                              </Select>

                            </div>

                            <div className="space-y-2">

                              <label className="text-sm font-medium text-[#E9A870]">

                                TÉCNICO <span className="text-red-400">\*</span>

                              </label>

                              <Select

                                value={editFormData.tecnico}

                                onValueChange={(value) => setEditFormData({ **...**editFormData, tecnico: value })}

                              >

                                <SelectTrigger className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200">

                                  <SelectValue placeholder="TÉCNICO" />

                                </SelectTrigger>

                                <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white custom-scroll">

                                  {tecnicos.map((tecnico) => (

                                    <SelectItem

                                      key={tecnico}

                                      value={tecnico}

                                      className="focus:bg-[#E9A870]/20 font-arial"

                                    >

                                      {tecnico}

                                    </SelectItem>

                                  ))}

                                </SelectContent>

                              </Select>

                            </div>

                          </div>

                          {*/\* Third Row \*/*}

                          <div className="space-y-2">

                            <label className="text-sm font-medium text-[#E9A870]">

                              DESCRIÇÃO DO DEFEITO <span className="text-red-400">\*</span>

                            </label>

                            <Textarea

                              placeholder="Ex: SEM APRESENTAR IMAGEM, PROVÁVEL DF NA MEMÓRIA RAM"

                              value={editFormData.descricao}

                              onChange={(e) => setEditFormData({ **...**editFormData, descricao: e.target.value })}

                              className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 min-h-[100px] font-arial resize-none custom-scroll transition-all duration-200"

                            />

                          </div>

                          <div className="flex justify-end gap-4 pt-6 border-t border-[#E9A870]/20">

                            <Button

                              type="button"

                              variant="outline"

                              onClick={() => setEditDialogOpen(false)}

                              className="bg-gradient-to-r from-[#2A231E] to-[#1C1815] border-[#E9A870]/40 text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#3D3024] hover:to-[#2A231E] hover:border-[#E9A870] transition-all duration-200 font-abel"

                            >

                              CANCELAR

                            </Button>

                            <Button

                              type="submit"

                              disabled={!isEditFormValid}

                              className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] hover:from-[#E0CAA5] hover:to-[#E9A870] text-[#0C0C0C] font-bold disabled:opacity-50 shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105"

                            >

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

                              ATUALIZAR REGISTRO

                            </Button>

                          </div>

                        </form>

                      </DialogContent>

                    </Dialog>

                  </div>

                </div>

              </CardContent>

            </Card>

          </div>

          {*/\* Sidebar \*/*}

          <div className="space-y-6">

            {*/\* Related Tickets \*/*}

            <Card className="bg-[#1C1815] border-[#E9A870] border-2 rounded-xl overflow-hidden shadow-2xl backdrop-blur-sm">

              <CardHeader className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] p-4">

                <h3 className="text-lg font-bold text-black font-arial">CHAMADOS RELACIONADOS</h3>

              </CardHeader>

              <CardContent className="p-4 bg-[#1C1815] custom-scroll">

                <div className="space-y-3">

                  {relatedTickets.map((ticket) => (

                    <div

                      key={ticket.id}

                      className="p-3 bg-[#121212] rounded-lg border border-[#E9A870]/20 hover:bg-[#262626] transition-colors cursor-pointer"

                      onClick={() => onNavigate("detalhes", ticket)}

                    >

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

                        <span className="font-bold text-white text-base font-abel">{ticket.id}</span>

                        <div className="flex items-center gap-1">

                          {getStatusIcon(ticket.status)}

                          <Badge className={`${getStatusColor(ticket.status)} text-sm font-arial`}>

                            {ticket.status}

                          </Badge>

                        </div>

                      </div>

                      <p className="text-sm text-gray-300 mb-1 font-arial">{ticket.tombamento}</p>

                      <p className="text-sm text-gray-400 mb-2 font-arial">{ticket.unidade}</p>

                      <p className="text-sm text-white font-arial">{ticket.descricao}</p>

                      <p className="text-sm text-[#E9A870] mt-1 font-arial">{ticket.dataAbertura}</p>

                    </div>

                  ))}

                </div>

              </CardContent>

            </Card>

            {*/\* History \*/*}

            <Card className="bg-[#1C1815] border-[#E9A870] border-2 rounded-xl overflow-hidden shadow-2xl backdrop-blur-sm">

              <CardHeader className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] p-4">

                <h3 className="text-lg font-bold text-black flex items-center gap-2 font-arial">

                  <History className="h-5 w-5" />

                  HISTÓRICO

                </h3>

              </CardHeader>

              <CardContent className="p-4 bg-[#1C1815] custom-scroll">

                <div className="space-y-3 max-h-96 overflow-y-auto custom-scroll">

                  {ticketHistory.map((item, index) => (

                    <div

                      key={item.id}

                      className="flex items-start gap-3 p-3 bg-[#121212] rounded-lg border border-[#E9A870]/20 hover:bg-[#262626] transition-colors"

                    >

                      <div className="w-2 h-2 bg-[#E9A870] rounded-full mt-2 flex-shrink-0"></div>

                      <div className="flex-1">

                        <div className="flex items-center justify-between mb-1">

                          <h4 className="font-bold text-white text-sm font-arial">{item.acao}</h4>

                          <span className="text-sm text-gray-400 font-arial">{item.data}</span>

                        </div>

                        <p className="text-sm text-gray-300 mb-1 font-arial">{item.detalhes}</p>

                        <p className="text-sm text-[#E9A870] font-arial">por {item.usuario}</p>

                      </div>

                    </div>

                  ))}

                </div>

              </CardContent>

            </Card>

          </div>

        </div>

      </div>

    </div>

  )

}

frontend/app/registro/page.tsx:

"use client"

import type React from "react"

import { useState } from "react"

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

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

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

import { Textarea } from "@/components/ui/textarea"

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

import { Menu, ChevronLeft, Check, Home, ChevronRight } from "lucide-react"

import { SidebarMenu } from "@/components/sidebar-menu"

import { toast, Toaster } from "sonner"

interface RegistroProps {

  onNavigate: (page: string, data?: any) => void

}

export default function Registro({ onNavigate }: RegistroProps) {

*const* [sidebarOpen, setSidebarOpen] = useState(false)

*const* [formData, setFormData] = useState({

    tombamento: "",

    ambiente: "",

    tipo: "",

    unidade: "",

    descricao: "",

    tecnico: "",

  })

*const* unidades = ["USF RIO DAS VELHAS II", "USF QUADROS III", "USF VIETNÃ", "USF CENTRO", "USF NORTE"]

*const* tecnicos = ["João Silva", "Maria Santos", "Pedro Costa", "Ana Lima", "Carlos Oliveira"]

*const* handleSubmit = (e: React.FormEvent) => {

    e.preventDefault()

*// Here you would normally submit the form data*

    console.log("Form submitted:", formData)

    toast.success("Chamado registrado com sucesso!", {

      duration: 3000,

      position: "top-right",

    })

    onNavigate("home")

  }

*const* isFormValid =

    formData.tombamento && formData.ambiente && formData.unidade && formData.descricao && formData.tecnico

*return* (

    <div className="min-h-screen bg-[#191818] text-white font-abel">

      {*/\* Toaster for notifications \*/*}

      <Toaster

        theme="dark"

        position="top-right"

        toastOptions={{

          style: {

            background: "#1C1815",

            border: "1px solid #E9A870",

            color: "#E0CAA5",

          },

        }}

      />

      {*/\* Sidebar Menu \*/*}

      <SidebarMenu

        isOpen={sidebarOpen}

        onClose={() => setSidebarOpen(false)}

        onNavigate={onNavigate}

        currentPage="registro"

      />

      {*/\* Header - Fixed/Sticky \*/*}

      <header className="sticky top-0 z-30 flex items-center justify-between px-6 py-4 bg-[#0C0C0C] border-b border-[#E9A870] shadow-lg backdrop-blur-sm">

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

          <Button

            variant="ghost"

            size="icon"

            className="text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

            onClick={() => setSidebarOpen(true)}

          >

            <Menu className="h-6 w-6" />

          </Button>

          <h1 className="text-[42px] font-abel text-[#E0CAA5] tracking-wider drop-shadow-sm">AGNUS</h1>

          {*/\* Breadcrumbs \*/*}

          <div className="flex items-center text-sm text-gray-400 ml-6">

            <Button

              variant="link"

              className="p-0 h-auto text-[#E0CAA5] hover:text-[#E9A870] transition-colors font-arial"

              onClick={() => onNavigate("home")}

            >

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

              <span>Início</span>

            </Button>

            <ChevronRight className="h-3 w-3 mx-2 text-gray-500" />

            <span className="text-gray-400 font-arial">Registro</span>

          </div>

        </div>

      </header>

      <div className="p-6">

        <div className="mb-6">

          <h2 className="text-[58px] font-afacad bg-gradient-to-r from-[#E9A870] via-[#E0CAA5] to-[#A8784F] bg-clip-text text-transparent font-semibold leading-tight drop-shadow-sm">

            REGISTRO

          </h2>

          <p className="text-[32px] font-afacad text-[#E0CAA5] leading-tight opacity-90">ABERTURA DE CHAMADOS</p>

        </div>

        {*/\* Centered Form Container \*/*}

        <div className="flex justify-center">

          <Card className="w-full max-w-5xl bg-[#1C1815] border-[#E9A870] border-2 rounded-xl overflow-hidden shadow-2xl backdrop-blur-sm hover:shadow-[#E9A870]/10 transition-all duration-300">

            <CardHeader className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] p-4">

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

                <h3 className="text-xl font-bold text-black font-arial">INFORMAÇÕES DE REGISTRO DE ENTRADA</h3>

                <Button

                  variant="ghost"

                  size="icon"

                  onClick={() => onNavigate("home")}

                  className="text-black hover:bg-black/10 rounded-lg transition-all duration-200 hover:scale-110"

                >

                  <ChevronLeft className="h-5 w-5" />

                </Button>

              </div>

            </CardHeader>

            <CardContent className="p-8 bg-[#1C1815] space-y-6 custom-scroll">

              <form onSubmit={handleSubmit} className="space-y-6">

                {*/\* First Row \*/*}

                <div className="grid grid-cols-1 md:grid-cols-3 gap-6">

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">

                      TOMBAMENTO <span className="text-red-400">\*</span>

                    </label>

                    <Input

                      placeholder="Ex: 202212345"

                      value={formData.tombamento}

                      onChange={(e) => setFormData({ **...**formData, tombamento: e.target.value })}

                      className="bg-[#121212] border-[#E9A870]/30 text-white placeholder:text-gray-500 focus:border-[#E9A870] font-abel h-12 text-base"

                    />

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">

                      AMBIENTE <span className="text-red-400">\*</span>

                    </label>

                    <Input

                      placeholder="Ex: Médico"

                      value={formData.ambiente}

                      onChange={(e) => setFormData({ **...**formData, ambiente: e.target.value })}

                      className="bg-[#121212] border-[#E9A870]/30 text-white placeholder:text-gray-500 focus:border-[#E9A870] font-abel h-12 text-base"

                    />

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">TIPO</label>

                    <Input

                      placeholder="Ex: TCORP"

                      value={formData.tipo}

                      onChange={(e) => setFormData({ **...**formData, tipo: e.target.value })}

                      className="bg-[#121212] border-[#E9A870]/30 text-white placeholder:text-gray-500 focus:border-[#E9A870] font-abel h-12 text-base"

                    />

                  </div>

                </div>

                {*/\* Second Row \*/*}

                <div className="space-y-2">

                  <label className="text-sm font-medium text-[#E9A870] font-arial">

                    UNIDADE <span className="text-red-400">\*</span>

                  </label>

                  <Select

                    value={formData.unidade}

                    onValueChange={(value) => setFormData({ **...**formData, unidade: value })}

                  >

                    <SelectTrigger className="bg-[#121212] border-[#E9A870]/30 text-white focus:border-[#E9A870] font-abel h-12 text-base">

                      <SelectValue placeholder="SELECIONE A UNIDADE" />

                    </SelectTrigger>

                    <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white custom-scroll">

                      {unidades.map((unidade) => (

                        <SelectItem key={unidade} value={unidade} className="focus:bg-[#E9A870]/20 font-arial">

                          {unidade}

                        </SelectItem>

                      ))}

                    </SelectContent>

                  </Select>

                </div>

                {*/\* Third Row \*/*}

                <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">

                  <div className="lg:col-span-2 space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">

                      DESCRIÇÃO DO DEFEITO <span className="text-red-400">\*</span>

                    </label>

                    <Textarea

                      placeholder="Ex: SEM APRESENTAR IMAGEM, PROVÁVEL DF NA MEMÓRIA RAM"

                      value={formData.descricao}

                      onChange={(e) => setFormData({ **...**formData, descricao: e.target.value })}

                      className="bg-[#121212] border-[#E9A870]/30 text-white placeholder:text-gray-500 focus:border-[#E9A870] min-h-[140px] font-arial resize-none text-base custom-scroll"

                    />

                  </div>

                  <div className="space-y-2">

                    <label className="text-sm font-medium text-[#E9A870] font-arial">

                      TÉCNICO <span className="text-red-400">\*</span>

                    </label>

                    <Select

                      value={formData.tecnico}

                      onValueChange={(value) => setFormData({ **...**formData, tecnico: value })}

                    >

                      <SelectTrigger className="bg-[#121212] border-[#E9A870]/30 text-white focus:border-[#E9A870] font-abel h-12 text-base">

                        <SelectValue placeholder="TÉCNICO" />

                      </SelectTrigger>

                      <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white custom-scroll">

                        {tecnicos.map((tecnico) => (

                          <SelectItem key={tecnico} value={tecnico} className="focus:bg-[#E9A870]/20 font-arial">

                            {tecnico}

                          </SelectItem>

                        ))}

                      </SelectContent>

                    </Select>

                  </div>

                </div>

                {*/\* Submit Button \*/*}

                <div className="flex justify-center pt-6 border-t border-[#E9A870]/20">

                  <Button

                    type="submit"

                    disabled={!isFormValid}

                    className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] hover:from-[#E0CAA5] hover:to-[#E9A870] text-black font-bold px-10 py-4 rounded-full shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105 disabled:opacity-50 disabled:cursor-not-allowed text-lg"

                  >

                    <Check className="h-6 w-6 mr-3" />

                    REGISTRAR CHAMADO

                  </Button>

                </div>

              </form>

            </CardContent>

          </Card>

        </div>

      </div>

    </div>

  )

}

frontend/app/tecnicos/page.tsx

"use client"

import React from "react"

import { useState } from "react"

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

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

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

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

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

import {

  AlertDialog,

  AlertDialogAction,

  AlertDialogCancel,

  AlertDialogContent,

  AlertDialogDescription,

  AlertDialogFooter,

  AlertDialogHeader,

  AlertDialogTitle,

  AlertDialogTrigger,

} from "@/components/ui/alert-dialog"

import { Menu, Plus, Filter, Edit, Trash2, Search, X, Phone, Mail, Check, Home, ChevronRight } from "lucide-react"

import { SidebarMenu } from "@/components/sidebar-menu"

import { cn } from "@/lib/utils"

import { toast, Toaster } from "sonner"

interface TecnicosProps {

  onNavigate: (page: string, data?: any) => void

  data?: any

}

*// Mock data for technicians*

*const* mockTecnicos = [

  {

    id: 1,

    nome: "JOÃO SILVA",

    atuacao: "INTERNO",

    telefone: "(81) 99999-1111",

    email: "joao.silva@saude.jaboatao.pe.gov.br",

    unidade: "USF RIO DAS VELHAS II",

    dataAdmissao: "2023-01-15",

    especialidades: ["Hardware", "Redes"],

    status: "ATIVO",

  },

  {

    id: 2,

    nome: "MARIA SANTOS",

    atuacao: "EXTERNO",

    telefone: "(81) 99999-2222",

    email: "maria.santos@empresa.com",

    unidade: "MÚLTIPLAS",

    dataAdmissao: "2023-03-20",

    especialidades: ["Software", "Sistemas"],

    status: "ATIVO",

  },

  {

    id: 3,

    nome: "PEDRO COSTA",

    atuacao: "INTERNO",

    telefone: "(81) 99999-3333",

    email: "pedro.costa@saude.jaboatao.pe.gov.br",

    unidade: "USF QUADROS III",

    dataAdmissao: "2022-11-10",

    especialidades: ["Hardware", "Manutenção"],

    status: "ATIVO",

  },

  {

    id: 4,

    nome: "ANA LIMA",

    atuacao: "INTERNO",

    telefone: "(81) 99999-4444",

    email: "ana.lima@saude.jaboatao.pe.gov.br",

    unidade: "USF VIETNÃ",

    dataAdmissao: "2023-06-01",

    especialidades: ["Software", "Suporte"],

    status: "ATIVO",

  },

  {

    id: 5,

    nome: "FELIPE FILIPE FELIPE",

    atuacao: "EXTERNO",

    telefone: "(81) 99999-5555",

    email: "felipe@empresa.com",

    unidade: "MÚLTIPLAS",

    dataAdmissao: "2023-08-15",

    especialidades: ["Redes", "Infraestrutura"],

    status: "ATIVO",

  },

]

export default function Tecnicos({ onNavigate, data }: TecnicosProps) {

*const* [sidebarOpen, setSidebarOpen] = useState(false)

*const* [activeTab, setActiveTab] = useState("TODOS")

*const* [searchOpen, setSearchOpen] = useState(false)

*const* [searchQuery, setSearchQuery] = useState("")

*const* [filteredTecnicos, setFilteredTecnicos] = useState(mockTecnicos)

*const* [isAddDialogOpen, setIsAddDialogOpen] = useState(false)

*const* [isEditDialogOpen, setIsEditDialogOpen] = useState(false)

*const* [deleteDialogOpen, setDeleteDialogOpen] = useState(false)

*const* [selectedTecnico, setSelectedTecnico] = useState<any>(null)

*const* [tecnicoToDelete, setTecnicoToDelete] = useState<any>(null)

*const* [formData, setFormData] = useState({

    nome: "",

    atuacao: "",

    telefone: "",

    email: "",

    unidade: "",

    especialidades: "",

  })

*const* tabs = ["TODOS", "INTERNOS", "EXTERNOS"]

*const* unidades = ["USF RIO DAS VELHAS II", "USF QUADROS III", "USF VIETNÃ", "USF CENTRO", "MÚLTIPLAS"]

*// Filter technicians based on search and tab*

  React.useEffect(() => {

*let* filtered = mockTecnicos

*// Apply search filter*

    if (searchQuery) {

      filtered = filtered.filter(

        (tecnico) =>

          tecnico.nome.toLowerCase().includes(searchQuery.toLowerCase()) ||

          tecnico.email.toLowerCase().includes(searchQuery.toLowerCase()) ||

          tecnico.unidade.toLowerCase().includes(searchQuery.toLowerCase()) ||

          tecnico.atuacao.toLowerCase().includes(searchQuery.toLowerCase()),

      )

    }

*// Apply tab filter*

    if (activeTab === "INTERNOS") {

      filtered = filtered.filter((tecnico) => tecnico.atuacao === "INTERNO")

    } else if (activeTab === "EXTERNOS") {

      filtered = filtered.filter((tecnico) => tecnico.atuacao === "EXTERNO")

    }

    setFilteredTecnicos(filtered)

  }, [searchQuery, activeTab])

*// Highlight selected technician if coming from another page*

  React.useEffect(() => {

    if (data?.selectedTecnico) {

*const* tecnico = mockTecnicos.find((t) => t.nome === data.selectedTecnico)

      if (tecnico) {

        setSelectedTecnico(tecnico)

*// Auto-scroll to technician or highlight it*

      }

    }

  }, [data])

*const* handleAddTecnico = (e: React.FormEvent) => {

    e.preventDefault()

*// Here you would normally submit to API*

    console.log("Adding technician:", formData)

    toast.success("Técnico adicionado com sucesso!", {

      duration: 3000,

      position: "top-right",

    })

    setIsAddDialogOpen(false)

    setFormData({

      nome: "",

      atuacao: "",

      telefone: "",

      email: "",

      unidade: "",

      especialidades: "",

    })

  }

*const* handleEditTecnico = (tecnico: any) => {

    setSelectedTecnico(tecnico)

    setFormData({

      nome: tecnico.nome,

      atuacao: tecnico.atuacao,

      telefone: tecnico.telefone,

      email: tecnico.email,

      unidade: tecnico.unidade,

      especialidades: tecnico.especialidades.join(", "),

    })

    setIsEditDialogOpen(true)

  }

*const* handleUpdateTecnico = (e: React.FormEvent) => {

    e.preventDefault()

*// Here you would normally submit to API*

    console.log("Updating technician:", formData)

    toast.success("Técnico atualizado com sucesso!", {

      duration: 3000,

      position: "top-right",

    })

    setIsEditDialogOpen(false)

    setSelectedTecnico(null)

  }

*const* handleDeleteClick = (tecnico: any) => {

    setTecnicoToDelete(tecnico)

    setDeleteDialogOpen(true)

  }

*const* confirmDelete = () => {

    if (tecnicoToDelete) {

      toast.success(`Técnico ${tecnicoToDelete.nome} foi excluído com sucesso`, {

        duration: 3000,

        position: "top-right",

      })

      setDeleteDialogOpen(false)

      setTecnicoToDelete(null)

    }

  }

*const* isFormValid = formData.nome && formData.atuacao && formData.telefone && formData.email && formData.unidade

*return* (

    <div className="min-h-screen bg-[#191818] text-white font-abel">

      {*/\* Toaster for notifications \*/*}

      <Toaster

        theme="dark"

        position="top-right"

        toastOptions={{

          style: {

            background: "#1C1815",

            border: "1px solid #E9A870",

            color: "#E0CAA5",

          },

        }}

      />

      {*/\* Sidebar Menu \*/*}

      <SidebarMenu

        isOpen={sidebarOpen}

        onClose={() => setSidebarOpen(false)}

        onNavigate={onNavigate}

        currentPage="tecnicos"

      />

      {*/\* Header - Fixed/Sticky \*/*}

      <header className="sticky top-0 z-30 flex items-center justify-between px-6 py-4 bg-[#0C0C0C] border-b border-[#E9A870] shadow-lg backdrop-blur-sm">

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

          <Button

            variant="ghost"

            size="icon"

            className="text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

            onClick={() => setSidebarOpen(true)}

          >

            <Menu className="h-6 w-6" />

          </Button>

          <h1 className="text-[42px] font-abel text-[#E0CAA5] tracking-wider drop-shadow-sm">AGNUS</h1>

          {*/\* Breadcrumbs \*/*}

          <div className="flex items-center text-sm text-gray-400 ml-6">

            <Button

              variant="link"

              className="p-0 h-auto text-[#E0CAA5] hover:text-[#E9A870] transition-colors font-arial"

              onClick={() => onNavigate("home")}

            >

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

              <span>Início</span>

            </Button>

            <ChevronRight className="h-3 w-3 mx-2 text-gray-500" />

            <span className="text-gray-400 font-arial">Técnicos</span>

          </div>

        </div>

      </header>

      <div className="p-6 custom-scroll">

        <div className="mb-6">

          <div className="flex items-center gap-4 mb-2">

            <h2 className="text-[58px] font-afacad bg-gradient-to-r from-[#E9A870] via-[#E0CAA5] to-[#A8784F] bg-clip-text text-transparent font-semibold leading-tight drop-shadow-sm">

              TÉCNICOS

            </h2>

            {*/\* Add Technician Dialog \*/*}

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

              <DialogTrigger asChild>

                <Button

                  size="icon"

                  className="bg-gradient-to-r from-[#3D3024] to-[#2a2a2a] hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 rounded-full h-10 w-10 shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105 border border-[#E9A870]/20"

                >

                  <Plus className="h-5 w-5 text-[#E9A870]" />

                </Button>

              </DialogTrigger>

              <DialogContent className="bg-gradient-to-b from-[#1C1815] to-[#0C0C0C] border-2 border-[#E9A870] text-white max-w-2xl font-abel shadow-2xl backdrop-blur-sm">

                <DialogHeader>

                  <DialogTitle className="text-[#E9A870] text-2xl font-bold bg-gradient-to-r from-[#E9A870] to-[#A8784F] bg-clip-text text-transparent">

                    ADICIONAR TÉCNICO

                  </DialogTitle>

                  <div className="w-full h-px bg-gradient-to-r from-transparent via-[#E9A870] to-transparent mt-4"></div>

                </DialogHeader>

                <form onSubmit={handleAddTecnico} className="space-y-6 custom-scroll max-h-[70vh] overflow-y-auto p-1">

                  <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                    <div className="space-y-2">

                      <label className="text-sm font-medium text-[#E9A870]">

                        NOME COMPLETO <span className="text-red-400">\*</span>

                      </label>

                      <Input

                        placeholder="Nome do técnico"

                        value={formData.nome}

                        onChange={(e) => setFormData({ **...**formData, nome: e.target.value })}

                        className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                      />

                    </div>

                    <div className="space-y-2">

                      <label className="text-sm font-medium text-[#E9A870]">

                        ATUAÇÃO <span className="text-red-400">\*</span>

                      </label>

                      <Select

                        value={formData.atuacao}

                        onValueChange={(value) => setFormData({ **...**formData, atuacao: value })}

                      >

                        <SelectTrigger className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200">

                          <SelectValue placeholder="Selecione" />

                        </SelectTrigger>

                        <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white custom-scroll">

                          <SelectItem value="INTERNO">INTERNO</SelectItem>

                          <SelectItem value="EXTERNO">EXTERNO</SelectItem>

                        </SelectContent>

                      </Select>

                    </div>

                    <div className="space-y-2">

                      <label className="text-sm font-medium text-[#E9A870]">

                        TELEFONE <span className="text-red-400">\*</span>

                      </label>

                      <Input

                        placeholder="(81) 99999-9999"

                        value={formData.telefone}

                        onChange={(e) => setFormData({ **...**formData, telefone: e.target.value })}

                        className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                      />

                    </div>

                    <div className="space-y-2">

                      <label className="text-sm font-medium text-[#E9A870]">

                        EMAIL <span className="text-red-400">\*</span>

                      </label>

                      <Input

                        type="email"

                        placeholder="email@exemplo.com"

                        value={formData.email}

                        onChange={(e) => setFormData({ **...**formData, email: e.target.value })}

                        className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                      />

                    </div>

                  </div>

                  <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                    <div className="space-y-2">

                      <label className="text-sm font-medium text-[#E9A870]">

                        UNIDADE <span className="text-red-400">\*</span>

                      </label>

                      <Select

                        value={formData.unidade}

                        onValueChange={(value) => setFormData({ **...**formData, unidade: value })}

                      >

                        <SelectTrigger className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200">

                          <SelectValue placeholder="Selecione a unidade" />

                        </SelectTrigger>

                        <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white custom-scroll">

                          {unidades.map((unidade) => (

                            <SelectItem key={unidade} value={unidade}>

                              {unidade}

                            </SelectItem>

                          ))}

                        </SelectContent>

                      </Select>

                    </div>

                    <div className="space-y-2">

                      <label className="text-sm font-medium text-[#E9A870]">ESPECIALIDADES</label>

                      <Input

                        placeholder="Hardware, Software, Redes..."

                        value={formData.especialidades}

                        onChange={(e) => setFormData({ **...**formData, especialidades: e.target.value })}

                        className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                      />

                    </div>

                  </div>

                  <div className="flex justify-end gap-4 pt-6 border-t border-[#E9A870]/20">

                    <Button

                      type="button"

                      variant="outline"

                      onClick={() => setIsAddDialogOpen(false)}

                      className="border-[#E9A870]/40 text-[#E0CAA5] hover:bg-[#E9A870]/10 hover:border-[#E9A870] transition-all duration-200 font-abel"

                    >

                      CANCELAR

                    </Button>

                    <Button

                      type="submit"

                      disabled={!isFormValid}

                      className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] hover:from-[#E0CAA5] hover:to-[#E9A870] text-[#0C0C0C] font-bold disabled:opacity-50 shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105"

                    >

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

                      ADICIONAR

                    </Button>

                  </div>

                </form>

              </DialogContent>

            </Dialog>

          </div>

          <p className="text-[32px] font-afacad text-[#E0CAA5] leading-tight opacity-90">INTERNOS E EXTERNOS</p>

          {*/\* Filter and Tabs \*/*}

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

            <Button

              variant="outline"

              size="icon"

              className="bg-[#3D3024] border-[#E9A870]/20 hover:bg-[#4D4034] rounded-lg h-10 w-10 shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105"

            >

              <Filter className="h-5 w-5 text-[#E9A870]" />

            </Button>

            {*/\* Expandable Search \*/*}

            <div

              className={cn(

                "flex items-center transition-all duration-300 overflow-hidden",

                searchOpen ? "w-64" : "w-10",

              )}

            >

              <div className="flex items-center bg-[#3D3024] rounded-lg overflow-hidden shadow-lg border border-[#E9A870]/20">

                <Button

                  variant="ghost"

                  size="icon"

                  className="h-10 w-10 bg-transparent hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

                  onClick={() => setSearchOpen(!searchOpen)}

                >

                  {searchOpen ? (

                    <X className="h-5 w-5 text-[#E9A870]" />

                  ) : (

                    <Search className="h-5 w-5 text-[#E9A870]" />

                  )}

                </Button>

                <Input

                  type="text"

                  placeholder="Pesquisar técnicos..."

                  value={searchQuery}

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

                  className={cn(

                    "h-10 bg-transparent border-0 focus-visible:ring-0 text-white placeholder:text-gray-400",

                    "transition-all duration-300",

                    searchOpen ? "w-full opacity-100" : "w-0 opacity-0 p-0",

                  )}

                />

              </div>

            </div>

            <div className="flex gap-4">

              {tabs.map((tab) => (

                <button

                  key={tab}

                  onClick={() => setActiveTab(tab)}

                  className={`text-base font-medium px-4 py-2 rounded-lg transition-all duration-300 hover:scale-105 ${

                    activeTab === tab

                      ? "text-[#E9A870] bg-gradient-to-r from-[#1C1815] to-[#1C1815] shadow-md border border-[#E9A870]/20 scale-105"

                      : "text-[#E0CAA5] hover:bg-gradient-to-r hover:from-gray-800/30 hover:to-gray-700/30 hover:shadow-sm"

                  }`}

                >

                  {tab}

                </button>

              ))}

            </div>

          </div>

        </div>

        {*/\* Table \*/*}

        <Card className="bg-[#1C1815] border-[#E9A870] border-2 rounded-xl overflow-hidden shadow-2xl backdrop-blur-sm">

          <CardHeader className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] p-0">

            <div className="grid grid-cols-4 gap-6 px-6 py-4 text-black font-bold text-lg">

              <div className="text-center">ID</div>

              <div>NOME</div>

              <div>ATUAÇÃO</div>

              <div>AÇÕES</div>

            </div>

          </CardHeader>

          <CardContent className="p-0 min-h-[350px] bg-[#1C1815] custom-scroll">

            {filteredTecnicos.length > 0 ? (

              filteredTecnicos.map((tecnico, index) => (

                <div

                  key={tecnico.id}

                  className={`grid grid-cols-4 gap-6 px-6 py-4 border-b border-[#E9A870]/30 items-center hover:bg-gradient-to-r hover:from-gray-800/20 hover:to-gray-700/20 transition-all duration-300 group hover:scale-[1.01] hover:shadow-lg ${

                    selectedTecnico?.id === tecnico.id ? "bg-[#E9A870]/10 border-[#E9A870]/50" : ""

                  }`}

                  style={{ animationDelay: `${index \* 100}ms` }}

                >

                  <div className="text-gray-300 group-hover:text-white transition-colors text-center font-bold">

                    {tecnico.id}

                  </div>

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

                    <div className="text-gray-300 group-hover:text-white transition-colors font-bold">

                      {tecnico.nome}

                    </div>

                    <div className="flex items-center gap-2 text-sm text-gray-400">

                      <Mail className="h-3 w-3" />

                      {tecnico.email}

                    </div>

                    <div className="flex items-center gap-2 text-sm text-gray-400">

                      <Phone className="h-3 w-3" />

                      {tecnico.telefone}

                    </div>

                  </div>

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

                    <div

                      className={`inline-block px-3 py-1 rounded-full text-sm font-bold ${

                        tecnico.atuacao === "INTERNO"

                          ? "bg-blue-500/20 text-blue-300 border border-blue-500/30"

                          : "bg-purple-500/20 text-purple-300 border border-purple-500/30"

                      }`}

                    >

                      {tecnico.atuacao}

                    </div>

                    <div className="text-sm text-gray-400">{tecnico.especialidades.join(", ")}</div>

                  </div>

                  <div className="flex gap-3">

                    <Button

                      variant="ghost"

                      size="icon"

                      onClick={() => handleEditTecnico(tecnico)}

                      className="h-8 w-8 hover:bg-gradient-to-r hover:from-gray-700/50 hover:to-gray-600/50 transition-all duration-200 hover:scale-110"

                    >

                      <Edit className="h-4 w-4 text-[#E0CAA5]" />

                    </Button>

                    <AlertDialog open={deleteDialogOpen} onOpenChange={setDeleteDialogOpen}>

                      <AlertDialogTrigger asChild>

                        <Button

                          variant="ghost"

                          size="icon"

                          onClick={() => handleDeleteClick(tecnico)}

                          className="h-8 w-8 hover:bg-gradient-to-r hover:from-red-700/50 hover:to-red-600/50 transition-all duration-200 hover:scale-110"

                        >

                          <Trash2 className="h-4 w-4 text-red-400" />

                        </Button>

                      </AlertDialogTrigger>

                      <AlertDialogContent className="bg-[#1C1815] border-[#E9A870] text-white">

                        <AlertDialogHeader>

                          <AlertDialogTitle className="text-[#E9A870] text-xl font-abel">

                            Confirmar Exclusão

                          </AlertDialogTitle>

                          <AlertDialogDescription className="text-gray-300 font-arial">

                            Tem certeza que deseja excluir o técnico{" "}

                            <span className="font-bold text-white">{tecnicoToDelete?.nome}</span>?

                            <br />

                            <span className="text-red-400 text-sm">Esta ação não pode ser desfeita.</span>

                          </AlertDialogDescription>

                        </AlertDialogHeader>

                        <AlertDialogFooter>

                          <AlertDialogCancel className="bg-gray-600 hover:bg-gray-700 text-white border-gray-500">

                            Cancelar

                          </AlertDialogCancel>

                          <AlertDialogAction

                            onClick={confirmDelete}

                            className="bg-gradient-to-r from-red-600 to-red-700 hover:from-red-700 hover:to-red-800 text-white"

                          >

                            <Trash2 className="h-4 w-4 mr-2" />

                            Excluir

                          </AlertDialogAction>

                        </AlertDialogFooter>

                      </AlertDialogContent>

                    </AlertDialog>

                  </div>

                </div>

              ))

            ) : (

              <div className="flex justify-center items-center h-40 text-gray-400">Nenhum técnico encontrado</div>

            )}

          </CardContent>

        </Card>

      </div>

      {*/\* Edit Dialog \*/*}

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

        <DialogContent className="bg-gradient-to-b from-[#1C1815] to-[#0C0C0C] border-2 border-[#E9A870] text-white max-w-2xl font-abel shadow-2xl backdrop-blur-sm">

          <DialogHeader>

            <DialogTitle className="text-[#E9A870] text-2xl font-bold bg-gradient-to-r from-[#E9A870] to-[#A8784F] bg-clip-text text-transparent">

              EDITAR TÉCNICO

            </DialogTitle>

            <div className="w-full h-px bg-gradient-to-r from-transparent via-[#E9A870] to-transparent mt-4"></div>

          </DialogHeader>

          <form onSubmit={handleUpdateTecnico} className="space-y-6 custom-scroll max-h-[70vh] overflow-y-auto p-1">

            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

              <div className="space-y-2">

                <label className="text-sm font-medium text-[#E9A870]">

                  NOME COMPLETO <span className="text-red-400">\*</span>

                </label>

                <Input

                  placeholder="Nome do técnico"

                  value={formData.nome}

                  onChange={(e) => setFormData({ **...**formData, nome: e.target.value })}

                  className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                />

              </div>

              <div className="space-y-2">

                <label className="text-sm font-medium text-[#E9A870]">

                  ATUAÇÃO <span className="text-red-400">\*</span>

                </label>

                <Select

                  value={formData.atuacao}

                  onValueChange={(value) => setFormData({ **...**formData, atuacao: value })}

                >

                  <SelectTrigger className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200">

                    <SelectValue placeholder="Selecione" />

                  </SelectTrigger>

                  <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white custom-scroll">

                    <SelectItem value="INTERNO">INTERNO</SelectItem>

                    <SelectItem value="EXTERNO">EXTERNO</SelectItem>

                  </SelectContent>

                </Select>

              </div>

              <div className="space-y-2">

                <label className="text-sm font-medium text-[#E9A870]">

                  TELEFONE <span className="text-red-400">\*</span>

                </label>

                <Input

                  placeholder="(81) 99999-9999"

                  value={formData.telefone}

                  onChange={(e) => setFormData({ **...**formData, telefone: e.target.value })}

                  className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                />

              </div>

              <div className="space-y-2">

                <label className="text-sm font-medium text-[#E9A870]">

                  EMAIL <span className="text-red-400">\*</span>

                </label>

                <Input

                  type="email"

                  placeholder="email@exemplo.com"

                  value={formData.email}

                  onChange={(e) => setFormData({ **...**formData, email: e.target.value })}

                  className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                />

              </div>

            </div>

            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

              <div className="space-y-2">

                <label className="text-sm font-medium text-[#E9A870]">

                  UNIDADE <span className="text-red-400">\*</span>

                </label>

                <Select

                  value={formData.unidade}

                  onValueChange={(value) => setFormData({ **...**formData, unidade: value })}

                >

                  <SelectTrigger className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200">

                    <SelectValue placeholder="Selecione a unidade" />

                  </SelectTrigger>

                  <SelectContent className="bg-[#1C1815] border-[#E9A870]/30 text-white custom-scroll">

                    {unidades.map((unidade) => (

                      <SelectItem key={unidade} value={unidade}>

                        {unidade}

                      </SelectItem>

                    ))}

                  </SelectContent>

                </Select>

              </div>

              <div className="space-y-2">

                <label className="text-sm font-medium text-[#E9A870]">ESPECIALIDADES</label>

                <Input

                  placeholder="Hardware, Software, Redes..."

                  value={formData.especialidades}

                  onChange={(e) => setFormData({ **...**formData, especialidades: e.target.value })}

                  className="bg-gradient-to-r from-[#121212] to-[#0C0C0C] border-[#E9A870]/40 text-white placeholder:text-gray-400 focus:border-[#E9A870] focus:ring-1 focus:ring-[#E9A870]/50 font-abel transition-all duration-200"

                />

              </div>

            </div>

            <div className="flex justify-end gap-4 pt-6 border-t border-[#E9A870]/20">

              <Button

                type="button"

                variant="outline"

                onClick={() => setIsEditDialogOpen(false)}

                className="bg-gradient-to-r from-[#2A231E] to-[#1C1815] border-[#E9A870]/40 text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#3D3024] hover:to-[#2A231E] hover:border-[#E9A870] transition-all duration-200 font-abel"

              >

                CANCELAR

              </Button>

              <Button

                type="submit"

                disabled={!isFormValid}

                className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] hover:from-[#E0CAA5] hover:to-[#E9A870] text-[#0C0C0C] font-bold disabled:opacity-50 shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105"

              >

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

                ATUALIZAR

              </Button>

            </div>

          </form>

        </DialogContent>

      </Dialog>

    </div>

  )

}

frontend/app/ClientLayout.tsx:

"use client"

import type React from "react"

import { Abel, Afacad } from "next/font/google"

import "./globals.css"

import { useState } from "react"

import Dashboard from "./page"

import Registro from "./registro/page"

import DetalhesRegistro from "./detalhes/page"

import Tecnicos from "./tecnicos/page"

import DashboardPage from "./dashboard/page"

import { ThemeProvider } from "./contexts/ThemeContext"

*const* abel = Abel({

  weight: "400",

  subsets: ["latin"],

  variable: "--font-abel",

})

*const* afacad = Afacad({

  subsets: ["latin"],

  variable: "--font-afacad",

})

export default function ClientLayout({

  children,

}: {

  children: React.ReactNode

}) {

*const* [currentPage, setCurrentPage] = useState("home")

*const* [pageData, setPageData] = useState<any>(null)

*const* handleNavigate = (page: string, data?: any) => {

    setCurrentPage(page)

    setPageData(data)

  }

*const* renderPage = () => {

    switch (currentPage) {

      case "home":

*return* <Dashboard onNavigate={handleNavigate} />

      case "registro":

*return* <Registro onNavigate={handleNavigate} />

      case "detalhes":

*return* <DetalhesRegistro onNavigate={handleNavigate} data={pageData} />

      case "tecnicos":

*return* <Tecnicos onNavigate={handleNavigate} data={pageData} />

      case "dashboard":

*return* <DashboardPage onNavigate={handleNavigate} />

      default:

*return* <Dashboard onNavigate={handleNavigate} />

    }

  }

*return* (

    <html lang="en">

      <body className={`${abel.variable} ${afacad.variable} antialiased`}>

        <ThemeProvider>{renderPage()}</ThemeProvider>

      </body>

    </html>

  )

}

frontend/app/layout.tsx:

import type React from "react"

import type { Metadata } from "next"

export *const* metadata: Metadata = {

  title: "AGNUS",

  description: "Controle de Entradas e Saídas de Ativos da GTI",

}

import ClientLayout from "./ClientLayout"

export default function RootLayout({

  children,

}: {

  children: React.ReactNode

}) {

*return* <ClientLayout>{children}</ClientLayout>

}

frontend/app/page.tsx:

"use client"

import React, { useState, useRef, useEffect, useCallback, useMemo } from "react"

import debounce from "lodash.debounce"

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

import {

  Menu,

  Plus,

  Filter,

  Eye,

  Edit,

  Trash2,

  Search,

  Bell,

  Home,

  ChevronRight,

  X,

  Clock,

  AlertTriangle,

  CheckCircle,

  Check,

} from "lucide-react"

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

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

import {

  AlertDialog,

  AlertDialogAction,

  AlertDialogCancel,

  AlertDialogContent,

  AlertDialogDescription,

  AlertDialogFooter,

  AlertDialogHeader,

  AlertDialogTitle,

  AlertDialogTrigger,

} from "@/components/ui/alert-dialog"

import { cn } from "@/lib/utils"

import { SidebarMenu } from "@/components/sidebar-menu"

import { toast, Toaster } from "sonner"

*// Mock data for computers*

*const* mockComputers = [

  {

    id: "O1",

    tombamento: "202259123",

    status: "ENTRADA",

    statusColor: "red",

    unidade: "USF RIO DAS VELHAS II",

    priority: "high",

    timestamp: "2h atrás",

    tecnico: "João Silva",

    ambiente: "MÉDICO",

    tipo: "DESKTOP",

    descricao: "Computador não liga",

  },

  {

    id: "O2",

    tombamento: "202259124",

    status: "SAÍDA",

    statusColor: "green",

    unidade: "USF QUADROS III",

    priority: "normal",

    timestamp: "1h atrás",

    tecnico: "Maria Santos",

    ambiente: "RECEPÇÃO",

    tipo: "NOTEBOOK",

    descricao: "Problema no teclado",

  },

  {

    id: "O3",

    tombamento: "202259125",

    status: "PRONTO",

    statusColor: "blue",

    unidade: "USF VIETNÃ",

    priority: "normal",

    timestamp: "30min atrás",

    tecnico: "Pedro Costa",

    ambiente: "ENFERMARIA",

    tipo: "DESKTOP",

    descricao: "Lentidão no sistema",

  },

  {

    id: "O4",

    tombamento: "202259126",

    status: "ENTRADA",

    statusColor: "red",

    unidade: "USF RIO DAS VELHAS II",

    priority: "urgent",

    timestamp: "15min atrás",

    tecnico: "Ana Lima",

    ambiente: "LABORATÓRIO",

    tipo: "DESKTOP",

    descricao: "Tela azul frequente",

  },

]

interface DashboardProps {

  onNavigate: (page: string, data?: any) => void

}

export default function Dashboard({ onNavigate }: DashboardProps) {

*const* [activeTab, setActiveTab] = useState("TODOS")

*const* [sidebarOpen, setSidebarOpen] = useState(false)

*const* [searchOpen, setSearchOpen] = useState(false)

*const* [searchQuery, setSearchQuery] = useState("")

*const* [filteredComputers, setFilteredComputers] = useState(mockComputers)

*const* [isLoading, setIsLoading] = useState(false)

*const* [statusFilter, setStatusFilter] = useState<string | null>(null)

*const* [unidadeFilter, setUnidadeFilter] = useState<string | null>(null)

*const* [deleteDialogOpen, setDeleteDialogOpen] = useState(false)

*const* [computerToDelete, setComputerToDelete] = useState<any>(null)

*const* searchInputRef = useRef<HTMLInputElement>(null)

*// Notifications state*

*const* [notifications, setNotifications] = useState([

    {

      id: 1,

      title: "Novo computador adicionado",

      time: "Agora mesmo",

      read: false,

      type: "info",

    },

    {

      id: 2,

      title: "Manutenção concluída",

      time: "5 minutos atrás",

      read: false,

      type: "success",

    },

    {

      id: 3,

      title: "Alerta de sistema",

      time: "1 hora atrás",

      read: true,

      type: "warning",

    },

    {

      id: 4,

      title: "Técnico João Silva finalizou reparo",

      time: "2 horas atrás",

      read: false,

      type: "success",

    },

    {

      id: 5,

      title: "USF Centro com alta demanda",

      time: "3 horas atrás",

      read: true,

      type: "warning",

    },

  ])

*const* tabs = ["TODOS", "RECOLHIDOS", "ENTREGUES"]

*const* readyItems = [

    { text: "MÉDICO - USF RIO DAS VELHAS II", id: "O5", tombamento: "202259127", timestamp: "Pronto há 2h" },

    { text: "ENFERMEIRA - USF QUADROS III", id: "O6", tombamento: "202259128", timestamp: "Pronto há 1h" },

    { text: "RECEPÇÃO - USF VIETNÃ", id: "O7", tombamento: "202259129", timestamp: "Pronto há 30min" },

  ]

*// Handle search*

  useEffect(() => {

    if (searchOpen && searchInputRef.current) {

      searchInputRef.current.focus()

    }

  }, [searchOpen])

*// Debounced search handler*

*const* [debouncedSearch, setDebouncedSearch] = useState("")

*// Debounce searchQuery updates*

*const* debouncedSetSearch = useMemo(

    () => debounce((value: string) => setDebouncedSearch(value), 300),

    [],

  )

  useEffect(() => {

    debouncedSetSearch(searchQuery)

*return* () => debouncedSetSearch.cancel()

  }, [searchQuery, debouncedSetSearch])

*// Memoized handlers*

*const* handleRowClick = useCallback(

    (computer: any) => {

      onNavigate("detalhes", computer)

    },

    [onNavigate],

  )

*const* handleTecnicoClick = useCallback(

    (tecnico: string, e: React.MouseEvent) => {

      e.stopPropagation()

      onNavigate("tecnicos", { selectedTecnico: tecnico })

    },

    [onNavigate],

  )

*const* handleDeleteClick = useCallback(

    (computer: any, e: React.MouseEvent) => {

      e.stopPropagation()

      setComputerToDelete(computer)

      setDeleteDialogOpen(true)

    },

    [],

  )

*// Filter computers based on search, status, and unidade*

  useEffect(() => {

    if (debouncedSearch || statusFilter || unidadeFilter || activeTab !== "TODOS") {

      setIsLoading(true)

*const* timer = setTimeout(() => {

*let* filtered = mockComputers

*// Apply search filter*

        if (debouncedSearch) {

          filtered = filtered.filter(

            (computer) =>

              computer.id.toLowerCase().includes(debouncedSearch.toLowerCase()) ||

              computer.tombamento.toLowerCase().includes(debouncedSearch.toLowerCase()) ||

              computer.unidade.toLowerCase().includes(debouncedSearch.toLowerCase()) ||

              computer.status.toLowerCase().includes(debouncedSearch.toLowerCase()) ||

              computer.tecnico.toLowerCase().includes(debouncedSearch.toLowerCase()),

          )

        }

*// Apply status filter*

        if (statusFilter) {

          filtered = filtered.filter((computer) => computer.status === statusFilter)

        }

*// Apply unidade filter*

        if (unidadeFilter) {

          filtered = filtered.filter((computer) => computer.unidade === unidadeFilter)

        }

*// Apply tab filter*

        if (activeTab === "RECOLHIDOS") {

          filtered = filtered.filter((computer) => computer.status === "ENTRADA")

        } else if (activeTab === "ENTREGUES") {

          filtered = filtered.filter((computer) => computer.status === "SAÍDA")

        }

        setFilteredComputers(filtered)

        setIsLoading(false)

      }, 0)

*return* () => clearTimeout(timer)

    } else {

      setFilteredComputers(mockComputers)

      setIsLoading(false)

    }

  }, [debouncedSearch, statusFilter, unidadeFilter, activeTab])

*// Notification functions*

*const* markAllAsRead = () => {

    setNotifications((prev) => prev.map((notification) => ({ **...**notification, read: true })))

    toast.success("Todas as notificações foram marcadas como lidas", {

      duration: 3000,

      position: "top-right",

    })

  }

*const* clearAllNotifications = () => {

    setNotifications([])

    toast.success("Todas as notificações foram removidas", {

      duration: 3000,

      position: "top-right",

    })

  }

*const* markAsRead = (id: number) => {

    setNotifications((prev) =>

      prev.map((notification) => (notification.id === id ? { **...**notification, read: true } : notification)),

    )

  }

*// Get unread notifications count*

*const* unreadCount = notifications.filter((n) => !n.read).length

*// Get status LED color and icon*

*const* getStatusIndicator = (status: string, priority?: string) => {

*const* baseClasses = "w-3 h-3 rounded-full shadow-sm"

*let* ledColor = ""

*let* icon = null

*const* pulseClass = ""

    switch (status) {

      case "ENTRADA":

        ledColor = "bg-red-500"

        icon = <Clock className="h-3 w-3 text-red-400" />

        break

      case "SAÍDA":

        ledColor = "bg-green-500"

        icon = <CheckCircle className="h-3 w-3 text-green-400" />

        break

      case "PRONTO":

        ledColor = "bg-blue-600"

        icon = <AlertTriangle className="h-3 w-3 text-blue-400" />

        break

      default:

        ledColor = "bg-gray-500"

    }

*return* { ledColor, icon, pulseClass, baseClasses }

  }

*const* getNotificationIcon = (type: string) => {

    switch (type) {

      case "success":

*return* <CheckCircle className="h-4 w-4 text-green-400" />

      case "warning":

*return* <AlertTriangle className="h-4 w-4 text-yellow-400" />

      case "info":

*return* <Bell className="h-4 w-4 text-blue-400" />

      default:

*return* <Bell className="h-4 w-4 text-gray-400" />

    }

  }

*// Clear all filters*

*const* clearFilters = () => {

    setStatusFilter(null)

    setUnidadeFilter(null)

    setSearchQuery("")

    setActiveTab("TODOS")

  }

*// Memoized TableSkeleton*

*const* TableSkeletonComponent = () => (

    <div className="space-y-4 p-6">

      {Array.from({ length: 4 }).map((\_, i) => (

        <div key={i} className="grid grid-cols-5 gap-6 items-center animate-pulse">

          <div className="h-4 bg-gradient-to-r from-gray-700 to-gray-600 rounded"></div>

          <div className="h-4 bg-gradient-to-r from-gray-700 to-gray-600 rounded"></div>

          <div className="h-4 bg-gradient-to-r from-gray-700 to-gray-600 rounded"></div>

          <div className="h-4 bg-gradient-to-r from-gray-700 to-gray-600 rounded"></div>

          <div className="flex gap-2">

            <div className="h-6 w-6 bg-gradient-to-r from-gray-700 to-gray-600 rounded"></div>

            <div className="h-6 w-6 bg-gradient-to-r from-gray-700 to-gray-600 rounded"></div>

            <div className="h-6 w-6 bg-gradient-to-r from-gray-700 to-gray-600 rounded"></div>

          </div>

        </div>

      ))}

    </div>

  )

*const* TableSkeleton = useMemo(() => React.memo(TableSkeletonComponent), [])

*const* confirmDelete = () => {

    if (computerToDelete) {

      toast.success(`Registro ${computerToDelete.tombamento} foi excluído com sucesso`, {

        duration: 3000,

        position: "top-right",

      })

      setDeleteDialogOpen(false)

      setComputerToDelete(null)

    }

  }

*// Memoize sidebar handlers to avoid unnecessary re-renders*

*const* handleSidebarClose = useCallback(() => setSidebarOpen(false), [])

*const* handleSidebarNavigate = useCallback((page: string, data?: any) => onNavigate(page, data), [onNavigate])

*return* (

    <div className="min-h-screen bg-[#191818] text-white transition-all duration-500 flex flex-col">

      {*/\* Toaster for notifications \*/*}

      <Toaster

        theme="dark"

        position="top-right"

        toastOptions={{

          style: {

            background: "#1C1815",

            border: "1px solid #E9A870",

            color: "#E0CAA5",

          },

        }}

      />

      {*/\* Sidebar Menu \*/*}

      <SidebarMenu

        isOpen={sidebarOpen}

        onClose={handleSidebarClose}

        onNavigate={handleSidebarNavigate}

        currentPage="home"

      />

      {*/\* Header - Fixed/Sticky \*/*}

      <header className="sticky top-0 z-30 flex items-center justify-between px-6 py-4 bg-[#0C0C0C] border-b border-[#E9A870] shadow-lg backdrop-blur-sm transition-all duration-300">

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

          <Button

            variant="ghost"

            size="icon"

            className="text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

            onClick={() => setSidebarOpen(true)}

          >

            <Menu className="h-6 w-6" />

          </Button>

          <h1 className="text-[42px] font-abel text-[#E0CAA5] tracking-wider drop-shadow-sm hover:drop-shadow-lg transition-all duration-300">

            AGNUS

          </h1>

          {*/\* Breadcrumbs \*/*}

          <div className="flex items-center text-sm text-gray-400 ml-6">

            <Button

              variant="link"

              className="p-0 h-auto text-[#E0CAA5] hover:text-[#E9A870] transition-colors font-arial"

            >

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

              <span>Início</span>

            </Button>

            <ChevronRight className="h-3 w-3 mx-2 text-gray-500" />

            <span className="text-gray-400 font-arial">Computadores</span>

          </div>

        </div>

        {*/\* Notifications \*/*}

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

          <DropdownMenu>

            <DropdownMenuTrigger asChild>

              <Button

                variant="ghost"

                size="icon"

                className="relative text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

              >

                <Bell className="h-5 w-5" />

                {unreadCount > 0 && (

                  <span className="absolute -top-1 -right-1 w-5 h-5 bg-gradient-to-r from-red-500 to-red-600 rounded-full text-sm font-bold flex items-center justify-center animate-pulse">

                    {unreadCount}

                  </span>

                )}

              </Button>

            </DropdownMenuTrigger>

            <DropdownMenuContent

              align="end"

              className="w-80 bg-gradient-to-b from-[#1C1815] to-[#1C1815] border-[#3D3024] text-white shadow-xl backdrop-blur-sm animate-in slide-in-from-top-2 duration-200 max-h-96 overflow-y-auto custom-scroll"

            >

              <div className="flex items-center justify-between px-4 py-3 border-b border-[#3D3024]">

                <span className="font-medium font-arial">Notificações</span>

                {unreadCount > 0 && (

                  <span className="text-sm bg-gradient-to-r from-[#E9A870] to-[#A8784F] text-black px-2 py-1 rounded-full font-arial">

                    {unreadCount} novas

                  </span>

                )}

              </div>

              {*/\* Action Buttons - Icons Only \*/*}

              {notifications.length > 0 && (

                <div className="flex justify-center gap-4 px-4 py-3 border-b border-[#3D3024]">

                  <Button

                    variant="ghost"

                    size="icon"

                    onClick={markAllAsRead}

                    className="h-8 w-8 text-[#E9A870] hover:bg-[#E9A870]/10 rounded-full"

                    disabled={unreadCount === 0}

                    title="Marcar todas como lidas"

                  >

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

                  </Button>

                  <Button

                    variant="ghost"

                    size="icon"

                    onClick={clearAllNotifications}

                    className="h-8 w-8 text-red-400 hover:bg-red-400/10 rounded-full"

                    title="Limpar notificações"

                  >

                    <Trash2 className="h-4 w-4" />

                  </Button>

                </div>

              )}

              {notifications.length > 0 ? (

                notifications.map((notification) => (

                  <DropdownMenuItem

                    key={notification.id}

                    className={`px-4 py-3 focus:bg-gradient-to-r focus:from-[#3D3024] focus:to-[#2a2a2a] focus:text-white transition-all duration-200 hover:translate-x-1 cursor-pointer ${

                      !notification.read ? "bg-[#E9A870]/5 border-l-2 border-l-[#E9A870]" : ""

                    }`}

                    onClick={() => markAsRead(notification.id)}

                  >

                    <div className="flex items-start gap-3 w-full">

                      {getNotificationIcon(notification.type)}

                      <div className="flex-1">

                        <div className="flex items-center justify-between mb-1">

                          <span

                            className={`font-medium font-arial text-sm ${!notification.read ? "text-white" : "text-gray-300"}`}

                          >

                            {notification.title}

                          </span>

                          {!notification.read && (

                            <div className="w-2 h-2 bg-[#E9A870] rounded-full flex-shrink-0"></div>

                          )}

                        </div>

                        <div className="text-sm text-gray-400 font-arial">{notification.time}</div>

                      </div>

                    </div>

                  </DropdownMenuItem>

                ))

              ) : (

                <div className="px-4 py-8 text-center text-gray-400">

                  <Bell className="h-8 w-8 mx-auto mb-2 opacity-50" />

                  <p className="text-sm font-arial">Nenhuma notificação</p>

                </div>

              )}

              {notifications.length > 0 && (

                <div className="p-2 text-center border-t border-[#3D3024]">

                  <Button

                    variant="link"

                    className="text-[#E9A870] hover:text-[#E0CAA5] text-sm w-full transition-colors font-arial"

                  >

                    Ver todas as notificações

                  </Button>

                </div>

              )}

            </DropdownMenuContent>

          </DropdownMenu>

        </div>

      </header>

      <div className="flex flex-1 min-h-0">

        {*/\* Main Content - Left Side \*/*}

        <div className="flex-1 flex flex-col min-h-0">

          <div className="p-6 pb-0">

            <div className="mb-6">

              <div className="flex items-center gap-4 mb-2">

                <h2 className="text-[58px] font-afacad bg-gradient-to-r from-[#E9A870] via-[#E0CAA5] to-[#A8784F] bg-clip-text text-transparent font-semibold leading-tight drop-shadow-sm">

                  EQUIPAMENTOS

                </h2>

                {*/\* Plus Button to Register Page \*/*}

                <Button

                  size="icon"

                  onClick={() => onNavigate("registro")}

                  className="bg-gradient-to-r from-[#3D3024] to-[#2a2a2a] hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 rounded-full h-10 w-10 shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105 border border-[#E9A870]/20"

                >

                  <Plus className="h-5 w-5 text-[#E9A870]" />

                </Button>

              </div>

              <p className="text-[32px] font-afacad text-[#E0CAA5] mb-6 leading-tight opacity-90 hover:opacity-100 transition-opacity duration-300">

                MONITORAMENTO

              </p>

              {*/\* Filter and Tabs \*/*}

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

                <DropdownMenu>

                  <DropdownMenuTrigger asChild>

                    <Button

                      variant="outline"

                      size="icon"

                      className="bg-[#3D3024] border-[#E9A870]/20 hover:bg-[#4D4034] rounded-lg h-10 w-10 shadow-lg hover:shadow-xl transition-all duration-200 hover:scale-105"

                    >

                      <Filter className="h-5 w-5 text-[#E9A870]" />

                    </Button>

                  </DropdownMenuTrigger>

                  <DropdownMenuContent

                    align="start"

                    className="w-56 bg-gradient-to-b from-[#1C1815] to-[#1C1815] border-[#3D3024] text-white shadow-xl backdrop-blur-sm animate-in slide-in-from-left-2 duration-200 custom-scroll"

                  >

                    <div className="px-3 py-2 text-sm font-medium border-b border-[#3D3024] font-arial">

                      Filtrar por

                    </div>

                    <DropdownMenuItem

                      className="focus:bg-gradient-to-r focus:from-[#3D3024] focus:to-[#2a2a2a] focus:text-white transition-all duration-200 hover:translate-x-1 font-arial"

                      onClick={() => setStatusFilter("ENTRADA")}

                    >

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

                        <span>Status: ENTRADA</span>

                        <div className="w-3 h-3 bg-red-500 rounded-full"></div>

                      </div>

                    </DropdownMenuItem>

                    <DropdownMenuItem

                      className="focus:bg-gradient-to-r focus:from-[#3D3024] focus:to-[#2a2a2a] focus:text-white transition-all duration-200 hover:translate-x-1 font-arial"

                      onClick={() => setStatusFilter("SAÍDA")}

                    >

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

                        <span>Status: SAÍDA</span>

                        <div className="w-3 h-3 bg-green-500 rounded-full"></div>

                      </div>

                    </DropdownMenuItem>

                    <DropdownMenuItem

                      className="focus:bg-gradient-to-r focus:from-[#3D3024] focus:to-[#2a2a2a] focus:text-white transition-all duration-200 hover:translate-x-1 font-arial"

                      onClick={() => setStatusFilter("PRONTO")}

                    >

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

                        <span>Status: PRONTO</span>

                        <div className="w-3 h-3 bg-blue-600 rounded-full"></div>

                      </div>

                    </DropdownMenuItem>

                    <DropdownMenuItem

                      className="focus:bg-gradient-to-r focus:from-[#3D3024] focus:to-[#2a2a2a] focus:text-white transition-all duration-200 hover:translate-x-1 font-arial"

                      onClick={() => setUnidadeFilter("USF RIO DAS VELHAS II")}

                    >

                      Unidade: USF RIO DAS VELHAS II

                    </DropdownMenuItem>

                    <DropdownMenuItem

                      className="focus:bg-gradient-to-r focus:from-[#3D3024] focus:to-[#2a2a2a] focus:text-white transition-all duration-200 hover:translate-x-1 font-arial"

                      onClick={() => setUnidadeFilter("USF QUADROS III")}

                    >

                      Unidade: USF QUADROS III

                    </DropdownMenuItem>

                    <DropdownMenuItem

                      className="focus:bg-gradient-to-r focus:from-[#3D3024] focus:to-[#2a2a2a] focus:text-white transition-all duration-200 hover:translate-x-1 font-arial"

                      onClick={() => setUnidadeFilter("USF VIETNÃ")}

                    >

                      Unidade: USF VIETNÃ

                    </DropdownMenuItem>

                    <div className="px-3 py-2 mt-1 border-t border-[#3D3024] text-center">

                      <Button

                        variant="link"

                        onClick={clearFilters}

                        className="text-[#E9A870] hover:text-[#E0CAA5] text-sm w-full transition-colors font-arial"

                      >

                        Limpar filtros

                      </Button>

                    </div>

                  </DropdownMenuContent>

                </DropdownMenu>

                {*/\* Expandable Search \*/*}

                <div

                  className={cn(

                    "flex items-center transition-all duration-300 overflow-hidden",

                    searchOpen ? "w-64" : "w-10",

                  )}

                >

                  <div className="flex items-center bg-[#3D3024] rounded-lg overflow-hidden shadow-lg border border-[#E9A870]/20">

                    <Button

                      variant="ghost"

                      size="icon"

                      className="h-10 w-10 bg-transparent hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 transition-all duration-200 hover:scale-110"

                      onClick={() => setSearchOpen(!searchOpen)}

                    >

                      {searchOpen ? (

                        <X className="h-5 w-5 text-[#E9A870]" />

                      ) : (

                        <Search className="h-5 w-5 text-[#E9A870]" />

                      )}

                    </Button>

                    <Input

                      ref={searchInputRef}

                      type="text"

                      placeholder="Pesquisar..."

                      value={searchQuery}

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

                      className={cn(

                        "h-10 bg-transparent border-0 focus-visible:ring-0 text-white placeholder:text-gray-400 font-arial",

                        "transition-all duration-300",

                        searchOpen ? "w-full opacity-100" : "w-0 opacity-0 p-0",

                      )}

                    />

                  </div>

                </div>

                <div className="flex gap-4">

                  {tabs.map((tab) => (

                    <button

                      key={tab}

                      onClick={() => setActiveTab(tab)}

                      className={`text-base font-medium px-4 py-2 rounded-lg transition-all duration-300 hover:scale-105 font-arial ${

                        activeTab === tab

                          ? "text-[#E9A870] bg-gradient-to-r from-[#1C1815] to-[#1C1815] shadow-md border border-[#E9A870]/20 scale-105"

                          : "text-[#E0CAA5] hover:bg-gradient-to-r hover:from-gray-800/30 hover:to-gray-700/30 hover:shadow-sm"

                      }`}

                    >

                      {tab}

                    </button>

                  ))}

                </div>

              </div>

            </div>

          </div>

          {*/\* Table - Full Width with Edge-to-Edge Borders \*/*}

          <div className="flex-1 px-6 pb-6">

            <div className="bg-[#1C1815] border-[#E9A870] border-2 rounded-xl overflow-hidden shadow-2xl backdrop-blur-sm hover:shadow-[#E9A870]/10 transition-all duration-300 h-full flex flex-col">

              <div className="bg-gradient-to-r from-[#E9A870] to-[#A8784F] p-0 flex-shrink-0">

                <div className="grid grid-cols-5 gap-6 px-6 py-4 text-black font-bold text-lg font-arial">

                  <div>TOMBAMENTO</div>

                  <div>STATUS</div>

                  <div>UNIDADE</div>

                  <div className="text-center">ID</div>

                  <div>AÇÕES</div>

                </div>

              </div>

              <div className="flex-1 bg-[#1C1815] custom-scroll overflow-y-auto">

                {isLoading ? (

                  <TableSkeleton />

                ) : filteredComputers.length > 0 ? (

                  filteredComputers.map((computer, index) => {

*const* { ledColor, icon, pulseClass, baseClasses } = getStatusIndicator(

                      computer.status,

                      computer.priority,

                    )

*return* (

                      <div

                        key={computer.id}

                        className="grid grid-cols-5 gap-6 px-6 py-4 border-b border-[#E9A870]/30 items-center hover:bg-gradient-to-r hover:from-gray-800/20 hover:to-gray-700/20 transition-all duration-300 group hover:scale-[1.01] hover:shadow-lg cursor-pointer"

                        style={{ animationDelay: `${index \* 100}ms` }}

                        onClick={() => handleRowClick(computer)}

                      >

                        <div className="text-gray-300 group-hover:text-white transition-colors font-bold text-base font-abel">

                          {computer.tombamento}

                        </div>

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

                          <div className="flex items-center gap-2">

                            {icon}

                            <span className="text-gray-300 text-sm font-arial">{computer.status}</span>

                          </div>

                          <div className={`${baseClasses} ${ledColor} ${pulseClass}`}></div>

                        </div>

                        <div className="text-gray-300 group-hover:text-white transition-colors font-bold text-base font-abel">

                          <button

                            onClick={(e) => handleTecnicoClick(computer.tecnico, e)}

                            className="hover:text-[#E9A870] transition-colors"

                          >

                            {computer.unidade}

                          </button>

                        </div>

                        <div className="text-gray-300 group-hover:text-white transition-colors text-center text-base font-abel">

                          {computer.id}

                        </div>

                        <div className="flex gap-3">

                          <Button

                            variant="ghost"

                            size="icon"

                            onClick={(e) => {

                              e.stopPropagation()

                              handleRowClick(computer)

                            }}

                            className="h-8 w-8 hover:bg-gradient-to-r hover:from-gray-700/50 hover:to-gray-600/50 transition-all duration-200 hover:scale-110"

                          >

                            <Eye className="h-4 w-4 text-[#E0CAA5]" />

                          </Button>

                          <Button

                            variant="ghost"

                            size="icon"

                            onClick={(e) => e.stopPropagation()}

                            className="h-8 w-8 hover:bg-gradient-to-r hover:from-gray-700/50 hover:to-gray-600/50 transition-all duration-200 hover:scale-110"

                          >

                            <Edit className="h-4 w-4 text-[#E0CAA5]" />

                          </Button>

                          <AlertDialog open={deleteDialogOpen} onOpenChange={setDeleteDialogOpen}>

                            <AlertDialogTrigger asChild>

                              <Button

                                variant="ghost"

                                size="icon"

                                onClick={(e) => handleDeleteClick(computer, e)}

                                className="h-8 w-8 hover:bg-gradient-to-r hover:from-red-700/50 hover:to-red-600/50 transition-all duration-200 hover:scale-110"

                              >

                                <Trash2 className="h-4 w-4 text-red-400" />

                              </Button>

                            </AlertDialogTrigger>

                            <AlertDialogContent className="bg-[#1C1815] border-[#E9A870] text-white">

                              <AlertDialogHeader>

                                <AlertDialogTitle className="text-[#E9A870] text-xl font-abel">

                                  Confirmar Exclusão

                                </AlertDialogTitle>

                                <AlertDialogDescription className="text-gray-300 font-arial">

                                  Tem certeza que deseja excluir o registro do computador{" "}

                                  <span className="font-bold text-white">{computerToDelete?.tombamento}</span>?

                                  <br />

                                  <span className="text-red-400 text-sm">Esta ação não pode ser desfeita.</span>

                                </AlertDialogDescription>

                              </AlertDialogHeader>

                              <AlertDialogFooter>

                                <AlertDialogCancel className="bg-gray-600 hover:bg-gray-700 text-white border-gray-500">

                                  Cancelar

                                </AlertDialogCancel>

                                <AlertDialogAction

                                  onClick={confirmDelete}

                                  className="bg-gradient-to-r from-red-600 to-red-700 hover:from-red-700 hover:to-red-800 text-white"

                                >

                                  <Trash2 className="h-4 w-4 mr-2" />

                                  Excluir

                                </AlertDialogAction>

                              </AlertDialogFooter>

                            </AlertDialogContent>

                          </AlertDialog>

                        </div>

                      </div>

                    )

                  })

                ) : (

                  <div className="flex justify-center items-center h-40 text-gray-400 font-arial">

                    Nenhum resultado encontrado

                  </div>

                )}

              </div>

            </div>

          </div>

        </div>

        {*/\* Right Sidebar - Full Height \*/*}

        <div className="w-72 border-l border-[#E9A870] bg-[#191818] flex-shrink-0 flex flex-col">

          <div className="p-6 flex-1 custom-scroll">

            <h3 className="text-[32px] font-afacad bg-gradient-to-r from-[#E9A870] via-[#E0CAA5] to-[#A8784F] bg-clip-text text-transparent font-semibold mb-6 leading-tight text-center drop-shadow-sm">

              PRONTOS

            </h3>

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

              {readyItems.map((item, index) => (

                <div

                  key={index}

                  className="flex items-center justify-between py-3 px-3 rounded-lg hover:bg-gradient-to-r hover:from-gray-800/50 hover:to-gray-700/50 transition-all duration-300 hover:shadow-md group border border-transparent hover:border-[#E9A870]/20 hover:scale-105 hover:-translate-y-1 cursor-pointer"

                  style={{ animationDelay: `${index \* 150}ms` }}

                  onClick={() => onNavigate("detalhes", { id: item.id, tombamento: item.tombamento })}

                >

                  <div className="flex items-center gap-4 flex-1">

                    <div className="flex flex-col items-center gap-1">

                      <CheckCircle className="h-3 w-3 text-blue-400 group-hover:scale-110 transition-transform duration-200" />

                      <div className="w-3 h-3 bg-blue-600 rounded-full shadow-sm group-hover:shadow-lg transition-shadow duration-200"></div>

                    </div>

                    <div className="flex-1">

                      <span className="text-sm text-gray-300 leading-relaxed group-hover:text-white transition-colors font-arial">

                        {item.text}

                      </span>

                      <div className="flex items-center gap-2 mt-1">

                        <span className="text-sm text-gray-500 font-arial">TOMBAMENTO: {item.tombamento}</span>

                        <span className="text-sm text-blue-400 font-arial">• {item.timestamp}</span>

                      </div>

                    </div>

                  </div>

                  <Button

                    variant="ghost"

                    size="icon"

                    className="h-8 w-8 hover:bg-gradient-to-r hover:from-gray-700/50 hover:to-gray-600/50 transition-all duration-200 hover:scale-110"

                  >

                    <Eye className="h-4 w-4 text-[#E0CAA5]" />

                  </Button>

                </div>

              ))}

            </div>

          </div>

        </div>

      </div>

    </div>

  )

}

frontend/app/global.css:

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer base {

  :root {

    --background: 0 0% 100%;

    --foreground: 0 0% 3.9%;

    --card: 0 0% 100%;

    --card-foreground: 0 0% 3.9%;

    --popover: 0 0% 100%;

    --popover-foreground: 0 0% 3.9%;

    --primary: 0 0% 9%;

    --primary-foreground: 0 0% 98%;

    --secondary: 0 0% 96.1%;

    --secondary-foreground: 0 0% 9%;

    --muted: 0 0% 96.1%;

    --muted-foreground: 0 0% 45.1%;

    --accent: 0 0% 96.1%;

    --accent-foreground: 0 0% 9%;

    --destructive: 0 84.2% 60.2%;

    --destructive-foreground: 0 0% 98%;

    --border: 0 0% 89.8%;

    --input: 0 0% 89.8%;

    --ring: 0 0% 3.9%;

    --chart-1: 12 76% 61%;

    --chart-2: 173 58% 39%;

    --chart-3: 197 37% 24%;

    --chart-4: 43 74% 66%;

    --chart-5: 27 87% 67%;

    --radius: 0.5rem;

  }

  .dark {

    --background: 0 0% 3.9%;

    --foreground: 0 0% 98%;

    --card: 0 0% 3.9%;

    --card-foreground: 0 0% 98%;

    --popover: 0 0% 3.9%;

    --popover-foreground: 0 0% 98%;

    --primary: 0 0% 98%;

    --primary-foreground: 0 0% 9%;

    --secondary: 0 0% 14.9%;

    --secondary-foreground: 0 0% 98%;

    --muted: 0 0% 14.9%;

    --muted-foreground: 0 0% 63.9%;

    --accent: 0 0% 14.9%;

    --accent-foreground: 0 0% 98%;

    --destructive: 0 62.8% 30.6%;

    --destructive-foreground: 0 0% 98%;

    --border: 0 0% 14.9%;

    --input: 0 0% 14.9%;

    --ring: 0 0% 83.1%;

    --chart-1: 220 70% 50%;

    --chart-2: 160 60% 45%;

    --chart-3: 30 80% 55%;

    --chart-4: 280 65% 60%;

    --chart-5: 340 75% 55%;

  }

}

*/\* Theme-specific custom properties \*/*

[data-theme="dark"] {

  --agnus-bg-primary: #191818;

  --agnus-bg-secondary: #1c1815;

  --agnus-bg-tertiary: #0c0c0c;

  --agnus-bg-card: #1c1815;

  --agnus-bg-input: #121212;

  --agnus-bg-hover: #2a231e;

  --agnus-text-primary: #ffffff;

  --agnus-text-secondary: #e0caa5;

  --agnus-text-muted: #9ca3af;

  --agnus-text-accent: #e9a870;

  --agnus-border-primary: #e9a870;

  --agnus-border-secondary: rgba(233, 168, 112, 0.3);

  --agnus-border-muted: rgba(233, 168, 112, 0.1);

  --agnus-accent-primary: #e9a870;

  --agnus-accent-secondary: #a8784f;

  --agnus-accent-gradient: linear-gradient(to right, #e9a870, #a8784f);

}

[data-theme="light"] {

  --agnus-bg-primary: #fafafa;

  --agnus-bg-secondary: #ffffff;

  --agnus-bg-tertiary: #f5f5f5;

  --agnus-bg-card: #ffffff;

  --agnus-bg-input: #f9f9f9;

  --agnus-bg-hover: #f0f0f0;

  --agnus-text-primary: #1f2937;

  --agnus-text-secondary: #374151;

  --agnus-text-muted: #6b7280;

  --agnus-text-accent: #d97706;

  --agnus-border-primary: #d97706;

  --agnus-border-secondary: rgba(217, 119, 6, 0.3);

  --agnus-border-muted: rgba(217, 119, 6, 0.1);

  --agnus-accent-primary: #d97706;

  --agnus-accent-secondary: #92400e;

  --agnus-accent-gradient: linear-gradient(to right, #d97706, #92400e);

}

@layer base {

  \* {

    @apply border-border;

  }

  body {

    @apply bg-background text-foreground;

    background-color: var(--agnus-bg-primary);

    color: var(--agnus-text-primary);

    transition: background-color 0.3s ease, color 0.3s ease;

  }

}

.font-abel {

  font-family: var(--font-abel);

}

.font-afacad {

  font-family: var(--font-afacad);

}

*/\* Standardized component styles \*/*

.agnus-card {

  background-color: var(--agnus-bg-card);

  border: 2px solid var(--agnus-border-secondary);

  border-radius: 12px;

  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);

  transition: all 0.3s ease;

}

.agnus-card:hover {

  border-color: var(--agnus-border-primary);

  box-shadow: 0 15px 35px rgba(233, 168, 112, 0.1);

}

.agnus-header {

  background: var(--agnus-accent-gradient);

  color: var(--agnus-bg-primary);

  padding: 16px 24px;

  font-weight: bold;

  font-size: 18px;

}

.agnus-button-primary {

  background: var(--agnus-accent-gradient);

  color: var(--agnus-bg-primary);

  border: none;

  padding: 12px 24px;

  border-radius: 8px;

  font-weight: bold;

  transition: all 0.2s ease;

  box-shadow: 0 4px 12px rgba(233, 168, 112, 0.3);

}

.agnus-button-primary:hover {

  transform: translateY(-2px);

  box-shadow: 0 6px 20px rgba(233, 168, 112, 0.4);

}

.agnus-button-secondary {

  background-color: var(--agnus-bg-hover);

  color: var(--agnus-text-accent);

  border: 1px solid var(--agnus-border-secondary);

  padding: 8px 16px;

  border-radius: 8px;

  transition: all 0.2s ease;

}

.agnus-button-secondary:hover {

  background-color: var(--agnus-bg-card);

  border-color: var(--agnus-border-primary);

  transform: scale(1.05);

}

.agnus-input {

  background-color: var(--agnus-bg-input);

  border: 1px solid var(--agnus-border-secondary);

  color: var(--agnus-text-primary);

  padding: 12px 16px;

  border-radius: 8px;

  font-size: 16px;

  transition: all 0.2s ease;

}

.agnus-input:focus {

  border-color: var(--agnus-border-primary);

  box-shadow: 0 0 0 3px rgba(233, 168, 112, 0.1);

  outline: none;

}

.agnus-text-title {

  background: var(--agnus-accent-gradient);

  background-clip: text;

  -webkit-background-clip: text;

  -webkit-text-fill-color: transparent;

  font-weight: 600;

}

*/\* TV-friendly styles \*/*

.tv-display {

  font-size: 18px;

  line-height: 1.6;

  letter-spacing: 0.5px;

}

.tv-display .text-lg {

  font-size: 22px !important;

}

.tv-display .text-xl {

  font-size: 28px !important;

}

.tv-display .text-2xl {

  font-size: 32px !important;

}

.tv-display .text-3xl {

  font-size: 40px !important;

}

.tv-display .text-4xl {

  font-size: 48px !important;

}

*/\* Auto-refresh indicator - Removed the floating badge \*/*

.auto-refresh-active {

  color: #10b981; */\* Green color for active auto-refresh \*/*

}

@keyframes pulse {

  0%,

  100% {

    opacity: 1;

  }

  50% {

    opacity: 0.7;

  }

}

*/\* Custom Scrollbar Styles \*/*

\* {

  scrollbar-width: thin;

  scrollbar-color: var(--agnus-border-secondary) transparent;

}

\*::-webkit-scrollbar {

  width: 8px;

  height: 8px;

}

\*::-webkit-scrollbar-track {

  background: transparent;

  border-radius: 10px;

}

\*::-webkit-scrollbar-thumb {

  background: var(--agnus-border-secondary);

  border-radius: 10px;

  transition: all 0.3s ease;

}

\*::-webkit-scrollbar-thumb:hover {

  background: var(--agnus-border-primary);

}

\*::-webkit-scrollbar-corner {

  background: transparent;

}

*/\* Smooth scrolling \*/*

html {

  scroll-behavior: smooth;

}

*/\* Custom scrollbar for specific components \*/*

.custom-scroll {

  scrollbar-width: thin;

  scrollbar-color: var(--agnus-border-secondary) transparent;

}

.custom-scroll::-webkit-scrollbar {

  width: 6px;

  height: 6px;

}

.custom-scroll::-webkit-scrollbar-track {

  background: transparent;

  border-radius: 10px;

}

.custom-scroll::-webkit-scrollbar-thumb {

  background: var(--agnus-border-secondary);

  border-radius: 10px;

  transition: all 0.3s ease;

}

.custom-scroll::-webkit-scrollbar-thumb:hover {

  background: var(--agnus-border-primary);

}

*/\* Ensure minimum font size of 14px \*/*

.text-xs {

  font-size: 14px !important;

}

*/\* Status indicators with better contrast \*/*

.status-entrada {

  background-color: rgba(239, 68, 68, 0.2);

  color: #fca5a5;

  border: 1px solid rgba(239, 68, 68, 0.4);

}

.status-saida {

  background-color: rgba(34, 197, 94, 0.2);

  color: #86efac;

  border: 1px solid rgba(34, 197, 94, 0.4);

}

.status-pronto {

  background-color: rgba(59, 130, 246, 0.2);

  color: #93c5fd;

  border: 1px solid rgba(59, 130, 246, 0.4);

}

[data-theme="light"] .status-entrada {

  background-color: rgba(239, 68, 68, 0.1);

  color: #dc2626;

  border: 1px solid rgba(239, 68, 68, 0.3);

}

[data-theme="light"] .status-saida {

  background-color: rgba(34, 197, 94, 0.1);

  color: #16a34a;

  border: 1px solid rgba(34, 197, 94, 0.3);

}

[data-theme="light"] .status-pronto {

  background-color: rgba(59, 130, 246, 0.1);

  color: #2563eb;

  border: 1px solid rgba(59, 130, 246, 0.3);

}

*/\* Enhanced focus styles for accessibility \*/*

.agnus-focus:focus {

  outline: 2px solid var(--agnus-accent-primary);

  outline-offset: 2px;

}

*/\* Improved animations \*/*

.agnus-animate-in {

  animation: slideInUp 0.3s ease-out;

}

@keyframes slideInUp {

  from {

    opacity: 0;

    transform: translateY(20px);

  }

  to {

    opacity: 1;

    transform: translateY(0);

  }

}

.agnus-hover-lift:hover {

  transform: translateY(-4px);

  transition: transform 0.2s ease;

}

*/\* Loading states \*/*

.agnus-loading {

  background: linear-gradient(90deg, var(--agnus-bg-hover) 25%, var(--agnus-bg-card) 50%, var(--agnus-bg-hover) 75%);

  background-size: 200% 100%;

  animation: loading 1.5s infinite;

}

@keyframes loading {

  0% {

    background-position: 200% 0;

  }

  100% {

    background-position: -200% 0;

  }

}

frontend/components/sidebar-menu.tsx:

"use client"

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

import { HomeIcon as House, PenTool, Wrench, LayoutDashboard, ChevronLeft } from "lucide-react"

import { useTheme } from "@/app/contexts/ThemeContext"

import React, { useMemo } from "react"

import Image from "next/image"

interface SidebarMenuProps {

  isOpen: boolean

  onClose: () => void

  onNavigate: (page: string, data?: any) => void

  currentPage?: string

}

*// Deer/Stag SVG Icon*

*const* DeerIcon = React.memo(() => (

  <div className="w-34 h-34 rounded-lg p-2 flex items-center justify-center shadow-lg">

    <Image src="/lamb.svg" alt="Deer Icon" width={70} height={70} />

  </div>

))

*// QR Code SVG with proper gradient*

*const* QRCode = React.memo(() => (

  <div className="w-34 h-34 rounded-lg p-2 flex items-center justify-center shadow-lg">

    <Image src="/qr-code.svg" alt="QR Code" width={120} height={120} />

  </div>

))

export function SidebarMenu({ isOpen, onClose, onNavigate, currentPage = "home" }: SidebarMenuProps) {

*const* { theme, toggleTheme } = useTheme()

*// Memoize menuItems*

*const* menuItems = useMemo(

    () => [

      {

        id: "HOME",

        label: "HOME",

        icon: House,

        page: "home",

      },

      {

        id: "REGISTRO",

        label: "REGISTRO",

        icon: PenTool,

        page: "registro",

      },

      {

        id: "TECNICOS",

        label: "TÉCNICOS",

        icon: Wrench,

        page: "tecnicos",

      },

      {

        id: "DASHBOARD",

        label: "DASHBOARD",

        icon: LayoutDashboard,

        page: "dashboard",

      },

    ],

    []

  )

*// Derive activeItem from currentPage*

*const* pageToMenuMap: { [key: string]: string } = {

    home: "HOME",

    registro: "REGISTRO",

    tecnicos: "TECNICOS",

    dashboard: "DASHBOARD",

    detalhes: "HOME",

  }

*const* activeItem = pageToMenuMap[currentPage] || "HOME"

*const* handleMenuClick = (item: any) => {

    onNavigate(item.page)

    onClose()

  }

*return* (

    <>

      {*/\* Overlay \*/*}

      {isOpen && <div className="fixed inset-0 bg-black/60 backdrop-blur-sm z-40" onClick={onClose} />}

      {*/\* Sidebar \*/*}

      <div

        className={`

          fixed top-0 left-0 h-full w-72

          bg-gradient-to-b from-[#0C0C0C] via-[#1A1A1A] to-[#0C0C0C]

          z-50 transform transition-transform duration-300 ease-in-out

          ${isOpen ? "translate-x-0" : "-translate-x-full"}

          flex flex-col font-abel shadow-2xl border-r border-[#E9A870]/20

        `}

      >

        {*/\* Header \*/*}

        <div className="flex items-center justify-between p-6 border-b border-[#E9A870]/10">

          <DeerIcon />

          <Button

            variant="ghost"

            size="icon"

            onClick={onClose}

            className="text-[#E0CAA5] hover:bg-gradient-to-r hover:from-[#E9A870]/20 hover:to-[#A8784F]/20 w-8 h-8 rounded-lg transition-all duration-200"

          >

            <ChevronLeft className="h-5 w-5" />

          </Button>

        </div>

        {*/\* Navigation Menu - Centered \*/*}

        <div className="flex-1 flex items-center justify-center px-6">

          <div className="space-y-4 w-full max-w-[200px]">

            {menuItems.map((item) => (

              <button

                key={item.id}

                onClick={() => handleMenuClick(item)}

                className={`

                  w-full flex items-center justify-center gap-3 px-4 py-3 rounded-xl font-medium text-sm

                  transition-all duration-300 transform hover:scale-105

                  ${

                    activeItem === item.id

                      ? "bg-gradient-to-r from-[#E9A870] to-[#A8784F] text-black shadow-lg shadow-[#E9A870]/25"

                      : "bg-[#3D3024] text-[#E0CAA5] hover:bg-[#4D4034] hover:shadow-md"

                  }

                `}

              >

                <item.icon className={`h-5 w-5 ${item.id === "TECNICOS" ? "stroke-[2.5]" : ""}`} />

                <span className="tracking-wide">{item.label}</span>

              </button>

            ))}

            <div className="mt-6 pt-4 border-t border-[#E9A870]/20">

              <button

                onClick={toggleTheme}

                className={`

                  w-full flex items-center justify-center gap-3 px-4 py-3 rounded-xl font-medium text-sm

                  transition-all duration-300 transform hover:scale-105

                  bg-[#3D3024] text-[#E0CAA5] hover:bg-[#4D4034] hover:shadow-md

                `}

              >

                {theme === "dark" ? (

                  <>

                    <svg className="h-5 w-5" fill="none" stroke="currentColor" viewBox="0 0 24 24">

                      <path

                        strokeLinecap="round"

                        strokeLinejoin="round"

                        strokeWidth={2}

                        d="M12 3v1m0 16v1m9-9h-1M4 12H3m15.364 6.364l-.707-.707M6.343 6.343l-.707-.707m12.728 0l-.707.707M6.343 17.657l-.707.707M16 12a4 4 0 11-8 0 4 4 0 018 0z"

                      />

                    </svg>

                    <span className="tracking-wide">Light Mode</span>

                  </>

                ) : (

                  <>

                    <svg className="h-5 w-5" fill="none" stroke="currentColor" viewBox="0 0 24 24">

                      <path

                        strokeLinecap="round"

                        strokeLinejoin="round"

                        strokeWidth={2}

                        d="M20.354 15.354A9 9 0 018.646 3.646 9.003 9.003 0 0012 21a9.003 9.003 0 008.354-5.646z"

                      />

                    </svg>

                    <span className="tracking-wide">Dark Mode</span>

                  </>

                )}

              </button>

            </div>

          </div>

        </div>

        {*/\* Footer with QR Code and gradient background \*/*}

        <div className="p-6 flex flex-col items-center bg-gradient-to-t from-[#2A231E] via-[#1A1A1A] to-transparent rounded-t-2xl">

          <QRCode />

          <div className="mt-4 text-center bg-gradient-to-b from-[#E9A870] to-[#A8784F] bg-clip-text text-transparent">

            <p className="text-xs leading-relaxed font-medium">

              AGNUS - GETI DA SEC. DE SAÚDE -<br />

              JABOATÃO DOS GUARARAPES - JUNHO DE

              <br />

              2025 - VER: 4.2

            </p>

          </div>

        </div>

      </div>

    </>

  )

}