

Sistema de Control de Terminaciones - Next.js 14

Metodología de Desarrollo por Sprints

Sprint Overview:

1. **Sprint 1:** Setup y Dashboard básico (2 semanas)
 2. **Sprint 2:** Autenticación y roles (1.5 semanas)
 3. **Sprint 3:** Mantenedores base (2 semanas)
 4. **Sprint 4:** Vista de pisos y departamentos (2 semanas)
 5. **Sprint 5:** Control de actividades y fotos (2.5 semanas)
 6. **Sprint 6:** Reportes y deployment (1.5 semanas)
-

SPRINT 1: Setup y Dashboard Básico

Objetivos del Sprint:

- ☒ Configurar proyecto Next.js 14 con versiones estables
- ☒ Integrar Supabase sin Prisma
- ☒ Crear dashboard con datos de ejemplo
- ☒ Estructura base responsive para tablets/móviles

Versiones Estables a Utilizar:

```
{  
  "next": "14.2.5",  
  "react": "18.3.1",  
  "react-dom": "18.3.1",  
  "@supabase/supabase-js": "2.45.1",  
  "tailwindcss": "3.4.7",  
  "lucide-react": "0.424.0",  
  "react-hook-form": "7.52.1",
```

```
"date-fns": "3.6.0",  
"react-hot-toast": "2.4.1",  
"recharts": "2.12.7",  
"jspdf": "2.5.1"  
}
```

Paso 1.1: Crear Proyecto Base

Crear proyecto Next.js 14

```
npx create-next-app@14.2.5 sistema-control-terminaciones --typescript --tailwind --  
eslint --app --src-dir
```

```
cd sistema-control-terminaciones
```

Instalar dependencias estables

```
npm install @supabase/supabase-js@2.45.1 lucide-react@0.424.0 react-hook-  
form@7.52.1 date-fns@3.6.0 react-hot-toast@2.4.1 recharts@2.12.7 jspdf@2.5.1
```

Dependencias de desarrollo

```
npm install -D @types/node@20.14.12
```

Paso 1.2: Estructura de Carpetas

```
src/
```

```
├── app/  
│   ├── (auth)/      # Rutas protegidas  
│   │   ├── dashboard/  
│   │   ├── pisos/  
│   │   ├── equipos/  
│   │   └── reportes/
```

```
|   ├── api/          # API Routes
|   ├── login/        # Página de login
|   ├── globals.css
|   ├── layout.tsx
|   └── page.tsx
└── components/
    ├── ui/           # Componentes base
    ├── dashboard/    # Componentes dashboard
    ├── layout/       # Layout components
    └── common/       # Componentes comunes
└── lib/
    ├── supabase.ts   # Cliente Supabase
    ├── utils.ts      # Utilidades
    └── constants.ts  # Constantes del sistema
└── hooks/           # Custom hooks
└── types/           # TypeScript types
└── data/            # Datos de ejemplo
```

Paso 1.3: Configurar Supabase

Archivo: .env.local

NEXT_PUBLIC_SUPABASE_URL=your_supabase_url

NEXT_PUBLIC_SUPABASE_ANON_KEY=your_supabase_anon_key

SUPABASE_SERVICE_ROLE_KEY=your_service_role_key

Archivo: src/lib/supabase.ts

```
import { createClient } from '@supabase/supabase-js'
```

```
const supabaseUrl = process.env.NEXT_PUBLIC_SUPABASE_URL!

const supabaseAnonKey = process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!


export const supabase = createClient(supabaseUrl, supabaseAnonKey)


// Cliente para Server Components

export const createServerSupabaseClient = () => {
  return createClient(supabaseUrl, supabaseAnonKey)
}


export type Database = {
  public: {
    Tables: {
      projects: {
        Row: {
          id: number
          name: string
          address: string | null
          total_floors: number | null
          units_per_floor: number | null
          start_date: string | null
          estimated_completion: string | null
          status: string
          created_at: string
        }
      }
    }
    Insert: {
```

```
    id?: number
    name: string
    address?: string | null
    total_floors?: number | null
    units_per_floor?: number | null
    start_date?: string | null
    estimated_completion?: string | null
    status?: string
    created_at?: string
  }
  Update: {
    id?: number
    name?: string
    address?: string | null
    total_floors?: number | null
    units_per_floor?: number | null
    start_date?: string | null
    estimated_completion?: string | null
    status?: string
    created_at?: string
  }
}

// Más tablas se agregarán en siguientes sprints
}
}
}
```

Paso 1.4: Configurar Tailwind CSS

Archivo: **tailwind.config.ts**

```
import type { Config } from 'tailwindcss'

const config: Config = {
  content: [
    './src/pages/**/*.tsx',
    './src/components/**/*.tsx',
    './src/app/**/*.tsx',
  ],
  theme: {
    extend: {
      colors: {
        primary: {
          50: '#eff6ff',
          100: '#dbeafe',
          200: '#bfdbfe',
          300: '#93c5fd',
          400: '#60a5fa',
          500: '#3b82f6',
          600: '#2563eb',
          700: '#1d4ed8',
          800: '#1e40af',
          900: '#1e3a8a',
        },
      },
    },
  },
  success: {
```

```
    50: '#f0fdf4',
    100: '#dcfce7',
    500: '#22c55e',
    600: '#16a34a',
    700: '#15803d',
  },
  warning: {
    50: '#ffffbe',
    100: '#fef3c7',
    500: '#f59e0b',
    600: '#d97706',
    700: '#b45309',
  },
  danger: {
    50: '#fef2f2',
    100: '#fee2e2',
    500: '#ef4444',
    600: '#dc2626',
    700: '#b91c1c',
  }
},
  screens: {
    'xs': '475px',
  }
},
},
```

```
plugins: [],  
}
```

export default config

Paso 1.5: Crear Componentes Base

Archivo: src/lib/utils.ts

```
import { type ClassValue, clsx } from "clsx"  
import { twMerge } from "tailwind-merge"
```

```
export function cn(...inputs: ClassValue[]) {  
  return twMerge(clsx(inputs))  
}
```

```
export function getStatusColor(status: string): string {  
  switch (status) {  
    case 'completed':  
      return 'bg-success-500 text-white'  
    case 'good':  
      return 'bg-success-400 text-white'  
    case 'in-progress':  
      return 'bg-primary-500 text-white'  
    case 'warning':  
      return 'bg-warning-500 text-white'  
    case 'danger':  
      return 'bg-danger-500 text-white'  
    case 'blocked':
```



```
    return 'bg-danger-600 text-white'
  case 'pending':
    return 'bg-gray-400 text-white'
  default:
    return 'bg-gray-300 text-gray-700'
}
}
```

```
export function getStatusEmoji(status: string): string {
  switch (status) {
    case 'completed': return '✅'
    case 'good': return '🟢'
    case 'in-progress': return '🟡'
    case 'warning': return '🟡'
    case 'danger': return '🔴'
    case 'blocked': return '🔴'
    case 'pending': return '⬜'
    default: return '⬜'
  }
}
```

```
export function formatDate(date: string | Date): string {
  return new Date(date).toLocaleDateString('es-CL', {
    day: '2-digit',
    month: 'short',
```

```
    year: 'numeric'
  })
}
```

```
export function calculateProgress(activities: any[]): number {
  if (!activities || activities.length === 0) return 0

  const totalProgress = activities.reduce((sum, activity) => sum + (activity.progress || 0), 0)

  return Math.round(totalProgress / activities.length)
}
```

Archivo: src/components/ui/Button.tsx

```
import { cn } from '@lib/utils'

import { ButtonHTMLAttributes, forwardRef } from 'react'

export interface ButtonProps extends ButtonHTMLAttributes<HTMLButtonElement> {
  variant?: 'primary' | 'secondary' | 'success' | 'warning' | 'danger' | 'outline'
  size?: 'sm' | 'md' | 'lg' | 'xl'
}

const Button = forwardRef<HTMLButtonElement, ButtonProps>(
  ({ className, variant = 'primary', size = 'md', ...props }, ref) => {
    const baseStyles = 'inline-flex items-center justify-center rounded-lg font-medium transition-colors focus:outline-none focus:ring-2 focus:ring-offset-2 disabled:opacity-50 disabled:pointer-events-none'

    const variants = {
      primary: 'bg-primary-600 text-white hover:bg-primary-700 focus:ring-primary-500',
```

```
    secondary: 'bg-gray-200 text-gray-900 hover:bg-gray-300 focus:ring-gray-500',
    success: 'bg-success-600 text-white hover:bg-success-700 focus:ring-success-500',
    warning: 'bg-warning-600 text-white hover:bg-warning-700 focus:ring-warning-500',
    danger: 'bg-danger-600 text-white hover:bg-danger-700 focus:ring-danger-500',
    outline: 'border border-gray-300 bg-white text-gray-700 hover:bg-gray-50 focus:ring-primary-500',
  }
}
```

```
const sizes = {
  sm: 'px-3 py-1.5 text-sm',
  md: 'px-4 py-2 text-sm',
  lg: 'px-6 py-3 text-base',
  xl: 'px-8 py-4 text-lg',
}
```

```
return (
  <button
    className={cn(
      baseStyles,
      variants[variant],
      sizes[size],
      className
    )}
    ref={ref}
    {...props}
  />
```

```
)  
}  
)
```

```
Button.displayName = 'Button'
```

```
export { Button }
```

Archivo: src/components/ui/Card.tsx

```
import { cn } from '@lib/Utils'
```

```
import { HTMLAttributes, forwardRef } from 'react'
```

```
const Card = forwardRef<HTMLDivElement, HTMLAttributes<HTMLDivElement>>(  
  ({ className, ...props }, ref) => (  
    <div  
      ref={ref}  
      className={cn(  
        'rounded-lg border bg-white p-6 shadow-sm',  
        className  
      )}  
      {...props}  
    />  
  )  
)
```

```
Card.displayName = 'Card'
```

```
const CardHeader = forwardRef<HTMLDivElement,  
HTMLAttributes<HTMLDivElement>>(  
  
  ({ className, ...props }, ref) => (  
  
    <div  
  
      ref={ref}  
  
      className={cn('flex flex-col space-y-1.5 pb-4', className)}  
  
      {...props}  
  
    />  
  
  )  
  
)  
  
CardHeader.displayName = 'CardHeader'
```

```
const CardTitle = forwardRef<HTMLHeadingElement,  
HTMLAttributes<HTMLHeadingElement>>(  
  
  ({ className, ...props }, ref) => (  
  
    <h3  
  
      ref={ref}  
  
      className={cn('text-lg font-semibold leading-none tracking-tight', className)}  
  
      {...props}  
  
    />  
  
  )  
  
)  
  
CardTitle.displayName = 'CardTitle'
```

```
const CardContent = forwardRef<HTMLDivElement,  
HTMLAttributes<HTMLDivElement>>(  
  
  ({ className, ...props }, ref) => (  
  
    <div  
  
      ref={ref}  
  
      className={cn('flex flex-col space-y-1.5', className)}  
  
      {...props}  
  
    />  
  
  )  
  
)  
  
CardContent.displayName = 'CardContent'
```

```
    <div ref={ref} className={cn('pt-0', className)} {...props} />
  )
)
CardContent.displayName = 'CardContent'
```

```
export { Card, CardHeader, CardTitle, CardContent }
```



Paso 1.6: Datos de Ejemplo

Archivo: src/data/mockData.ts

```
export interface Project {
  id: number
  name: string
  address: string
  totalFloors: number
  unitsPerFloor: number
  globalProgress: number
  status: 'active' | 'completed' | 'paused'
  estimatedCompletion: string
  currentFloor: number
}

export interface Floor {
  id: number
  projectId: number
  floorNumber: number
  progress: number
  status: 'completed' | 'good' | 'warning' | 'danger'
```

```
estimatedCompletion: string
delayDays: number
activeTeams: number
apartments: Apartment[]
}
```

```
export interface Apartment {
  id: string
  floorId: number
  apartmentNumber: string
  apartmentType: string
  area: number
  progress: number
  status: 'completed' | 'good' | 'warning' | 'danger' | 'pending'
  nextTask: string
  activities: Activity[]
}
```

```
export interface Activity {
  id: number
  name: string
  category: 'Estructura' | 'Instalaciones' | 'Acabados' | 'Pisos' | 'Carpintería' |
'Terminaciones'
  status: 'completed' | 'in-progress' | 'blocked' | 'pending'
  progress: number
  startDate: string
}
```

```
    endDate: string
    estimatedHours: number
    actualHours: number
    team: string
    supervisor: string
    notes: string
    photos: number
    dependencies: number[]
  }
```

// Actividades base del sistema (flujo de terminaciones)

```
export const ACTIVITIES_TEMPLATE: Omit<Activity, 'id' | 'startDate' | 'endDate' |
'actualHours' | 'notes' | 'photos'>[] = [
  {
    name: 'Tabiquería',
    category: 'Estructura',
    status: 'pending',
    progress: 0,
    estimatedHours: 24,
    team: 'Equipo Albañilería',
    supervisor: 'Carlos Mendoza',
    dependencies: []
  },
  {
    name: 'Instalación Eléctrica',
    category: 'Instalaciones',
```



```
status: 'pending',
progress: 0,
estimatedHours: 16,
team: 'Equipo Eléctrico',
supervisor: 'José Ramírez',
dependencies: [1] // Depende de Tabiquería
},
{
name: 'Instalación Sanitaria',
category: 'Instalaciones',
status: 'pending',
progress: 0,
estimatedHours: 20,
team: 'Equipo Gasfitería',
supervisor: 'Miguel Torres',
dependencies: [1] // Depende de Tabiquería
},
{
name: 'Estuco/Yeso',
category: 'Acabados',
status: 'pending',
progress: 0,
estimatedHours: 32,
team: 'Equipo Estuco',
supervisor: 'Antonio Silva',
dependencies: [2, 3] // Depende de instalaciones
```

```
},  
{  
  name: 'Pintura',  
  category: 'Acabados',  
  status: 'pending',  
  progress: 0,  
  estimatedHours: 24,  
  team: 'Equipo Pintura',  
  supervisor: 'Luis González',  
  dependencies: [4] // Depende de Estuco  
},  
{  
  name: 'Piso Flotante',  
  category: 'Pisos',  
  status: 'pending',  
  progress: 0,  
  estimatedHours: 16,  
  team: 'Equipo Pisos',  
  supervisor: 'Roberto Vega',  
  dependencies: [5] // Depende de Pintura  
},  
{  
  name: 'Puertas Interiores',  
  category: 'Carpintería',  
  status: 'pending',  
  progress: 0,
```

```
    estimatedHours: 8,  
    team: 'Equipo Carpintería',  
    supervisor: 'Fernando Rojas',  
    dependencies: [6] // Depende de Pisos  
  },  
  {  
    name: 'Accesorios y Grifería',  
    category: 'Terminaciones',  
    status: 'pending',  
    progress: 0,  
    estimatedHours: 6,  
    team: 'Equipo Terminaciones',  
    supervisor: 'Carmen López',  
    dependencies: [7] // Depende de Puertas  
  }  
]
```

// Datos de ejemplo para desarrollo

```
export const MOCK_PROJECTS: Project[] = [  
  {  
    id: 1,  
    name: 'EDIFICIO VISTA HERMOSA',  
    address: 'Av. Los Robles 2580, Las Condes',  
    totalFloors: 15,  
    unitsPerFloor: 4,  
    globalProgress: 78,
```

```
status: 'active',
estimatedCompletion: '2025-12-15',
currentFloor: 13
},
{
id: 2,
name: 'TORRES DEL SOL',
address: 'Calle Principal 1234, Providencia',
totalFloors: 20,
unitsPerFloor: 6,
globalProgress: 45,
status: 'active',
estimatedCompletion: '2026-03-20',
currentFloor: 8
},
{
id: 3,
name: 'RESIDENCIAL MAIPÚ',
address: 'Av. Pajaritos 3456, Maipú',
totalFloors: 8,
unitsPerFloor: 4,
globalProgress: 92,
status: 'active',
estimatedCompletion: '2025-10-30',
currentFloor: 8
}
```

]

```
export const MOCK_TEAMS = [  
  { id: 1, name: 'Equipo Albañilería', specialty: 'Tabiquería', supervisor: 'Carlos  
Mendoza', phone: '+56912345678', activeProjects: 2 },  
  
  { id: 2, name: 'Equipo Eléctrico', specialty: 'Instalaciones', supervisor: 'José Ramírez',  
phone: '+56912345679', activeProjects: 3 },  
  
  { id: 3, name: 'Equipo Gasfitería', specialty: 'Instalaciones', supervisor: 'Miguel Torres',  
phone: '+56912345680', activeProjects: 2 },  
  
  { id: 4, name: 'Equipo Estuco', specialty: 'Acabados', supervisor: 'Antonio Silva',  
phone: '+56912345681', activeProjects: 1 },  
  
  { id: 5, name: 'Equipo Pintura', specialty: 'Acabados', supervisor: 'Luis González',  
phone: '+56912345682', activeProjects: 2 },  
  
  { id: 6, name: 'Equipo Pisos', specialty: 'Pisos', supervisor: 'Roberto Vega', phone:  
'+56912345683', activeProjects: 1 },  
  
  { id: 7, name: 'Equipo Carpintería', specialty: 'Carpintería', supervisor: 'Fernando  
Rojas', phone: '+56912345684', activeProjects: 2 },  
  
  { id: 8, name: 'Equipo Terminaciones', specialty: 'Terminaciones', supervisor: 'Carmen  
López', phone: '+56912345685', activeProjects: 1 },  
]
```



Paso 1.7: Layout Principal

Archivo: src/app/layout.tsx

```
import './globals.css'  
  
import { Inter } from 'next/font/google'  
  
import { Toaster } from 'react-hot-toast'  
  
  
const inter = Inter({ subsets: ['latin'] })
```

```
export const metadata = {  
  title: 'Sistema Control Terminaciones',  
  description: 'Sistema de control y seguimiento de terminaciones en proyectos de construcción',  
}
```

```
export default function RootLayout({  
  children,  
}: {  
  children: React.ReactNode  
}) {  
  return (  
    <html lang="es">  
      <body className={inter.className}>  
        <div className="min-h-screen bg-gradient-to-br from-gray-50 to-blue-50">  
          {children}  
        </div>  
        <Toaster  
          position="top-right"  
          toastOptions={{  
            duration: 4000,  
            style: {  
              background: '#363636',  
              color: '#fff',  
            },  
          }}  
        </Toaster>  
      </body>  
    </html>  
  )  
}
```

```

    />

  </body>

</html>

)
}

```

Paso 1.8: Dashboard Principal

Archivo: src/app/page.tsx

```

import Link from 'next/link'

import { Building, Home, Users, ClipboardList, BarChart3, TrendingUp } from 'lucide-react'

import { Card, CardHeader, CardTitle, CardContent } from '@components/ui/Card'
import { Button } from '@components/ui/Button'

import { MOCK_PROJECTS, MOCK_TEAMS } from '@data/mockData'

export default function DashboardPage() {
  const totalProjects = MOCK_PROJECTS.length

  const activeProjects = MOCK_PROJECTS.filter(p => p.status === 'active').length

  const completedUnits = MOCK_PROJECTS.reduce((sum, project) => {
    return sum + Math.floor((project.globalProgress / 100) * (project.totalFloors * project.unitsPerFloor))
  }, 0)

  const totalUnits = MOCK_PROJECTS.reduce((sum, project) => {
    return sum + (project.totalFloors * project.unitsPerFloor)
  }, 0)

  const activeTeams = MOCK_TEAMS.length

  return (

```

```
<div className="min-h-screen p-4 md:p-6">

  <div className="max-w-7xl mx-auto space-y-6">

    {/* Header */}

    <Card className="bg-gradient-to-r from-blue-600 to-blue-700 text-white">

      <CardHeader>

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

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

            <Building size={40} />

            <div>

              <CardTitle className="text-white text-2xl md:text-3xl">

                Sistema Control Terminaciones

              </CardTitle>

              <p className="text-blue-100 mt-1">

                Gestión y seguimiento de proyectos de terminación

              </p>

            </div>

          </div>

          <div className="text-right">

            <div className="text-sm text-blue-100">Última actualización</div>

            <div className="font-semibold text-lg">

              {new Date().toLocaleDateString('es-CL')}

            </div>

          </div>

        </div>

      </CardHeader>

    </Card>

  </div>

</div>
```



```
{/* Métricas Principales */}
```

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
```

```
<Card className="bg-gradient-to-r from-green-500 to-green-600 text-white">
```

```
<CardContent className="p-6">
```

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

```
<div>
```

```
<p className="text-green-100 text-sm">Proyectos Activos</p>
```

```
<p className="text-3xl font-bold">{activeProjects}</p>
```

```
<p className="text-green-100 text-xs">de {totalProjects} totales</p>
```

```
</div>
```

```
<Building className="text-green-100" size={32} />
```

```
</div>
```

```
</CardContent>
```

```
</Card>
```

```
<Card className="bg-gradient-to-r from-blue-500 to-blue-600 text-white">
```

```
<CardContent className="p-6">
```

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

```
<div>
```

```
<p className="text-blue-100 text-sm">Unidades Terminadas</p>
```

```
<p className="text-3xl font-bold">{completedUnits}</p>
```

```
<p className="text-blue-100 text-xs">de {totalUnits} totales</p>
```

```
</div>
```

```
<Home className="text-blue-100" size={32} />
```

```
</div>
```

</CardContent>

</Card>

<Card className="bg-gradient-to-r from-purple-500 to-purple-600 text-white">

<CardContent className="p-6">

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

<div>

<p className="text-purple-100 text-sm">Equipos Activos</p>

<p className="text-3xl font-bold">{activeTeams}</p>

<p className="text-purple-100 text-xs">especialidades</p>

</div>

<Users className="text-purple-100" size={32} />

</div>

</CardContent>

</Card>

<Card className="bg-gradient-to-r from-orange-500 to-orange-600 text-white">

<CardContent className="p-6">

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

<div>

<p className="text-orange-100 text-sm">Avance Promedio</p>

<p className="text-3xl font-bold">

{Math.round(MOCK_PROJECTS.reduce((sum, p) => sum + p.globalProgress,
0) / MOCK_PROJECTS.length)}%

</p>

<p className="text-orange-100 text-xs">todos los proyectos</p>

```
</div>

<TrendingUp className="text-orange-100" size={32} />

</div>

</CardContent>

</Card>

</div>
```

```
{/* Menú Principal */}

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">

  <Link href="/dashboard" className="group">

    <Card className="hover:shadow-lg transition-all duration-200 border-2
border-transparent hover:border-blue-300 group-hover:scale-105">

      <CardContent className="p-6 text-center">

        <BarChart3 className="mx-auto mb-4 text-blue-600 group-hover:text-blue-
700" size={48} />

        <h3 className="text-xl font-bold text-gray-800 mb-2"><img alt="Bar chart icon" data-bbox="668 545 690 562" /> DASHBOARD</h3>

        <p className="text-gray-600 text-sm">Vista general y métricas</p>

      </CardContent>

    </Card>

  </Link>

  <Link href="/pisos" className="group">

    <Card className="hover:shadow-lg transition-all duration-200 border-2
border-transparent hover:border-green-300 group-hover:scale-105">

      <CardContent className="p-6 text-center">

        <Home className="mx-auto mb-4 text-green-600 group-hover:text-green-
700" size={48} />
```

```
<h3 className="text-xl font-bold text-gray-800 mb-2">🏠 PISOS</h3>
<p className="text-gray-600 text-sm">Estado por piso y departamento</p>
</CardContent>
</Card>
</Link>
```

```
<Link href="/equipos" className="group">
  <Card className="hover:shadow-lg transition-all duration-200 border-2
border-transparent hover:border-purple-300 group-hover:scale-105">
    <CardContent className="p-6 text-center">
      <Users className="mx-auto mb-4 text-purple-600 group-hover:text-purple-
700" size={48} />
      <h3 className="text-xl font-bold text-gray-800 mb-2">👷 EQUIPOS</h3>
      <p className="text-gray-600 text-sm">Gestión de cuadrillas</p>
    </CardContent>
  </Card>
</Link>
```

```
<Link href="/tareas" className="group">
  <Card className="hover:shadow-lg transition-all duration-200 border-2
border-transparent hover:border-orange-300 group-hover:scale-105">
    <CardContent className="p-6 text-center">
      <ClipboardList className="mx-auto mb-4 text-orange-600 group-hover:text-
orange-700" size={48} />
      <h3 className="text-xl font-bold text-gray-800 mb-2">📋 TAREAS</h3>
      <p className="text-gray-600 text-sm">Tareas pendientes hoy</p>
    </CardContent>
```

```
</Card>
```

```
</Link>
```

```
</div>
```


```
{/* Resumen de Proyectos */}
```

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

```
<Card>
```

```
<CardHeader>
```

```
<CardTitle className="flex items-center gap-2">
```

```
 Proyectos en Curso
```

```
</CardTitle>
```

```
</CardHeader>
```

```
<CardContent>
```

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

```
{MOCK_PROJECTS.map((project) => (
```

```
<div key={project.id} className="flex items-center justify-between p-4 bg-gray-50 rounded-lg hover:bg-gray-100 transition-colors">
```

```
<div className="flex-1">
```

```
<h4 className="font-semibold text-gray-800">{project.name}</h4>
```

```
<p className="text-sm text-gray-600">{project.address}</p>
```

```
<div className="flex items-center gap-4 mt-2 text-sm text-gray-500">
```

```
<span> {project.totalFloors} pisos</span>
```

```
<span> {project.totalFloors * project.unitsPerFloor} unidades</span>
```

```
</div>
```

```
</div>
```

```
<div className="text-right">
```

```
<div className="text-2xl font-bold text-blue-600 mb-1">
  {project.globalProgress}%
</div>

<div className="w-24 bg-gray-200 rounded-full h-2">
  <div
    className="bg-blue-600 h-2 rounded-full transition-all duration-300"
    style={{ width: `${project.globalProgress}%` }}
  />
</div>

<div className="text-xs text-gray-500 mt-1">
  Piso {project.currentFloor}
</div>


</div>

</div>

)}}
</div>

</CardContent>

</Card>
```

```
<Card>
  <CardHeader>
    <CardTitle className="flex items-center gap-2">
       Equipos de Trabajo
    </CardTitle>
  </CardHeader>
  <CardContent>
```

```

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

  {MOCK_TEAMS.slice(0, 6).map((team) => (

    <div key={team.id} className="flex items-center justify-between p-4 bg-
gray-50 rounded-lg">

      <div>

        <h4 className="font-semibold text-gray-800">{team.name}</h4>

        <p className="text-sm text-gray-600">{team.specialty}</p>

        <p className="text-xs text-gray-500">👤 {team.supervisor}</p>

      </div>

      <div className="text-right">

        <div className="bg-blue-100 text-blue-800 px-3 py-1 rounded-full text-sm
font-semibold">

          🟢 {team.activeProjects} proyecto{team.activeProjects !== 1 ? 's' : ''}

        </div>

      </div>

    </div>

  )})

</div>
</CardContent>
</Card>
</div>

```

```

{/* Acciones Rápidas */}

```

```

<Card>

```

```

<CardHeader>

```

```

<CardTitle>⚡ Acciones Rápidas</CardTitle>

```

```

</CardHeader>
<CardContent>
  <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
    <Button variant="primary" className="h-16 flex-col gap-2">
      <ClipboardList size={20} />
      <span className="text-xs">Nuevo Proyecto</span>
    </Button>
    <Button variant="success" className="h-16 flex-col gap-2">
      <Home size={20} />
      <span className="text-xs">Ver Pisos</span>
    </Button>
    <Button variant="warning" className="h-16 flex-col gap-2">
      <Users size={20} />
      <span className="text-xs">Gestionar Equipos</span>
    </Button>
    <Button variant="outline" className="h-16 flex-col gap-2">
      <BarChart3 size={20} />
      <span className="text-xs">Generar Reporte</span>
    </Button>
  </div>
</CardContent>
</Card>
</div>
</div>
)
}

```


✅ Criterios de Aceptación - Sprint 1:

1. ✅ **Proyecto configurado** con Next.js 14 y todas las dependencias estables
2. ✅ **Supabase integrado** sin Prisma, configuración básica lista
3. ✅ **Dashboard responsive** que funciona en tablets y móviles
4. ✅ **Componentes base** (Button, Card) funcionando correctamente
5. ✅ **Datos de ejemplo** cargados y mostrándose
6. ✅ **Estructura de carpetas** organizada y escalable
7. ✅ **Tailwind configurado** con colores del sistema
8. ✅ **TypeScript** funcionando sin errores

🔧 Testing Sprint 1:

Verificar que todo compila sin errores

npm run build

Ejecutar en desarrollo

npm run dev

Verificar responsive design en:

- Móviles (375px)



- Tablets (768px)

- Desktop (1024px)

🔒 SPRINT 2: Autenticación y Roles con Supabase

📊 Objetivos del Sprint:

- ✅ Implementar autenticación con Supabase Auth
- ✅ Crear sistema de roles (admin, supervisor, jefe cuadrilla, maestro)

-  Proteger rutas según roles
-  Crear páginas de login y gestión de perfiles

Paso 2.1: Configurar Tablas de Usuarios en Supabase

SQL para ejecutar en Supabase:

-- Habilitar RLS (Row Level Security)

```
ALTER TABLE auth.users ENABLE ROW LEVEL SECURITY;
```

-- Crear tabla de perfiles de usuario

```
CREATE TABLE public.user_profiles (
  id UUID REFERENCES auth.users(id) ON DELETE CASCADE PRIMARY KEY,
  email TEXT NOT NULL,
  full_name TEXT NOT NULL,
  role TEXT NOT NULL CHECK (role IN ('admin', 'supervisor', 'jefe_cuadrilla', 'maestro')),
  phone TEXT,
  specialty TEXT, -- Para jefes de cuadrilla y maestros
  assigned_projects INTEGER[], -- Array de IDs de proyectos asignados
  is_active BOOLEAN DEFAULT true,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

-- Crear función para manejar nuevos usuarios

```
CREATE OR REPLACE FUNCTION public.handle_new_user()
RETURNS TRIGGER AS $
BEGIN
  INSERT INTO public.user_profiles (id, email, full_name, role)
```

```

VALUES (
    NEW.id,
    NEW.email,
    COALESCE(NEW.raw_user_meta_data->>'full_name', 'Usuario'),
    COALESCE(NEW.raw_user_meta_data->>'role', 'maestro')
);

RETURN NEW;

END;

$ LANGUAGE plpgsql SECURITY DEFINER;


-- Crear trigger para nuevos usuarios

CREATE OR REPLACE TRIGGER on_auth_user_created

AFTER INSERT ON auth.users

FOR EACH ROW EXECUTE FUNCTION public.handle_new_user();


-- Políticas RLS para user_profiles

CREATE POLICY "Users can view own profile" ON public.user_profiles

FOR SELECT USING (auth.uid() = id);


CREATE POLICY "Users can update own profile" ON public.user_profiles

FOR UPDATE USING (auth.uid() = id);


-- Admins pueden ver todos los perfiles

CREATE POLICY "Admins can view all profiles" ON public.user_profiles

FOR SELECT USING (
    EXISTS (

```

```
SELECT 1 FROM public.user_profiles
WHERE id = auth.uid() AND role = 'admin'
)
);
```

-- Habilitar RLS en la tabla

```
ALTER TABLE public.user_profiles ENABLE ROW LEVEL SECURITY;
```

-- Crear tabla de sesiones de usuario para tracking

```
CREATE TABLE public.user_sessions (
  id SERIAL PRIMARY KEY,
  user_id UUID REFERENCES public.user_profiles(id) ON DELETE CASCADE,
  login_at TIMESTAMPTZ WITH TIME ZONE DEFAULT NOW(),
  ip_address TEXT,
  user_agent TEXT
);
```

```
ALTER TABLE public.user_sessions ENABLE ROW LEVEL SECURITY;
```

-- Insertar usuario admin por defecto (cambiar email y contraseña)

```
INSERT INTO auth.users (
  instance_id,
  id,
  aud,
  role,
  email,
```

```
encrypted_password,  
email_confirmed_at,  
recovery_sent_at,  
last_sign_in_at,  
raw_app_meta_data,  
raw_user_meta_data,  
created_at,  
updated_at,  
confirmation_token,  
email_change,  
email_change_token_new,  
recovery_token  
) VALUES (  
  '00000000-0000-0000-0000-000000000000',  
  gen_random_uuid(),  
  'authenticated',  
  'authenticated',  
  'admin@sistema.com',  
  crypt('admin123', gen_salt('bf')), -- Cambiar esta contraseña  
  NOW(),  
  NOW(),  
  NOW(),  
  '{"provider":"email","providers":["email"]}',  
  '{"full_name":"Administrador del Sistema","role":"admin"}',  
  NOW(),  
  NOW(),
```

```
"  
"  
"  
"  
"
```

```
);
```

Paso 2.2: Crear Contexto de Autenticación

Archivo: src/contexts/AuthContext.tsx

```
'use client'
```

```
import { createContext, useContext, useEffect, useState } from 'react'
```

```
import { User, Session } from '@supabase/supabase-js'
```

```
import { supabase } from '@lib/supabase'
```

```
import { useRouter } from 'next/navigation'
```

```
import toast from 'react-hot-toast'
```

```
interface UserProfile {
```

```
  id: string
```

```
  email: string
```

```
  full_name: string
```

```
  role: 'admin' | 'supervisor' | 'jefe_cuadrilla' | 'maestro'
```

```
  phone?: string
```

```
  specialty?: string
```

```
  assigned_projects?: number[]
```

```
  is_active: boolean
```

```
}
```

```

interface AuthContextType {
  user: User | null
  profile: UserProfile | null
  session: Session | null
  loading: boolean
  signIn: (email: string, password: string) => Promise<{ error: any }>
  signOut: () => Promise<void>
  hasRole: (roles: string | string[]) => boolean
  canAccessProject: (projectId: number) => boolean
  refreshProfile: () => Promise<void>
}

```

```

const AuthContext = createContext<AuthContextType | undefined>(undefined)

```

```

export function AuthProvider({ children }: { children: React.ReactNode }) {
  const [user, setUser] = useState<User | null>(null)
  const [profile, setProfile] = useState<UserProfile | null>(null)
  const [session, setSession] = useState<Session | null>(null)
  const [loading, setLoading] = useState(true)
  const router = useRouter()

  useEffect(() => {
    // Obtener sesión inicial
    supabase.auth.getSession().then(({ data: { session } }) => {
      setSession(session)
      setUser(session?.user ?? null)
    })
  })
}

```

```
if (session?.user) {  
  fetchUserProfile(session.user.id)  
} else {  
  setLoading(false)  
}  
})
```

// Escuchar cambios en la autenticación

```
const {  
  data: { subscription },  
} = supabase.auth.onAuthStateChange(async (event, session) => {  
  setSession(session)  
  setUser(session?.user ?? null)
```

```
if (event === 'SIGNED_IN' && session?.user) {  
  await fetchUserProfile(session.user.id)  
  toast.success('Sesión iniciada correctamente')  
} else if (event === 'SIGNED_OUT') {  
  setProfile(null)  
  setLoading(false)  
  toast.success('Sesión cerrada')  
}  
})
```

```
return () => subscription.unsubscribe()  
}, [])
```



```
const fetchUserProfile = async (userId: string) => {
```

```
  try {
```

```
    const { data, error } = await supabase
```

```
      .from('user_profiles')
```

```
      .select('*')
```

```
      .eq('id', userId)
```

```
      .single()
```

```
    if (error) throw error
```

```
    setProfile(data)
```

```
  } catch (error) {
```

```
    console.error('Error fetching profile:', error)
```

```
    toast.error('Error al cargar el perfil de usuario')
```

```
  } finally {
```

```
    setLoading(false)
```

```
  }
```

```
}
```

```
const signIn = async (email: string, password: string) => {
```

```
  setLoading(true)
```

```
  try {
```

```
    const { data, error } = await supabase.auth.signInWithPassword({
```

```
      email,
```

```
      password,
```

```
  })
```

```
  if (error) throw error
```

```
  // Registrar sesión
```

```
  if (data.user) {
```

```
    await supabase
```

```
      .from('user_sessions')
```

```
      .insert({
```

```
        user_id: data.user.id,
```

```
        ip_address: 'unknown', // En producción obtener IP real
```

```
        user_agent: navigator.userAgent
```

```
      })
```

```
    }
```

```
    return { error: null }
```

```
  } catch (error: any) {
```

```
    toast.error(error.message || 'Error al iniciar sesión')
```

```
    return { error }
```

```
  } finally {
```

```
    setLoading(false)
```

```
  }
```

```
}
```

```
const signOut = async () => {
```

```
  try {
```

```
    await supabase.auth.signOut()
    setUser(null)
    setProfile(null)
    setSession(null)
    router.push('/login')
  } catch (error: any) {
    toast.error('Error al cerrar sesión')
  }
}
```

```
const hasRole = (roles: string | string[]): boolean => {
  if (!profile) return false
  const roleArray = Array.isArray(roles) ? roles : [roles]
  return roleArray.includes(profile.role)
}
```

```
const canAccessProject = (projectId: number): boolean => {
  if (!profile) return false
```

```
  // Admin y supervisor pueden acceder a todos los proyectos
```

```
  if (profile.role === 'admin' || profile.role === 'supervisor') return true
```

```
  // Jefe cuadrilla y maestro solo a proyectos asignados
```

```
  return profile.assigned_projects?.includes(projectId) ?? false
```

```
}
```

```
const refreshProfile = async () => {  
  if (user) {  
    await fetchUserProfile(user.id)  
  }  
}
```

```
const value: AuthContextType = {  
  user,  
  profile,  
  session,  
  loading,  
  signIn,  
  signOut,  
  hasRole,  
  canAccessProject,  
  refreshProfile,  
}
```

```
return (  
  <AuthContext.Provider value={value}>  
    {children}  
  </AuthContext.Provider>  
)  
}
```

```
export const useAuth = () => {
```

```

const context = useContext(AuthContext)

if (context === undefined) {
  throw new Error('useAuth must be used within an AuthProvider')
}

return context
}

```

Paso 2.3: Crear Componente de Protección de Rutas

Archivo: src/components/auth/ProtectedRoute.tsx

'use client'

```

import { useAuth } from '@contexts/AuthContext'
import { useRouter } from 'next/navigation'
import { useEffect, ReactNode } from 'react'
import { Card, CardContent } from '@components/ui/Card'
import { Loader2 } from 'lucide-react'

```

```

interface ProtectedRouteProps {
  children: ReactNode
  requiredRoles?: string[]
  requiredProject?: number
  fallback?: ReactNode
}

```

```

export function ProtectedRoute({
  children,
  requiredRoles = [],

```

```

    requiredProject,
    fallback
  }: ProtectedRouteProps) {
    const { user, profile, loading, hasRole, canAccessProject } = useAuth()
    const router = useRouter()

    useEffect(() => {
      if (!loading && !user) {
        router.push('/login')
      }
    }, [user, loading, router])

    // Mostrar loading
    if (loading) {
      return (
        <div className="min-h-screen flex items-center justify-center bg-gradient-to-br
from-gray-50 to-blue-50">
          <Card className="w-96">
            <CardContent className="p-8 text-center">
              <Loader2 className="mx-auto mb-4 animate-spin text-blue-600" size={48} />
              <h3 className="text-lg font-semibold text-gray-800 mb-2">
                Cargando...
              </h3>
              <p className="text-gray-600">
                Verificando permisos de usuario
              </p>
            </CardContent>
          </Card>
        </div>
      )
    }
  }
}

```

```
    </CardContent>
  </Card>
</div>
)
}
```

```
// Usuario no autenticado
```

```
if (!user || !profile) {
  return null // El useEffect redirigirá a login
}
```

```
// Verificar roles requeridos
```

```
if (requiredRoles.length > 0 && !hasRole(requiredRoles)) {
  return fallback || (
    <div className="min-h-screen flex items-center justify-center bg-gradient-to-br
from-gray-50 to-blue-50">
      <Card className="w-96">
        <CardContent className="p-8 text-center">
          <div className="text-6xl mb-4">🚫</div>
          <h3 className="text-lg font-semibold text-gray-800 mb-2">
            Acceso Denegado
          </h3>
          <p className="text-gray-600 mb-4">
            No tienes permisos para acceder a esta sección.
          </p>
          <p className="text-sm text-gray-500">
```

```

        Tu rol actual: <span className="font-semibold">{profile.role}</span>

    </p>

</CardContent>

</Card>

</div>

)

}

// Verificar acceso a proyecto específico

if (requiredProject && !canAccessProject(requiredProject)) {

    return fallback || (

        <div className="min-h-screen flex items-center justify-center bg-gradient-to-br
from-gray-50 to-blue-50">

            <Card className="w-96">

                <CardContent className="p-8 text-center">

                    <div className="text-6xl mb-4">🔒</div>

                    <h3 className="text-lg font-semibold text-gray-800 mb-2">

                        Proyecto No Asignado

                    </h3>

                    <p className="text-gray-600">

                        No tienes acceso a este proyecto específico.

                    </p>

                </CardContent>

            </Card>

        </div>

    )

}

```



```
}
```

```
return <>{children}</>
```

```
}
```



Paso 2.4: Página de Login

Archivo: src/app/login/page.tsx

```
'use client'
```

```
import { useState, useEffect } from 'react'
```

```
import { useAuth } from '@contexts/AuthContext'
```

```
import { useRouter } from 'next/navigation'
```

```
import { Card, CardHeader, CardTitle, CardContent } from '@components/ui/Card'
```

```
import { Button } from '@components/ui/Button'
```

```
import { Building, Eye, EyeOff, Loader2 } from 'lucide-react'
```

```
import toast from 'react-hot-toast'
```

```
export default function LoginPage() {
```

```
  const [email, setEmail] = useState('')
```

```
  const [password, setPassword] = useState('')
```

```
  const [showPassword, setShowPassword] = useState(false)
```

```
  const [isLoading, setIsLoading] = useState(false)
```

```
  const { signIn, user } = useAuth()
```

```
  const router = useRouter()
```

```
  useEffect(() => {
```

```
    // Redirigir si ya está autenticado
```

```
if (user) {  
  router.push('/')  
}  
}, [user, router])
```

```
const handleSubmit = async (e: React.FormEvent) => {  
  e.preventDefault()  
  setIsLoading(true)
```

```
  try {  
    const { error } = await signIn(email, password)
```

```
    if (error) {  
      toast.error('Credenciales incorrectas')  
    } else {  
      router.push('/')  
    }  
  } catch (error) {  
    toast.error('Error inesperado al iniciar sesión')  
  } finally {  
    setIsLoading(false)  
  }  
}
```

```
return (
```

```
<div className="min-h-screen flex items-center justify-center bg-gradient-to-br
from-blue-50 to-indigo-100 p-4">
```

```
<Card className="w-full max-w-md">
```

```
<CardHeader className="text-center">
```

```
<div className="flex justify-center mb-4">
```

```
<div className="bg-blue-600 p-3 rounded-full">
```

```
<Building className="text-white" size={32} />
```

```
</div>
```

```
</div>
```

```
<CardTitle className="text-2xl font-bold text-gray-800">
```

```
Sistema Control Terminaciones
```

```
</CardTitle>
```

```
<p className="text-gray-600 mt-2">
```

```
Ingresa tus credenciales para acceder
```

```
</p>
```

```
</CardHeader>
```

```
<CardContent>
```

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

```
<div>
```

```
<label htmlFor="email" className="block text-sm font-medium text-gray-700
mb-2">
```

```
Correo Electrónico
```

```
</label>
```

```
<input
```

```
id="email"
```

```
type="email"
```

```

    value={email}

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

    className="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-transparent"

    placeholder="usuario@empresa.com"

    required

    disabled={isLoading}

  />
</div>

<div>
  <label htmlFor="password" className="block text-sm font-medium text-gray-
700 mb-2">
    Contraseña
  </label>

  <div className="relative">

    <input

      id="password"

      type={showPassword ? 'text' : 'password'}

      value={password}

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

      className="w-full px-3 py-2 pr-10 border border-gray-300 rounded-lg
focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-transparent"

      placeholder="●●●●●●●●"

      required

      disabled={isLoading}

    />
  </div>
</div>

```

```

<button
  type="button"
  onClick={() => setShowPassword(!showPassword)}
  className="absolute right-3 top-1/2 transform -translate-y-1/2 text-gray-400
hover:text-gray-600"
  disabled={isLoading}
>
  {showPassword ? <EyeOff size={20} /> : <Eye size={20} />}
</button>
</div>
</div>

```

```

<Button
  type="submit"
  className="w-full"
  disabled={isLoading}
  size="lg"
>
  {isLoading ? (
    <>
      <Loader2 className="mr-2 animate-spin" size={20} />
      Iniciando sesión...
    </>
  ) : (
    'Iniciar Sesión'
  )}

```

```

    </Button>

  </form>

  { /* Credenciales de ejemplo para desarrollo */ }

  <div className="mt-6 p-4 bg-gray-50 rounded-lg">

    <h4 className="text-sm font-semibold text-gray-700 mb-2">

      🔑 Credenciales de desarrollo:

    </h4>

    <div className="text-xs text-gray-600 space-y-1">

      <p><strong>Admin:</strong> admin@sistema.com / admin123</p>

      <p><strong>Supervisor:</strong> supervisor@sistema.com / super123</p>

      <p><strong>Jefe Cuadrilla:</strong> jefe@sistema.com / jefe123</p>

      <p><strong>Maestro:</strong> maestro@sistema.com / maestro123</p>

    </div>

  </div>

</CardContent>

</Card>

</div>

)

}

```

Paso 2.5: Actualizar Layout Principal

Archivo: src/app/layout.tsx (actualizar)

```

import './globals.css'

import { Inter } from 'next/font/google'

import { Toaster } from 'react-hot-toast'

import { AuthProvider } from '@contexts/AuthContext'

```

```
const inter = Inter({ subsets: ['latin'] })
```

```
export const metadata = {
```

```
  title: 'Sistema Control Terminaciones',
```

```
  description: 'Sistema de control y seguimiento de terminaciones en proyectos de construcción',
```

```
}
```

```
export default function RootLayout({
```

```
  children,
```

```
}: {
```

```
  children: React.ReactNode
```

```
}) {
```

```
  return (
```

```
    <html lang="es">
```

```
      <body className={inter.className}>
```

```
        <AuthProvider>
```

```
          <div className="min-h-screen bg-gradient-to-br from-gray-50 to-blue-50">
```

```
            {children}
```

```
          </div>
```

```
          <Toaster
```

```
            position="top-right"
```

```
            toastOptions={{
```

```
              duration: 4000,
```

```
              style: {
```

```

        background: '#363636',
        color: '#fff',
      },
    }}
  />
</AuthProvider>
</body>
</html>
)
}

```

🕒 Paso 2.6: Crear Layout para Rutas Protegidas

Archivo: src/app/(auth)/layout.tsx

```
'use client'
```


```

import { useAuth } from '@contexts/AuthContext'
import { ProtectedRoute } from '@components/auth/ProtectedRoute'
import { Button } from '@components/ui/Button'
import { Building, User, LogOut, Menu, X } from 'lucide-react'
import { useState } from 'react'
import Link from 'next/link'

const navigation = [
  { name: 'Dashboard', href: '/', icon: '🏠', roles: ['admin', 'supervisor', 'jefe_cuadrilla', 'maestro'] },
  { name: 'Pisos', href: '/pisos', icon: '🏠', roles: ['admin', 'supervisor', 'jefe_cuadrilla'] },
  { name: 'Equipos', href: '/equipos', icon: '👤', roles: ['admin', 'supervisor'] },

```



```
{ name: 'Reportes', href: '/reportes', icon: '
```

```
export default function AuthLayout({
  children,
}): {
  children: React.ReactNode
} {
  const [sidebarOpen, setSidebarOpen] = useState(false)

  return (
    <ProtectedRoute>
      <AuthLayoutContent sidebarOpen={sidebarOpen}
setSidebarOpen={setSidebarOpen}>
        {children}
      </AuthLayoutContent>
    </ProtectedRoute>
  )
}
```

```
function AuthLayoutContent({
  children,
  sidebarOpen,
  setSidebarOpen
}): {
  children: React.ReactNode
```

```

sidebarOpen: boolean

setSidebarOpen: (open: boolean) => void

}) {

  const { profile, signOut, hasRole } = useAuth()

  const filteredNavigation = navigation.filter(item =>
    hasRole(item.roles)
  )

  return (
    <div className="flex h-screen bg-gray-100">
      {/* Sidebar móvil */}
      <div className={`lg:hidden fixed inset-0 flex z-40 ${sidebarOpen ? '' : 'pointer-events-none'} `}>
        <div
          className={`fixed inset-0 bg-gray-600 bg-opacity-75 transition-opacity ${
            sidebarOpen ? 'opacity-100' : 'opacity-0'
          } `}
          onClick={() => setSidebarOpen(false)}
        />

        <div className={`relative flex-1 flex flex-col max-w-xs w-full bg-white transform
          transition-transform ${
            sidebarOpen ? 'translate-x-0' : '-translate-x-full'
          } `}>
          <div className="absolute top-0 right-0 -mr-12 pt-2">
            <button

```

```

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

        className="ml-1 flex items-center justify-center h-10 w-10 rounded-full
focus:outline-none focus:ring-2 focus:ring-inset focus:ring-white"

        >

        <X className="h-6 w-6 text-white" />

    </button>

</div>

```

```

        <SidebarContent navigation={filteredNavigation} profile={profile}
signOut={signOut} />

    </div>

</div>

```

```

    {/* Sidebar desktop */}

    <div className="hidden lg:flex lg:flex-shrink-0">

        <div className="flex flex-col w-64">

            <SidebarContent navigation={filteredNavigation} profile={profile}
signOut={signOut} />

        </div>

    </div>

```

```

    {/* Contenido principal */}

    <div className="flex-1 overflow-hidden flex flex-col">

        {/* Header móvil */}

        <div className="lg:hidden bg-white shadow-sm px-4 py-2 flex items-center
justify-between">

            <button

```

```

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

        className="p-2 rounded-md text-gray-400 hover:text-gray-500 hover:bg-gray-
100"

        >

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

    </button>

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

        <Building className="text-blue-600" size={24} />

        <span className="font-semibold text-gray-800">Control
Terminaciones</span>

    </div>

    <div className="w-8" /> { /* Spacer */ }

</div>

    { /* Contenido */ }

    <main className="flex-1 overflow-auto">

        {children}

    </main>

</div>

</div>

)

}

```

```

function SidebarContent({
    navigation,

```

```

    profile,
    signOut
  }: {
    navigation: any[]
    profile: any
    signOut: () => void
  }) {
    return (
      <div className="flex flex-col h-full bg-white shadow-lg">
        {/* Header del sidebar */}
        <div className="flex items-center gap-3 px-4 py-6 border-b">
          <div className="bg-blue-600 p-2 rounded-lg">
            <Building className="text-white" size={24} />
          </div>
          <div>
            <h2 className="text-lg font-bold text-gray-800">Control</h2>
            <p className="text-sm text-gray-600">Terminaciones</p>
          </div>
        </div>

        {/* Navegación */}
        <nav className="flex-1 px-4 py-6 space-y-2">
          {navigation.map((item) => (
            <Link
              key={item.name}
              href={item.href}

```

```
      className="flex items-center gap-3 px-3 py-2 text-gray-700 rounded-lg
      hover:bg-blue-50 hover:text-blue-700 transition-colors"
```

```
>
```

```
<span className="text-lg">{item.icon}</span>
```

```
<span className="font-medium">{item.name}</span>
```

```
</Link>
```

```
    )}
```

```
</nav>
```

```
{/* Perfil de usuario */}
```

```
<div className="border-t p-4">
```

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

```
<div className="bg-gray-200 p-2 rounded-full">
```

```
<User className="text-gray-600" size={20} />
```

```
</div>
```

```
<div className="flex-1">
```

```
<p className="font-medium text-gray-800 text-sm">{profile?.full_name}</p>
```

```
<p className="text-xs text-gray-500 capitalize">{profile?.role?.replace('_', '
')}}</p>
```

```
</div>
```

```
</div>
```

```
<Button
```

```
  onClick={signOut}
```

```
  variant="outline"
```

```
  size="sm"
```

```
  className="w-full flex items-center gap-2"
```

```

    >
    <Logout size={16} />
    Cerrar Sesión
  </Button>
</div>
</div>
)
}

```

Paso 2.7: Actualizar Página Principal

Archivo: src/app/(auth)/page.tsx (mover desde src/app/page.tsx)

```

import { ProtectedRoute } from '@components/auth/ProtectedRoute'
import DashboardContent from '@components/dashboard/DashboardContent'

export default function HomePage() {
  return <DashboardContent />
}

```

Archivo: src/components/dashboard/DashboardContent.tsx

```

'use client'

import Link from 'next/link'

import { Building, Home, Users, ClipboardList, BarChart3, TrendingUp } from 'lucide-react'

import { Card, CardHeader, CardTitle, CardContent } from '@components/ui/Card'
import { Button } from '@components/ui/Button'

import { MOCK_PROJECTS, MOCK_TEAMS } from '@data/mockData'
import { useAuth } from '@contexts/AuthContext'

```

```

export default function DashboardContent() {

  const { profile, hasRole } = useAuth()

  const totalProjects = MOCK_PROJECTS.length

  const activeProjects = MOCK_PROJECTS.filter(p => p.status === 'active').length

  const completedUnits = MOCK_PROJECTS.reduce((sum, project) => {

    return sum + Math.floor((project.globalProgress / 100) * (project.totalFloors *
project.unitsPerFloor))

  }, 0)

  const totalUnits = MOCK_PROJECTS.reduce((sum, project) => {

    return sum + (project.totalFloors * project.unitsPerFloor)

  }, 0)

  const activeTeams = MOCK_TEAMS.length

  return (

    <div className="p-4 md:p-6 space-y-6">

      {/* Bienvenida personalizada */}

      <Card className="bg-gradient-to-r from-blue-600 to-blue-700 text-white">

        <CardContent className="p-6">

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

            <div>

              <h1 className="text-2xl md:text-3xl font-bold mb-2">

                ¡Hola, {profile?.full_name}! 🤖

              </h1>

              <p className="text-blue-100">

```


Rol: {profile?.role?.replace('_', ' ')}

</p>

<p className="text-blue-100 text-sm mt-1">

Última actualización: {new Date().toLocaleDateString('es-CL')}

</p>

</div>

<Building size={48} className="text-blue-200" />

</div>

</CardContent>

</Card>

{/* Métricas - Visibles según rol */}

{hasRole(['admin', 'supervisor']) && (

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">

<Card className="bg-gradient-to-r from-green-500 to-green-600 text-white">

<CardContent className="p-6">

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

<div>

<p className="text-green-100 text-sm">Proyectos Activos</p>

<p className="text-3xl font-bold">{activeProjects}</p>

<p className="text-green-100 text-xs">de {totalProjects} totales</p>

</div>

<Building className="text-green-100" size={32} />

</div>

</CardContent>

</Card>

<Card className="bg-gradient-to-r from-blue-500 to-blue-600 text-white">

<CardContent className="p-6">

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

<div>

<p className="text-blue-100 text-sm">Unidades Terminadas</p>

<p className="text-3xl font-bold">{completedUnits}</p>

<p className="text-blue-100 text-xs">de {totalUnits} totales</p>

</div>

<Home className="text-blue-100" size={32} />

</div>

</CardContent>

</Card>

<Card className="bg-gradient-to-r from-purple-500 to-purple-600 text-white">

<CardContent className="p-6">

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

<div>

<p className="text-purple-100 text-sm">Equipos Activos</p>

<p className="text-3xl font-bold">{activeTeams}</p>

<p className="text-purple-100 text-xs">especialidades</p>

</div>

<Users className="text-purple-100" size={32} />

</div>

</CardContent>

```

</Card>

<Card className="bg-gradient-to-r from-orange-500 to-orange-600 text-white">
  <CardContent className="p-6">
    <div className="flex items-center justify-between">
      <div>
        <p className="text-orange-100 text-sm">Avance Promedio</p>
        <p className="text-3xl font-bold">
          {Math.round(MOCK_PROJECTS.reduce((sum, p) => sum + p.globalProgress,
0) / MOCK_PROJECTS.length)}%
        </p>
        <p className="text-orange-100 text-xs">todos los proyectos</p>
      </div>
      <TrendingUp className="text-orange-100" size={32} />
    </div>
  </CardContent>
</Card>
</div>
  )}

```

```

{/* Accesos rápidos según rol */}

```

```

<Card>
  <CardHeader>
    <CardTitle>⚡ Accesos Rápidos</CardTitle>
  </CardHeader>
  <CardContent>

```

```

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

  {hasRole(['admin', 'supervisor', 'jefe_cuadrilla']) && (

    <Link href="/pisos">

      <Button variant="primary" className="h-16 w-full flex-col gap-2 hover:scale-
105 transition-transform">

        <Home size={20} />

        <span className="text-xs">Ver Pisos</span>

      </Button>

    </Link>

  )}

  {hasRole(['admin', 'supervisor']) && (

    <Link href="/equipos">

      <Button variant="success" className="h-16 w-full flex-col gap-2 hover:scale-
105 transition-transform">

        <Users size={20} />

        <span className="text-xs">Gestionar Equipos</span>

      </Button>

    </Link>

  )}

  <Button variant="warning" className="h-16 w-full flex-col gap-2 hover:scale-
105 transition-transform">

    <ClipboardList size={20} />

    <span className="text-xs">Mis Tareas</span>

  </Button>

```

```

    {hasRole(['admin']) && (
      <Link href="/reportes">
        <Button variant="outline" className="h-16 w-full flex-col gap-2 hover:scale-105 transition-transform">
          <BarChart3 size={20} />
          <span className="text-xs">Reportes</span>
        </Button>
      </Link>
    )}
  </div>
</CardContent>
</Card>

```

```

{/* Vista específica por rol */}
{hasRole(['jefe_cuadrilla', 'maestro']) && (
  <Card>
    <CardHeader>
      <CardTitle>📋 Mis Proyectos Asignados</CardTitle>
    </CardHeader>
    <CardContent>
      <div className="space-y-4">
        {/* Filtrar proyectos según asignación del usuario */}
        {MOCK_PROJECTS.slice(0, 2).map((project) => (
          <div key={project.id} className="flex items-center justify-between p-4 bg-blue-50 rounded-lg">
            <div>
              <h4 className="font-semibold text-gray-800">{project.name}</h4>

```

```
        <p className="text-sm text-gray-600">Trabajando en piso  
{project.currentFloor}</p>
```

```
        <p className="text-xs text-gray-500 mt-1">
```

```
            {profile?.specialty && `Especialidad: ${profile.specialty}`} 
```

```
        </p>
```

```
    </div>
```

```
    <div className="text-right">
```

```
        <div className="text-2xl font-bold text-blue-  
600">{project.globalProgress}%</div>
```

```
        <Link href={` /pisos/${project.id}`} >
```

```
            <Button size="sm" className="mt-2">
```

```
                Ver Detalles
```

```
            </Button>
```

```
        </Link>
```

```
    </div>
```

```
</div>
```

```
    )}
```

```
</div>
```

```
</CardContent>
```

```
</Card>
```

```
)}
```

```
{/* Resumen para admin/supervisor */}
```

```
{hasRole(['admin', 'supervisor']) && (
```

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

```
        <Card>
```

```
            <CardHeader>
```

```

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

  🏗️ Proyectos en Curso

</CardTitle>

</CardHeader>

<CardContent>

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

    {MOCK_PROJECTS.map((project) => (

      <div key={project.id} className="flex items-center justify-between p-4 bg-
gray-50 rounded-lg hover:bg-gray-100 transition-colors">

        <div className="flex-1">

          <h4 className="font-semibold text-gray-800">{project.name}</h4>

          <p className="text-sm text-gray-600">{project.address}</p>

          <div className="flex items-center gap-4 mt-2 text-sm text-gray-500">

            <span> 🏢 {project.totalFloors} pisos</span>

            <span> 🏠 {project.totalFloors * project.unitsPerFloor} unidades</span>

          </div>

        </div>

        <div className="text-right">

          <div className="text-2xl font-bold text-blue-600 mb-1">

            {project.globalProgress}%

          </div>

          <div className="w-24 bg-gray-200 rounded-full h-2">

            <div

              className="bg-blue-600 h-2 rounded-full transition-all duration-300"

              style={{ width: `${project.globalProgress}%` }}

            />


```

```

    </div>

    <div className="text-xs text-gray-500 mt-1">

      Piso {project.currentFloor}

    </div>

  </div>

</div>

  )}}

</div>


</CardContent>

</Card>

<Card>

  <CardHeader>

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

       Equipos de Trabajo

    </CardTitle>

  </CardHeader>

  <CardContent>

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


      {MOCK_TEAMS.slice(0, 6).map((team) => (

        <div key={team.id} className="flex items-center justify-between p-4 bg-
gray-50 rounded-lg">

          <div>

            <h4 className="font-semibold text-gray-800">{team.name}</h4>

            <p className="text-sm text-gray-600">{team.specialty}</p>

            <p className="text-xs text-gray-500"> {team.supervisor}</p>

```




```

    </div>

    <div className="text-right">

        <div className="bg-green-100 text-green-800 px-3 py-1 rounded-full text-sm font-semibold">

             {team.activeProjects} proyecto{team.activeProjects !== 1 ? 's' : ''}

        </div>

    </div>

</div>

    )}

</div>

</CardContent>

</Card>

</div>

    )}

</div>

)

}

```



Paso 2.8: Crear Usuarios de Prueba en Supabase

SQL adicional para crear usuarios de prueba:

-- Insertar usuarios de prueba (ejecutar en SQL Editor de Supabase)

-- Supervisor

INSERT INTO auth.users (

instance_id, id, aud, role, email, encrypted_password, email_confirmed_at,

recovery_sent_at, last_sign_in_at, raw_app_meta_data, raw_user_meta_data,

created_at, updated_at, confirmation_token, email_change,

email_change_token_new, recovery_token

```

) VALUES (
  '00000000-0000-0000-0000-000000000000', gen_random_uuid(), 'authenticated',
  'authenticated',
  'supervisor@sistema.com', crypt('super123', gen_salt('bf')), NOW(), NOW(), NOW(),
  '{"provider":"email","providers":["email"]}',
  '{"full_name":"Carlos Supervisor","role":"supervisor"}',
  NOW(), NOW(), " ", " ", " "
);

```

-- Jefe de Cuadrilla

```

INSERT INTO auth.users (
  instance_id, id, aud, role, email, encrypted_password, email_confirmed_at,
  recovery_sent_at, last_sign_in_at, raw_app_meta_data, raw_user_meta_data,
  created_at, updated_at, confirmation_token, email_change,
  email_change_token_new, recovery_token
) VALUES (
  '00000000-0000-0000-0000-000000000000', gen_random_uuid(), 'authenticated',
  'authenticated',
  'jefe@sistema.com', crypt('jefe123', gen_salt('bf')), NOW(), NOW(), NOW(),
  '{"provider":"email","providers":["email"]}',
  '{"full_name":"Miguel Jefe Cuadrilla","role":"jefe_cuadrilla","specialty":"Pintura"}',
  NOW(), NOW(), " ", " ", " "
);

```

-- Maestro

```

INSERT INTO auth.users (
  instance_id, id, aud, role, email, encrypted_password, email_confirmed_at,

```

```

recovery_sent_at, last_sign_in_at, raw_app_meta_data, raw_user_meta_data,
created_at, updated_at, confirmation_token, email_change,
email_change_token_new, recovery_token
) VALUES (
'00000000-0000-0000-0000-000000000000', gen_random_uuid(), 'authenticated',
'authenticated',
'maestro@sistema.com', crypt('maestro123', gen_salt('bf')), NOW(), NOW(), NOW(),
 '{"provider":"email","providers":["email"]}',
 '{"full_name":"Juan Maestro","role":"maestro","specialty":"Instalaciones"}',
NOW(), NOW(), ",", ",", "
);

```

✅ Criterios de Aceptación - Sprint 2:

1. ✅ **Autenticación funcional** con Supabase Auth
2. ✅ **Sistema de roles** implementado (admin, supervisor, jefe_cuadrilla, maestro)
3. ✅ **Rutas protegidas** según permisos de usuario
4. ✅ **Dashboard personalizado** según rol del usuario
5. ✅ **Login/logout** funcional con manejo de errores
6. ✅ **Navegación adaptable** según permisos
7. ✅ **Layout responsive** para móviles y tablets

🔧 Testing Sprint 2:

Compilar y verificar

npm run build

Probar en desarrollo

npm run dev

Pruebas a realizar:

1. Login con cada tipo de usuario

2. Verificar accesos según roles

3. Navegación en móvil y desktop

4. Logout y redirección

5. Acceso a rutas protegidas sin auth



SPRINT 3: Mantenedores Base



Objetivos del Sprint:

- ☒ Crear mantenedores de proyectos, pisos y departamentos
- ☒ Implementar CRUD completo con Supabase
- ☒ Crear formularios con validación
- ☒ Gestión de equipos y actividades base



Paso 3.1: Completar Schema de Base de Datos

SQL para ejecutar en Supabase:

-- Tabla de Proyectos

```
CREATE TABLE IF NOT EXISTS public.projects (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(255) NOT NULL,  
  address TEXT,  
  total_floors INTEGER NOT NULL DEFAULT 1,  
  units_per_floor INTEGER NOT NULL DEFAULT 1,  
  start_date DATE,  
  estimated_completion DATE,
```

```
actual_completion DATE,  
  
status VARCHAR(50) DEFAULT 'planning' CHECK (status IN ('planning', 'active',  
'completed', 'paused')),  
  
created_by UUID REFERENCES public.user_profiles(id),  
  
created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),  
  
updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()  
);
```

-- Tabla de Pisos

```
CREATE TABLE IF NOT EXISTS public.floors (  
  
id SERIAL PRIMARY KEY,  
  
project_id INTEGER REFERENCES public.projects(id) ON DELETE CASCADE,  
  
floor_number INTEGER NOT NULL,  
  
status VARCHAR(50) DEFAULT 'pending' CHECK (status IN ('pending', 'in-progress',  
'completed')),  
  
estimated_completion DATE,  
  
actual_completion DATE,  
  
delay_days INTEGER DEFAULT 0,  
  
notes TEXT,  
  
created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),  
  
updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),  
  
UNIQUE(project_id, floor_number)  
);
```

-- Tabla de Departamentos

```
CREATE TABLE IF NOT EXISTS public.apartments (  
  
id SERIAL PRIMARY KEY,
```

```

floor_id INTEGER REFERENCES public.floors(id) ON DELETE CASCADE,
apartment_number VARCHAR(20) NOT NULL,
apartment_type VARCHAR(100), -- 3D+2B, 2D+1B, etc.
area DECIMAL(8,2),
status VARCHAR(50) DEFAULT 'pending' CHECK (status IN ('pending', 'in-progress',
'completed')),
estimated_completion DATE,
actual_completion DATE,
notes TEXT,
created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
UNIQUE(floor_id, apartment_number)
);

```

-- Tabla de Actividades Base (template)

```

CREATE TABLE IF NOT EXISTS public.activity_templates (
id SERIAL PRIMARY KEY,
name VARCHAR(255) NOT NULL,
category VARCHAR(100) NOT NULL, -- Estructura, Instalaciones, Acabados, etc.
estimated_hours INTEGER NOT NULL DEFAULT 8,
sort_order INTEGER DEFAULT 0,
dependencies INTEGER[], -- Array de IDs de actividades que deben completarse
antes
is_active BOOLEAN DEFAULT true,
created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

```

-- Tabla de Actividades por Departamento

```
CREATE TABLE IF NOT EXISTS public.apartment_activities (  
  id SERIAL PRIMARY KEY,  
  apartment_id INTEGER REFERENCES public.apartments(id) ON DELETE CASCADE,  
  activity_template_id INTEGER REFERENCES public.activity_templates(id),  
  status VARCHAR(50) DEFAULT 'pending' CHECK (status IN ('pending', 'in-progress',  
'completed', 'blocked')),  
  progress DECIMAL(5,2) DEFAULT 0 CHECK (progress >= 0 AND progress <= 100),  
  start_date DATE,  
  end_date DATE,  
  estimated_hours INTEGER,  
  actual_hours INTEGER DEFAULT 0,  
  assigned_team VARCHAR(255),  
  supervisor VARCHAR(255),  
  supervisor_phone VARCHAR(20),  
  notes TEXT,  
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),  
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()  
);
```

-- Tabla de Equipos/Cuadrillas

```
CREATE TABLE IF NOT EXISTS public.teams (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(255) NOT NULL,  
  specialty VARCHAR(100) NOT NULL,  
  supervisor_name VARCHAR(255) NOT NULL,
```

```
supervisor_phone VARCHAR(20),
supervisor_email VARCHAR(255),
status VARCHAR(50) DEFAULT 'active' CHECK (status IN ('active', 'inactive')),
assigned_projects INTEGER[], -- Array de project IDs
created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

-- Tabla de Fotos de Avance

```
CREATE TABLE IF NOT EXISTS public.progress_photos (
    id SERIAL PRIMARY KEY,
    apartment_activity_id INTEGER REFERENCES public.apartment_activities(id) ON
    DELETE CASCADE,
    file_url TEXT NOT NULL,
    file_name VARCHAR(255),
    description TEXT,
    taken_date TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    uploaded_by UUID REFERENCES public.user_profiles(id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

-- Tabla de Problemas/Issues

```
CREATE TABLE IF NOT EXISTS public.activity_issues (
    id SERIAL PRIMARY KEY,
    apartment_activity_id INTEGER REFERENCES public.apartment_activities(id) ON
    DELETE CASCADE,
```



```

type VARCHAR(50) NOT NULL CHECK (type IN ('material', 'dependency', 'quality',
'safety', 'other')),

priority VARCHAR(20) DEFAULT 'media' CHECK (priority IN ('baja', 'media', 'alta',
'critica')),

title VARCHAR(255) NOT NULL,

description TEXT NOT NULL,

status VARCHAR(50) DEFAULT 'open' CHECK (status IN ('open', 'in-progress',
'resolved', 'closed')),

reported_by UUID REFERENCES public.user_profiles(id),

assigned_to UUID REFERENCES public.user_profiles(id),

resolved_at TIMESTAMP WITH TIME ZONE,

created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

```

-- Insertar actividades base (template)

```

INSERT INTO public.activity_templates (name, category, estimated_hours, sort_order,
dependencies) VALUES

('Tabiquería', 'Estructura', 24, 1, '{}'),

('Instalación Eléctrica', 'Instalaciones', 16, 2, '{1}'),

('Instalación Sanitaria', 'Instalaciones', 20, 3, '{1}'),

('Estuco/Yeso', 'Acabados', 32, 4, '{2,3}'),

('Pintura', 'Acabados', 24, 5, '{4}'),

('Piso Flotante', 'Pisos', 16, 6, '{5}'),

('Puertas Interiores', 'Carpintería', 8, 7, '{6}'),

('Accesorios y Grifería', 'Terminaciones', 6, 8, '{7}');

```

-- Insertar equipos de ejemplo

```
INSERT INTO public.teams (name, specialty, supervisor_name, supervisor_phone, supervisor_email) VALUES
```

```
('Equipo Albañilería A', 'Tabiquería', 'Carlos Mendoza', '+56912345678', 'carlos@constructora.com'),
```

```
('Equipo Eléctrico A', 'Instalaciones Eléctricas', 'José Ramírez', '+56912345679', 'jose@constructora.com'),
```

```
('Equipo Gasfitería A', 'Instalaciones Sanitarias', 'Miguel Torres', '+56912345680', 'miguel@constructora.com'),
```

```
('Equipo Estuco A', 'Estuco y Yeso', 'Antonio Silva', '+56912345681', 'antonio@constructora.com'),
```

```
('Equipo Pintura A', 'Pintura', 'Luis González', '+56912345682', 'luis@constructora.com'),
```

```
('Equipo Pisos A', 'Pisos Flotantes', 'Roberto Vega', '+56912345683', 'roberto@constructora.com'),
```

```
('Equipo Carpintería A', 'Carpintería', 'Fernando Rojas', '+56912345684', 'fernando@constructora.com'),
```

```
('Equipo Terminaciones A', 'Terminaciones', 'Carmen López', '+56912345685', 'carmen@constructora.com');
```

-- Habilitar RLS en todas las tablas

```
ALTER TABLE public.projects ENABLE ROW LEVEL SECURITY;
```

```
ALTER TABLE public.floors ENABLE ROW LEVEL SECURITY;
```

```
ALTER TABLE public.apartments ENABLE ROW LEVEL SECURITY;
```

```
ALTER TABLE public.activity_templates ENABLE ROW LEVEL SECURITY;
```

```
ALTER TABLE public.apartment_activities ENABLE ROW LEVEL SECURITY;
```

```
ALTER TABLE public.teams ENABLE ROW LEVEL SECURITY;
```

```
ALTER TABLE public.progress_photos ENABLE ROW LEVEL SECURITY;
```

```
ALTER TABLE public.activity_issues ENABLE ROW LEVEL SECURITY;
```

-- Políticas básicas (todos pueden leer, admin puede modificar todo)

-- Proyectos

```
CREATE POLICY "Anyone can view projects" ON public.projects FOR SELECT USING (true);
```

```
CREATE POLICY "Admin can manage projects" ON public.projects FOR ALL USING (  
    EXISTS (SELECT 1 FROM public.user_profiles WHERE id = auth.uid() AND role =  
    'admin')  
);
```

-- Pisos

```
CREATE POLICY "Anyone can view floors" ON public.floors FOR SELECT USING (true);
```

```
CREATE POLICY "Admin can manage floors" ON public.floors FOR ALL USING (  
    EXISTS (SELECT 1 FROM public.user_profiles WHERE id = auth.uid() AND role IN  
    ('admin', 'supervisor'))  
);
```

-- Departamentos

```
CREATE POLICY "Anyone can view apartments" ON public.apartments FOR SELECT  
USING (true);
```

```
CREATE POLICY "Admin can manage apartments" ON public.apartments FOR ALL  
USING (  
    EXISTS (SELECT 1 FROM public.user_profiles WHERE id = auth.uid() AND role IN  
    ('admin', 'supervisor'))  
);
```

-- Actividades

```
CREATE POLICY "Anyone can view activity templates" ON public.activity_templates
FOR SELECT USING (true);
```

```
CREATE POLICY "Anyone can view apartment activities" ON
public.apartment_activities FOR SELECT USING (true);
```

```
CREATE POLICY "Users can update activities" ON public.apartment_activities FOR
UPDATE USING (true);
```

-- Equipos

```
CREATE POLICY "Anyone can view teams" ON public.teams FOR SELECT USING
(true);
```

```
CREATE POLICY "Admin can manage teams" ON public.teams FOR ALL USING (
  EXISTS (SELECT 1 FROM public.user_profiles WHERE id = auth.uid() AND role IN
    ('admin', 'supervisor'))
);
```

-- Fotos y issues

```
CREATE POLICY "Anyone can view photos" ON public.progress_photos FOR SELECT
USING (true);
```

```
CREATE POLICY "Users can upload photos" ON public.progress_photos FOR INSERT
WITH CHECK (auth.uid() = uploaded_by);
```

```
CREATE POLICY "Anyone can view issues" ON public.activity_issues FOR SELECT
USING (true);
```

```
CREATE POLICY "Users can create issues" ON public.activity_issues FOR INSERT
WITH CHECK (auth.uid() = reported_by);
```

Paso 3.2: Crear Servicios de API

Archivo: src/lib/api/projects.ts

```
import { supabase } from '@lib/supabase'

import type { Database } from '@lib/supabase'
```

```
type Project = Database['public']['Tables']['projects']['Row']
```

```
type ProjectInsert = Database['public']['Tables']['projects']['Insert']
```

```
type ProjectUpdate = Database['public']['Tables']['projects']['Update']
```

```
export class ProjectsService {
```

```
  static async getAllProjects() {
```

```
    const { data, error } = await supabase
```

```
      .from('projects')
```

```
      .select('*')
```

```
      .order('created_at', { ascending: false })
```

```
    if (error) throw error
```

```
    return data
```

```
  }
```

```
  static async getProjectById(id: number) {
```

```
    const { data, error } = await supabase
```

```
      .from('projects')
```

```
      .select(`
```

```
        *,
```

```
        floors (
```

```
          *,
```

```
          apartments (
```

```
            *,
```

```
            apartment_activities (
```

```
              *,
```

```
        activity_templates (*)
    )
)
)
`)
.eq('id', id)
.single()
```

```
if (error) throw error
return data
}
```

```
static async createProject(project: ProjectInsert) {
    const { data, error } = await supabase
        .from('projects')
        .insert(project)
        .select()
        .single()
```

```
if (error) throw error
return data
}
```

```
static async updateProject(id: number, updates: ProjectUpdate) {
    const { data, error } = await supabase
        .from('projects')
```

```
.update({  
  ...updates,  
  updated_at: new Date().toISOString()  
})  
.eq('id', id)  
.select()  
.single()
```

```
if (error) throw error  
return data  
}
```

```
static async deleteProject(id: number) {  
  const { error } = await supabase  
    .from('projects')  
    .delete()  
    .eq('id', id)
```

```
  if (error) throw error  
}
```

```
static async createProjectStructure(projectId: number, totalFloors: number,  
unitsPerFloor: number) {  
  // Crear pisos  
  const floors = Array.from({ length: totalFloors }, (_, i) => ({  
    project_id: projectId,
```

```
    floor_number: totalFloors - i, // Del más alto al más bajo
    status: 'pending' as const
  })
})
```

```
const { data: floorsData, error: floorsError } = await supabase
  .from('floors')
  .insert(floors)
  .select()
```

```
if (floorsError) throw floorsError
```

```
// Crear departamentos para cada piso
```

```
const apartments = []
```

```
for (const floor of floorsData) {
```

```
  for (let i = 1; i <= unitsPerFloor; i++) {
```

```
    apartments.push({
```

```
      floor_id: floor.id,
```

```
      apartment_number: `${floor.floor_number}${i.toString().padStart(2, '0')}` ,
```

```
      apartment_type: i <= 2 ? '3D+2B' : '2D+1B', // Ejemplo de tipos
```

```
      area: i <= 2 ? 89.5 : 65.2,
```

```
      status: 'pending' as const
```

```
    })
```

```
  }
```

```
}
```

```
const { data: apartmentsData, error: apartmentsError } = await supabase
```



```
.from('apartments')  
.insert(apartments)  
.select()
```

```
if (apartmentsError) throw apartmentsError
```

```
// Crear actividades para cada departamento
```

```
const { data: templates, error: templatesError } = await supabase
```

```
.from('activity_templates')  
.select('*')  
.eq('is_active', true)  
.order('sort_order')
```

```
if (templatesError) throw templatesError
```

```
const activities = []  
for (const apartment of apartmentsData) {  
  for (const template of templates) {  
    activities.push({  
      apartment_id: apartment.id,  
      activity_template_id: template.id,  
      estimated_hours: template.estimated_hours,  
      status: 'pending' as const,  
      progress: 0  
    })  
  }  
}
```

```

    }

    const { error: activitiesError } = await supabase
      .from('apartment_activities')
      .insert(activities)

    if (activitiesError) throw activitiesError

    return { floors: floorsData, apartments: apartmentsData }
  }
}

```

Archivo: src/lib/api/teams.ts

```

import { supabase } from '@lib/supabase'
import type { Database } from '@lib/supabase'

type Team = Database['public']['Tables']['teams']['Row']
type TeamInsert = Database['public']['Tables']['teams']['Insert']
type TeamUpdate = Database['public']['Tables']['teams']['Update']

export class TeamsService {
  static async getAllTeams() {
    const { data, error } = await supabase
      .from('teams')
      .select('*')
      .order('name')
  }
}

```

```
    if (error) throw error
    return data
  }
```

```
static async getTeamById(id: number) {
  const { data, error } = await supabase
    .from('teams')
    .select('*')
    .eq('id', id)
    .single()
```

```
    if (error) throw error
    return data
  }
```

```
static async createTeam(team: TeamInsert) {
  const { data, error } = await supabase
    .from('teams')
    .insert(team)
    .select()
    .single()
```

```
    if (error) throw error
    return data
  }
```

```
static async updateTeam(id: number, updates: TeamUpdate) {  
  const { data, error } = await supabase  
    .from('teams')  
    .update({  
      ...updates,  
      updated_at: new Date().toISOString()  
    })  
    .eq('id', id)  
    .select()  
    .single()  
  
  if (error) throw error  
  return data  
}
```

```
static async deleteTeam(id: number) {  
  const { error } = await supabase  
    .from('teams')  
    .delete()  
    .eq('id', id)  
  
  if (error) throw error  
}
```

```
static async assignTeamToProjects(teamId: number, projectIds: number[]) {  
  const { data, error } = await supabase
```

```

        .from('teams')

        .update({
            assigned_projects: projectIds,
            updated_at: new Date().toISOString()
        })

        .eq('id', teamId)

        .select()

        .single()

        if (error) throw error

        return data
    }
}

```

Archivo: src/lib/api/activities.ts

```

import { supabase } from '@lib/supabase'

import type { Database } from '@lib/supabase'

type ApartmentActivity = Database['public']['Tables']['apartment_activities']['Row']

type ApartmentActivityUpdate =
    Database['public']['Tables']['apartment_activities']['Update']

type ActivityTemplate = Database['public']['Tables']['activity_templates']['Row']

export class ActivitiesService {

    static async getActivityTemplates() {

        const { data, error } = await supabase

            .from('activity_templates')

```

```
.select('*')  
.eq('is_active', true)  
.order('sort_order')
```

```
if (error) throw error  
return data  
}
```

```
static async getApartmentActivities(apartmentId: number) {  
  const { data, error } = await supabase  
    .from('apartment_activities')  
    .select(`  
      *,  
      activity_templates (*)  
    `)  
    .eq('apartment_id', apartmentId)  
    .order('id')
```

```
if (error) throw error  
return data  
}
```

```
static async updateActivity(activityId: number, updates: ApartmentActivityUpdate) {  
  const { data, error } = await supabase  
    .from('apartment_activities')  
    .update({
```

```

        ...updates,
        updated_at: new Date().toISOString()
    })
    .eq('id', activityId)
    .select(`
        *,
        activity_templates (*)
    `)
    .single()

    if (error) throw error
    return data
}

```

```

static async completeActivity(activityId: number, actualHours?: number) {
    const { data, error } = await supabase
        .from('apartment_activities')
        .update({
            status: 'completed',
            progress: 100,
            end_date: new Date().toISOString().split('T')[0],
            actual_hours: actualHours,
            updated_at: new Date().toISOString()
        })
        .eq('id', activityId)
        .select(`

```

```
*,
activity_templates (*)
`)
.single()
```

```
if (error) throw error
```

```
return data
```

```
}
```

```
static async startActivity(activityId: number, assignedTeam?: string, supervisor?:
string) {
```

```
  const { data, error } = await supabase
```

```
    .from('apartment_activities')
```

```
    .update({
```

```
      status: 'in-progress',
```

```
      start_date: new Date().toISOString().split('T')[0],
```

```
      assigned_team: assignedTeam,
```

```
      supervisor: supervisor,
```

```
      updated_at: new Date().toISOString()
```

```
    })
```

```
    .eq('id', activityId)
```

```
    .select(`
```

```
      *,
```

```
      activity_templates (*)
```

```
    `)
```

```
    .single()
```



```
if (error) throw error  
return data  
}
```

```
static async blockActivity(activityId: number, reason: string) {  
  const { data, error } = await supabase  
    .from('apartment_activities')  
    .update({  
      status: 'blocked',  
      notes: reason,  
      updated_at: new Date().toISOString()  
    })  
    .eq('id', activityId)  
    .select(`  
      *,  
      activity_templates (*)  
    `)  
    .single()  
}
```

```
if (error) throw error  
return data  
}  
}
```

Paso 3.3: Crear Formularios con Validación

Instalar dependencias adicionales:

```
npm install react-hook-form@7.52.1 zod@3.23.8 @hookform/resolvers@3.3.4
```

Archivo: src/lib/validations.ts

```
import { z } from 'zod'
```

```
export const projectSchema = z.object({  
  name: z.string()  
    .min(3, 'El nombre debe tener al menos 3 caracteres')  
    .max(255, 'El nombre es demasiado largo'),  
  address: z.string()  
    .min(5, 'La dirección debe tener al menos 5 caracteres')  
    .optional(),  
  total_floors: z.number()  
    .min(1, 'Debe tener al menos 1 piso')  
    .max(50, 'Máximo 50 pisos'),  
  units_per_floor: z.number()  
    .min(1, 'Debe tener al menos 1 unidad por piso')  
    .max(20, 'Máximo 20 unidades por piso'),  
  start_date: z.string().optional(),  
  estimated_completion: z.string().optional(),  
  status: z.enum(['planning', 'active', 'completed', 'paused'])  
})
```

```
export const teamSchema = z.object({  
  name: z.string()  
    .min(3, 'El nombre debe tener al menos 3 caracteres')  
    .max(255, 'El nombre es demasiado largo'),
```

```
specialty: z.string()
    .min(3, 'La especialidad debe tener al menos 3 caracteres')
    .max(100, 'La especialidad es demasiado larga'),
supervisor_name: z.string()
    .min(3, 'El nombre del supervisor debe tener al menos 3 caracteres')
    .max(255, 'El nombre es demasiado largo'),
supervisor_phone: z.string()
    .min(8, 'El teléfono debe tener al menos 8 dígitos')
    .max(20, 'El teléfono es demasiado largo')
    .optional(),
supervisor_email: z.string()
    .email('Email inválido')
    .optional(),
status: z.enum(['active', 'inactive']),
assigned_projects: z.array(z.number()).optional()
})
```

```
export const activityUpdateSchema = z.object({
  status: z.enum(['pending', 'in-progress', 'completed', 'blocked']).optional(),
  progress: z.number()
    .min(0, 'El progreso no puede ser menor a 0')
    .max(100, 'El progreso no puede ser mayor a 100')
    .optional(),
  start_date: z.string().optional(),
  end_date: z.string().optional(),
  actual_hours: z.number()
})
```

```

        .min(0, 'Las horas no pueden ser menores a 0')
        .optional(),
    assigned_team: z.string().optional(),
    supervisor: z.string().optional(),
    supervisor_phone: z.string().optional(),
    notes: z.string().optional()
  })

export type ProjectFormData = z.infer<typeof projectSchema>
export type TeamFormData = z.infer<typeof teamSchema>
export type ActivityUpdateData = z.infer<typeof activityUpdateSchema>

```

Paso 3.4: Crear Componentes de Formularios

Archivo: src/components/ui/Form.tsx

```

'use client'

import { cn } from '@lib/Utils'
import { ReactNode, HTMLAttributes } from 'react'

export function FormField({
  children,
  className,
  ...props
}: HTMLAttributes<HTMLDivElement>) {
  return (
    <div className={cn('space-y-2', className)} {...props}>
      {children}
    </div>
  )
}

```

```
    </div>
  )
}
```

```
export function FormLabel({
  children,
  className,
  required,
  ...props
}: HTMLAttributes<HTMLLabelElement> & { required?: boolean }) {
  return (
    <label
      className={cn('block text-sm font-medium text-gray-700', className)}
      {...props}
    >
      {children}
      {required && <span className="text-red-500 ml-1">*</span>}
    </label>
  )
}
```

```
export function FormInput({
  className,
  error,
  ...props
}: React.InputHTMLAttributes<HTMLInputElement> & { error?: string }) {
```

```

return (
  <div>
    <input
      className={cn(
        'w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-
blue-500 focus:border-transparent transition-colors',
        error ? 'border-red-500' : 'border-gray-300',
        className
      )}
      {...props}
    />
    {error && (
      <p className="mt-1 text-sm text-red-600">{error}</p>
    )}
  </div>
)
}

```

```

export function FormSelect({
  children,
  className,
  error,
  ...props
}: React.SelectHTMLAttributes<HTMLSelectElement> & { error?: string }) {
  return (
    <div>

```

```

    <select
      className={cn(
        'w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-
blue-500 focus:border-transparent transition-colors',
        error ? 'border-red-500' : 'border-gray-300',
        className
      )}
      {...props}
    >
      {children}
    </select>
    {error && (
      <p className="mt-1 text-sm text-red-600">{error}</p>
    )}
  </div>
)
}

```

```

export function FormTextarea({
  className,
  error,
  ...props
}: React.TextareaHTMLAttributes<HTMLTextAreaElement> & { error?: string }) {
  return (
    <div>
      <textarea

```

```

        className={cn(
          'w-full px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-
blue-500 focus:border-transparent transition-colors resize-vertical',
          error ? 'border-red-500' : 'border-gray-300',
          className
        )}
        rows={3}
        {...props}
      />
      {error && (
        <p className="mt-1 text-sm text-red-600">{error}</p>
      )}
    </div>
  )
}

```

```

export function FormError({ message }: { message?: string }) {
  if (!message) return null
  return (
    <div className="bg-red-50 border border-red-200 rounded-lg p-4">
      <p className="text-sm text-red-600">❌ {message}</p>
    </div>
  )
}

```

```

export function FormSuccess({ message }: { message?: string }) {

```



```

if (!message) return null

return (

  <div className="bg-green-50 border border-green-200 rounded-lg p-4">

    <p className="text-sm text-green-600">✅ {message}</p>

  </div>

)

}

```

Paso 3.5: Crear Página de Gestión de Proyectos

Archivo: src/app/(auth)/proyectos/page.tsx

```

'use client'

import { useState, useEffect } from 'react'
import { useAuth } from '@/contexts/AuthContext'
import { ProtectedRoute } from '@/components/auth/ProtectedRoute'
import { Card, CardHeader, CardTitle, CardContent } from '@/components/ui/Card'
import { Button } from '@/components/ui/Button'
import { Plus, Building, Calendar, Users, Edit, Trash2, Eye } from 'lucide-react'
import { ProjectsService } from '@/lib/api/projects'
import { formatDate } from '@/lib/utills'
import ProjectForm from '@/components/projects/ProjectForm'
import toast from 'react-hot-toast'

export default function ProjectsPage() {
  return (
    <ProtectedRoute requiredRoles={['admin', 'supervisor']}>
      <ProjectsPageContent />
    </ProtectedRoute>
  )
}

```

```
    </ProtectedRoute>
  )
}
```

```
function ProjectsPageContent() {
  const [projects, setProjects] = useState<any[]>([])
  const [loading, setLoading] = useState(true)
  const [showForm, setShowForm] = useState(false)
  const [selectedProject, setSelectedProject] = useState<any>(null)
  const { hasRole } = useAuth()

  useEffect(() => {
    loadProjects()
  }, [])

  const loadProjects = async () => {
    try {
      const data = await ProjectsService.getAllProjects()
      setProjects(data)
    } catch (error: any) {
      toast.error('Error al cargar proyectos: ' + error.message)
    } finally {
      setLoading(false)
    }
  }
}
```

```
const handleCreateProject = () => {  
  setSelectedProject(null)  
  setShowForm(true)  
}
```

```
const handleEditProject = (project: any) => {  
  setSelectedProject(project)  
  setShowForm(true)  
}
```

```
const handleDeleteProject = async (project: any) => {  
  if (!confirm(`¿Estás seguro de eliminar el proyecto "${project.name}"?`)) {  
    return  
  }
```

```
  try {  
    await ProjectsService.deleteProject(project.id)  
    toast.success('Proyecto eliminado correctamente')  
    loadProjects()  
  } catch (error: any) {  
    toast.error('Error al eliminar proyecto: ' + error.message)  
  }  
}
```

```
const handleFormSubmit = () => {  
  setShowForm(false)
```

```
setSelectedProject(null)
loadProjects()
}
```

```
const getStatusColor = (status: string) => {
  switch (status) {
    case 'active': return 'bg-green-100 text-green-800'
    case 'completed': return 'bg-blue-100 text-blue-800'
    case 'paused': return 'bg-yellow-100 text-yellow-800'
    case 'planning': return 'bg-gray-100 text-gray-800'
    default: return 'bg-gray-100 text-gray-800'
  }
}
```

```
const getStatusText = (status: string) => {
  switch (status) {
    case 'active': return 'Activo'
    case 'completed': return 'Completado'
    case 'paused': return 'Pausado'
    case 'planning': return 'Planificación'
    default: return status
  }
}
```

```
if (loading) {
  return (
```

```

<div className="p-6">

  <Card>

    <CardContent className="p-8 text-center">

      <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-blue-600 mx-auto mb-4"></div>

      <p className="text-gray-600">Cargando proyectos...</p>

    </CardContent>

  </Card>

</div>

)
}

```

```

if (showForm) {
  return (
    <ProjectForm
      project={selectedProject}
      onSubmit={handleFormSubmit}
      onCancel={() => {
        setShowForm(false)
        setSelectedProject(null)
      }}
    />
  )
}

```

```

return (

```

```

<div className="p-6 space-y-6">

  {/* Header */}

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

    <div>

      <h1 className="text-3xl font-bold text-gray-800">Gestión de Proyectos</h1>

      <p className="text-gray-600 mt-1">Administra todos los proyectos de
terminaciones</p>

    </div>

    {hasRole(['admin']) && (

      <Button onClick={handleCreateProject} className="flex items-center gap-2">

        <Plus size={20} />

        Nuevo Proyecto

      </Button>

    )}

  </div>

  {/* Lista de proyectos */}

  <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

    {projects.map((project) => (

      <Card key={project.id} className="hover:shadow-lg transition-shadow">

        <CardHeader>

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

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

              <Building className="text-blue-600" size={24} />

            <div>

              <CardTitle className="text-lg">{project.name}</CardTitle>


```

```
    <span className={` inline-block px-2 py-1 rounded-full text-xs font-medium
    ${getStatusColor(project.status)} ` }>
```

```
      {getStatusText(project.status)}
```

```
    </span>
```

```
  </div>
```

```
</div>
```

```
</div>
```

```
</CardHeader>
```

```
<CardContent>
```

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

```
    <div className="text-sm text-gray-600">
```

```
      <p className="flex items-center gap-2">
```

```
        📍 {project.address || 'Sin dirección'}
```

```
      </p>
```

```
    </div>
```

```
  <div className="grid grid-cols-2 gap-4 text-sm">
```

```
    <div>
```

```
      <span className="text-gray-500">Pisos:</span>
```

```
      <div className="font-semibold">{project.total_floors}</div>
```

```
    </div>
```

```
    <div>
```

```
      <span className="text-gray-500">Unidades/Piso:</span>
```

```
      <div className="font-semibold">{project.units_per_floor}</div>
```

```
    </div>
```

```

<div>

  <span className="text-gray-500">Total Unidades:</span>

  <div className="font-semibold text-blue-600">

    {project.total_floors * project.units_per_floor}

  </div>

</div>

<div>

  <span className="text-gray-500">Creado:</span>

  <div className="font-semibold">{formatDate(project.created_at)}</div>

</div>

</div>

{project.estimated_completion && (

  <div className="flex items-center gap-2 text-sm text-gray-600">

    <Calendar size={16} />

    <span>Entrega estimada:

{formatDate(project.estimated_completion)}</span>

  </div>

  )}

</div>

<div className="flex gap-2 mt-4 pt-4 border-t">

  <Button variant="outline" size="sm" className="flex-1">

    <Eye size={16} className="mr-1" />

    Ver

  </Button>

```



```

{hasRole(['admin']) && (
  <>
    <Button
      variant="secondary"
      size="sm"
      onClick={() => handleEditProject(project)}
    >
      <Edit size={16} />
    </Button>
    <Button
      variant="danger"
      size="sm"
      onClick={() => handleDeleteProject(project)}
    >
      <Trash2 size={16} />
    </Button>
  </>
)}
</div>
</CardContent>
</Card>
))}
</div>

{projects.length === 0 && (
  <Card>

```

```

<CardContent className="p-12 text-center">
  <Building className="mx-auto mb-4 text-gray-400" size={64} />
  <h3 className="text-lg font-semibold text-gray-600 mb-2">
    No hay proyectos registrados
  </h3>
  <p className="text-gray-500 mb-6">
    Crea tu primer proyecto para comenzar a gestionar las terminaciones
  </p>
  {hasRole(['admin']) && (
    <Button onClick={handleCreateProject}>
      <Plus size={20} className="mr-2" />
      Crear Primer Proyecto
    </Button>
  )}
</CardContent>
</Card>
)}
</div>
)
}

```

Archivo: src/components/projects/ProjectForm.tsx

```
'use client'
```

```
import { useState } from 'react'
```

```
import { useForm } from 'react-hook-form'
```

```
import { zodResolver } from '@hookform/resolvers/zod'
```

```
import { Card, CardHeader, CardTitle, CardContent } from '@components/ui/Card'

import { Button } from '@components/ui/Button'

import { FormField, FormLabel, FormInput, FormSelect, FormError, FormSuccess }
from '@components/ui/Form'

import { ArrowLeft, Save, Loader2 } from 'lucide-react'

import { ProjectsService } from '@lib/api/projects'

import { projectSchema, type ProjectFormData } from '@lib/validations'

import toast from 'react-hot-toast'
```

```
interface ProjectFormProps {

  project?: any

  onSubmit: () => void

  onCancel: () => void

}
```

```
export default function ProjectForm({ project, onSubmit, onCancel }:
ProjectFormProps) {

  const [loading, setLoading] = useState(false)

  const [error, setError] = useState("")

  const [success, setSuccess] = useState("")

  const isEditing = !!project

  const {
    register,
    handleSubmit,
    formState: { errors },
```

watch

```
} = useForm<ProjectFormData>({
  resolver: zodResolver(projectSchema),
  defaultValues: {
    name: project?.name || '',
    address: project?.address || '',
    total_floors: project?.total_floors || 1,
    units_per_floor: project?.units_per_floor || 1,
    start_date: project?.start_date || '',
    estimated_completion: project?.estimated_completion || '',
    status: project?.status || 'planning'
  }
})
```

```
const totalFloors = watch('total_floors')
const unitsPerFloor = watch('units_per_floor')
const totalUnits = totalFloors * unitsPerFloor
```

```
const onSubmitForm = async (data: ProjectFormData) => {
  setLoading(true)
  setError('')
  setSuccess('')

  try {
    if (isEditing) {
      await ProjectsService.updateProject(project.id, data)
```

```
toast.success('Proyecto actualizado correctamente')
} else {
  const newProject = await ProjectsService.createProject(data)
  toast.success('Proyecto creado correctamente')

  // Crear estructura automáticamente si no está en planificación
  if (data.status !== 'planning') {
    await ProjectsService.createProjectStructure(
      newProject.id,
      data.total_floors,
      data.units_per_floor
    )
    toast.success('Estructura del proyecto creada')
  }
}

setSuccess(isEditing ? 'Proyecto actualizado' : 'Proyecto creado')
setTimeout(() => {
  onSubmit()
}, 1000)

} catch (error: any) {
  setError(error.message || 'Error inesperado')
  toast.error('Error al guardar proyecto')
} finally {
  setLoading(false)
```

```
}  
}
```

```
return (  
  <div className="p-6 max-w-2xl mx-auto">  
    <Card>  
      <CardHeader>  
        <div className="flex items-center gap-4">  
          <Button  
            variant="outline"  
            size="sm"  
            onClick={onCancel}  
            className="flex items-center gap-2"  
          >  
            <ArrowLeft size={16} />  
            Volver  
          </Button>  
          <CardTitle>  
            {isEditing ? 'Editar Proyecto' : 'Nuevo Proyecto'}  
          </CardTitle>  
        </div>  
      </CardHeader>  
  
      <CardContent>  
        <form onSubmit={handleSubmit(onSubmitForm)} className="space-y-6">  
          {error && <FormError message={error} />}
```

```
{success && <FormSuccess message={success} />}
```

```
<FormField>
```

```
<FormLabel required>Nombre del Proyecto</FormLabel>
```

```
<FormInput
```

```
{...register('name')}
```

```
placeholder="Ej: Edificio Vista Hermosa"
```

```
error={errors.name?.message}
```

```
disabled={loading}
```

```
/>
```

```
</FormField>
```

```
<FormField>
```

```
<FormLabel>Dirección</FormLabel>
```

```
<FormInput
```

```
{...register('address')}
```

```
placeholder="Ej: Av. Los Robles 2580, Las Condes"
```

```
error={errors.address?.message}
```

```
disabled={loading}
```

```
/>
```

```
</FormField>
```

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

```
<FormField>
```

```
<FormLabel required>Total de Pisos</FormLabel>
```

```
<FormInput
```

```
    type="number"
    min="1"
    max="20"
    {...register('units_per_floor', { valueAsNumber: true })}
    error={errors.units_per_floor?.message}
    disabled={loading}
  />
</FormField>
</div>
```

```
{/* Resumen automático */}

<div className="bg-blue-50 border border-blue-200 rounded-lg p-4">

  <h4 className="font-semibold text-blue-800 mb-2"><img alt="Bar chart icon" data-bbox="648 471 671 488"/> Resumen del
  Proyecto</h4>

  <div className="grid grid-cols-2 md:grid-cols-3 gap-4 text-sm">

    <div>

      <span className="text-blue-600">Total Pisos:</span>

      <div className="font-bold text-blue-800">{totalFloors}</div>

    </div>

    <div>

      <span className="text-blue-600">Unidades por Piso:</span>

      <div className="font-bold text-blue-800">{unitsPerFloor}</div>

    </div>

    <div>

      <span className="text-blue-600">Total Unidades:</span>

      <div className="font-bold text-blue-800 text-lg">{totalUnits}</div>
```



```
</div>
```

```
</div>
```

```
</div>
```

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

```
<FormField>
```

```
<FormLabel>Fecha de Inicio</FormLabel>
```

```
<FormInput
```

```
  type="date"
```

```
  {...register('start_date')}
```

```
  error={errors.start_date?.message}
```

```
  disabled={loading}
```

```
</FormField>
```

```
<FormField>
```

```
<FormLabel>Fecha Estimada de Entrega</FormLabel>
```

```
<FormInput
```

```
  type="date"
```

```
  {...register('estimated_completion')}
```

```
  error={errors.estimated_completion?.message}
```

```
  disabled={loading}
```

```
</FormField>
```

```
</div>
```

```

<FormField>

  <FormLabel required>Estado del Proyecto</FormLabel>

  <FormSelect
    {...register('status')}
    error={errors.status?.message}
    disabled={loading}
  >

    <option value="planning">Planificación</option>
    <option value="active">Activo</option>
    <option value="paused">Pausado</option>
    <option value="completed">Completado</option>
  </FormSelect>

  <p className="text-xs text-gray-500 mt-1">

```

💡 Si eliges "Activo", se creará automáticamente la estructura completa del proyecto

```

  </p>

</FormField>

```

```

<div className="flex gap-4 pt-6 border-t">

  <Button
    type="button"
    variant="outline"
    onClick={onCancel}
    disabled={loading}
    className="flex-1"
  >

```

Cancelar

</Button>

<Button

type="submit"

disabled={loading}

className="flex-1 flex items-center gap-2"

>

{loading ? (

<>

<Loader2 className="animate-spin" size={16} />

{isEditing ? 'Actualizando...' : 'Creando...'}

</>

): (

<>

<Save size={16} />

{isEditing ? 'Actualizar' : 'Crear Proyecto'}

</>

)}

</Button>

</div>

</form>

</CardContent>

</Card>

</div>

)

```
}
```

Paso 3.6: Crear Página de Gestión de Equipos

Archivo: src/app/(auth)/equipos/page.tsx

```
'use client'
```

```
import { useState, useEffect } from 'react'
```

```
import { useAuth } from '@/contexts/AuthContext'
```

```
import { ProtectedRoute } from '@/components/auth/ProtectedRoute'
```

```
import { Card, CardHeader, CardTitle, CardContent } from '@/components/ui/Card'
```

```
import { Button } from '@/components/ui/Button'
```

```
import { Plus, Users, Phone, Mail, Edit, Trash2, Building } from 'lucide-react'
```

```
import { TeamsService } from '@/lib/api/teams'
```

```
import { ProjectsService } from '@/lib/api/projects'
```

```
import TeamForm from '@/components/teams/TeamForm'
```

```
import toast from 'react-hot-toast'
```

```
export default function TeamsPage() {
```

```
  return (
```

```
    <ProtectedRoute requiredRoles={['admin', 'supervisor']}>
```

```
      <TeamsPageContent />
```

```
    </ProtectedRoute>
```

```
  )
```

```
}
```

```
function TeamsPageContent() {
```

```
  const [teams, setTeams] = useState<any[]>([])
```

```
const [projects, setProjects] = useState<any[]>([])
const [loading, setLoading] = useState(true)
const [showForm, setShowForm] = useState(false)
const [selectedTeam, setSelectedTeam] = useState<any>(null)
const { hasRole } = useAuth()
```

```
useEffect(() => {
  loadData()
}, [])
```

```
const loadData = async () => {
  try {
    const [teamsData, projectsData] = await Promise.all([
      TeamsService.getAllTeams(),
      ProjectsService.getAllProjects()
    ])
    setTeams(teamsData)
    setProjects(projectsData)
  } catch (error: any) {
    toast.error('Error al cargar datos: ' + error.message)
  } finally {
    setLoading(false)
  }
}
```

```
const handleCreateTeam = () => {
```

```
setSelectedTeam(null)
setShowForm(true)
}
```

```
const handleEditTeam = (team: any) => {
  setSelectedTeam(team)
  setShowForm(true)
}
```

```
const handleDeleteTeam = async (team: any) => {
  if (!confirm(`¿Estás seguro de eliminar el equipo "${team.name}"?`)) {
    return
  }
}
```

```
try {
  await TeamsService.deleteTeam(team.id)
  toast.success('Equipo eliminado correctamente')
  loadData()
} catch (error: any) {
  toast.error('Error al eliminar equipo: ' + error.message)
}
}
```

```
const handleFormSubmit = () => {
  setShowForm(false)
  setSelectedTeam(null)
}
```

```
loadData()
```

```
}
```

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

```
  return status === 'active'
```

```
    ? 'bg-green-100 text-green-800'
```

```
    : 'bg-gray-100 text-gray-800'
```

```
}
```

```
const getAssignedProjects = (assignedProjectIds: number[] = []) => {
```

```
  return projects.filter(project => assignedProjectIds.includes(project.id))
```

```
}
```

```
if (loading) {
```

```
  return (
```

```
    <div className="p-6">
```

```
      <Card>
```

```
        <CardContent className="p-8 text-center">
```

```
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-blue-600 mx-auto mb-4"></div>
```

```
          <p className="text-gray-600">Cargando equipos...</p>
```

```
        </CardContent>
```

```
      </Card>
```

```
    </div>
```

```
  )
```

```
}
```

```

if (showForm) {
  return (
    <TeamForm
      team={selectedTeam}
      projects={projects}
      onSubmit={handleFormSubmit}
      onCancel={() => {
        setShowForm(false)
        setSelectedTeam(null)
      }}
    />
  )
}

```

```

return (
  <div className="p-6 space-y-6">
    {/* Header */}
    <div className="flex justify-between items-center">
      <div>
        <h1 className="text-3xl font-bold text-gray-800">Gestión de Equipos</h1>
        <p className="text-gray-600 mt-1">Administra las cuadrillas y
especialistas</p>
      </div>
      {hasRole(['admin']) && (
        <Button onClick={handleCreateTeam} className="flex items-center gap-2">

```



```

        <Plus size={20} />

        Nuevo Equipo

    </Button>

  })

</div>

{/* Lista de equipos */}

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
  {teams.map((team) => {
    const assignedProjects = getAssignedProjects(team.assigned_projects)

    return (
      <Card key={team.id} className="hover:shadow-lg transition-shadow">
        <CardHeader>
          <div className="flex justify-between items-start">
            <div className="flex items-center gap-3">
              <Users className="text-blue-600" size={24} />
              <div>
                <CardTitle className="text-lg">{team.name}</CardTitle>
                <p className="text-sm text-gray-600">{team.specialty}</p>
              </div>
            </div>
            <span className={` inline-block px-2 py-1 rounded-full text-xs font-medium
${getStatusColor(team.status)} `>
              {team.status === 'active' ? 'Activo' : 'Inactivo'}
            </span>
          </div>
        </CardHeader>
      </Card>
    )
  })}
</div>

```

</div>

</CardHeader>

<CardContent>

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

{ /* Supervisor */ }

<div className="bg-gray-50 rounded-lg p-3">

<h4 className="font-semibold text-sm text-gray-700 mb-2">👤
Supervisor</h4>

<p className="font-medium">{team.supervisor_name}</p>

<div className="flex flex-col gap-1 mt-2 text-sm">

{team.supervisor_phone && (

<div className="flex items-center gap-2 text-gray-600">

<Phone size={14} />

{team.supervisor_phone}

</div>

)}

{team.supervisor_email && (

<div className="flex items-center gap-2 text-gray-600">

<Mail size={14} />

{team.supervisor_email}

</div>

)}

</div>

</div>

```

    { /* Proyectos asignados */ }

    <div>

      <h4 className="font-semibold text-sm text-gray-700 mb-2">

        🏠 Proyectos Asignados ({assignedProjects.length})

      </h4>

      {assignedProjects.length > 0 ? (

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

          {assignedProjects.slice(0, 2).map((project) => (

            <div key={project.id} className="text-xs bg-blue-50 text-blue-700 px-2
py-1 rounded">

              {project.name}

            </div>

          ))}

          {assignedProjects.length > 2 && (

            <div className="text-xs text-gray-500">

              +{assignedProjects.length - 2} más

            </div>

          )}

        </div>

      ) : (

        <p className="text-sm text-gray-500">Sin proyectos asignados</p>

      )}

    </div>

  </div>

  <div className="flex gap-2 mt-4 pt-4 border-t">

```

```

<Button
  variant="outline"
  size="sm"
  className="flex-1"
  disabled={!team.supervisor_phone}
>
  <Phone size={14} className="mr-1" />
  Llamar
</Button>
{hasRole(['admin']) && (
  <>
    <Button
      variant="secondary"
      size="sm"
      onClick={() => handleEditTeam(team)}
    >
      <Edit size={16} />
    </Button>
    <Button
      variant="danger"
      size="sm"
      onClick={() => handleDeleteTeam(team)}
    >
      <Trash2 size={16} />
    </Button>
  </>
)

```

```
    })
  </div>

  </CardContent>
</Card>

)
}}
</div>
```

```
{teams.length === 0 && (
  <Card>
    <CardContent className="p-12 text-center">
      <Users className="mx-auto mb-4 text-gray-400" size={64} />
      <h3 className="text-lg font-semibold text-gray-600 mb-2">
        No hay equipos registrados
      </h3>
      <p className="text-gray-500 mb-6">
        Crea tu primer equipo para comenzar a gestionar las cuadrillas
      </p>
      {hasRole(['admin']) && (
        <Button onClick={handleCreateTeam}>
          <Plus size={20} className="mr-2" />
          Crear Primer Equipo
        </Button>
      )}
    </CardContent>
  </Card>
```


```
}}
```

```
{/* Resumen por especialidades */}
```

```
{teams.length > 0 && (
```

```
<Card>
```

```
<CardHeader>
```

```
<CardTitle> Resumen por Especialidades</CardTitle>
```

```
</CardHeader>
```

```
<CardContent>
```

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

```
{Object.entries(
```

```
teams.reduce((acc: any, team) => {
```

```
acc[team.specialty] = (acc[team.specialty] || 0) + 1
```

```
return acc
```

```
}, {})
```

```
).map(([specialty, count]) => (
```

```
<div key={specialty} className="text-center p-4 bg-gray-50 rounded-lg">
```

```
<div className="text-2xl font-bold text-blue-600">{count}</div>
```

```
<div className="text-sm text-gray-600">{specialty}</div>
```

```
</div>
```

```
)))
```

```
</div>
```

```
</CardContent>
```

```
</Card>
```

```
}}
```

```
</div>
```

```
)  
}
```

Archivo: src/components/teams/TeamForm.tsx

```
'use client'
```

```
import { useState } from 'react'
```

```
import { useForm } from 'react-hook-form'
```

```
import { zodResolver } from '@hookform/resolvers/zod'
```

```
import { Card, CardHeader, CardTitle, CardContent } from '@components/ui/Card'
```

```
import { Button } from '@components/ui/Button'
```

```
import { FormField, FormLabel, FormInput, FormSelect, FormError, FormSuccess }  
from '@components/ui/Form'
```

```
import { ArrowLeft, Save, Loader2 } from 'lucide-react'
```

```
import { TeamsService } from '@lib/api/teams'
```

```
import { teamSchema, type TeamFormData } from '@lib/validations'
```

```
import toast from 'react-hot-toast'
```

```
interface TeamFormProps {
```

```
  team?: any
```

```
  projects: any[]
```

```
  onSubmit: () => void
```

```
  onCancel: () => void
```

```
}
```

```
const SPECIALTIES = [
```

```
  'Tabiquería',
```

```
'Instalaciones Eléctricas',  
'Instalaciones Sanitarias',  
'Estuco y Yeso',  
'Pintura',  
'Pisos Flotantes',  
'Carpintería',  
'Terminaciones',  
'General'  
]
```

```
export default function TeamForm({ team, projects, onSubmit, onCancel }:  
TeamFormProps) {  
  
  const [loading, setLoading] = useState(false)  
  
  const [error, setError] = useState("")  
  
  const [success, setSuccess] = useState("")  
  
  const [selectedProjects, setSelectedProjects] =  
    useState<number[]>(team?.assigned_projects || [])  
  
  
  const isEditing = !!team  
  
  
  const {  
    register,  
    handleSubmit,  
    formState: { errors }  
  } = useForm<TeamFormData>({  
    resolver: zodResolver(teamSchema),  
    defaultValues: {
```



```
name: team?.name || ",  
specialty: team?.specialty || ",  
supervisor_name: team?.supervisor_name || ",  
supervisor_phone: team?.supervisor_phone || ",  
supervisor_email: team?.supervisor_email || ",  
status: team?.status || 'active',  
assigned_projects: team?.assigned_projects || []  
}  
}))
```

```
const onSubmitForm = async (data: TeamFormData) => {  
  setLoading(true)  
  setError("")  
  setSuccess("")  
  
  try {  
    const formData = {  
      ...data,  
      assigned_projects: selectedProjects  
    }  
  
    if (isEditing) {  
      await TeamsService.updateTeam(team.id, formData)  
      toast.success('Equipo actualizado correctamente')  
    } else {  
      await TeamsService.createTeam(formData)
```

```
toast.success('Equipo creado correctamente')
}
```

```
setSuccess(isEditing ? 'Equipo actualizado' : 'Equipo creado')
setTimeout(() => {
  onSubmit()
}, 1000)
```

```
} catch (error: any) {
  setError(error.message || 'Error inesperado')
  toast.error('Error al guardar equipo')
} finally {
  setLoading(false)
}
}
```

```
const handleProjectToggle = (projectId: number) => {
  setSelectedProjects(prev =>
    prev.includes(projectId)
      ? prev.filter(id => id !== projectId)
      : [...prev, projectId]
  )
}
```

```
return (
  <div className="p-6 max-w-2xl mx-auto">
```

```

<Card>

  <CardHeader>

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

      <Button

        variant="outline"

        size="sm"

        onClick={onCancel}

        className="flex items-center gap-2"

      >

        <ArrowLeft size={16} />

        Volver

      </Button>

      <CardTitle>

        {isEditing ? 'Editar Equipo' : 'Nuevo Equipo'}

      </CardTitle>

    </div>

  </CardHeader>

  <CardContent>

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

      {error && <FormError message={error} />}

      {success && <FormSuccess message={success} />}

      <FormField>

        <FormLabel required>Nombre del Equipo</FormLabel>

        <FormInput

```

```
{...register('name')}  
placeholder="Ej: Equipo Pintura A"  
error={errors.name?.message}  
disabled={loading}  
/>  
</FormField>
```

```
<FormField>  
  <FormLabel required>Especialidad</FormLabel>  
  <FormSelect  
    {...register('specialty')}  
    error={errors.specialty?.message}  
    disabled={loading}  
  >  
    <option value="">Seleccionar especialidad...</option>  
    {SPECIALTIES.map((specialty) => (  
      <option key={specialty} value={specialty}>  
        {specialty}  
      </option>  
    ))}  
  </FormSelect>  
</FormField>
```

```
<div className="bg-gray-50 border border-gray-200 rounded-lg p-4">  
  <h4 className="font-semibold text-gray-700 mb-4">👤 Información del  
Supervisor</h4>
```

```
<div className="space-y-4">
  <FormField>
    <FormLabel required>Nombre del Supervisor</FormLabel>
    <FormInput
      {...register('supervisor_name')}
      placeholder="Ej: Carlos Mendoza"
      error={errors.supervisor_name?.message}
      disabled={loading}
    />
  </FormField>
```

```
<div className="grid grid-cols-1 md:grid-cols-2 gap-4">
  <FormField>
    <FormLabel>Teléfono</FormLabel>
    <FormInput
      {...register('supervisor_phone')}
      placeholder="Ej: +56912345678"
      error={errors.supervisor_phone?.message}
      disabled={loading}
    />
  </FormField>
```

```
<FormField>
  <FormLabel>Email</FormLabel>
  <FormInput
```

```

        type="email"
        {...register('supervisor_email')}
        placeholder="supervisor@empresa.com"
        error={errors.supervisor_email?.message}
        disabled={loading}
      />
    </FormField>
  </div>
</div>
</div>

<FormField>
  <FormLabel required>Estado</FormLabel>
  <FormSelect
    {...register('status')}
    error={errors.status?.message}
    disabled={loading}
  >
    <option value="active">Activo</option>
    <option value="inactive">Inactivo</option>
  </FormSelect>
</FormField>

{/* Asignación de proyectos */}
<div>
  <FormLabel>Proyectos Asignados</FormLabel>

```

```

<div className="mt-2 space-y-2 max-h-40 overflow-y-auto border border-
gray-200 rounded-lg p-3">

  {projects.length > 0 ? (

    projects.map((project) => (

      <label key={project.id} className="flex items-center gap-3 cursor-pointer
hover:bg-gray-50 p-2 rounded">

        <input

          type="checkbox"

          checked={selectedProjects.includes(project.id)}

          onChange={() => handleProjectToggle(project.id)}

          className="rounded border-gray-300 text-blue-600 focus:ring-blue-500"

          disabled={loading}

        />

        <div className="flex-1">

          <div className="font-medium text-sm">{project.name}</div>

          <div className="text-xs text-gray-500">{project.address}</div>

        </div>

      </label>

    ))

  ) : (

    <p className="text-sm text-gray-500 text-center py-4">

      No hay proyectos disponibles

    </p>

  )}

</div>

{selectedProjects.length > 0 && (

  <p className="text-xs text-gray-600 mt-2">

```

```
        {selectedProjects.length} proyecto(s) seleccionado(s)
    </p>
  )}
</div>
```

```
<div className="flex gap-4 pt-6 border-t">
```

```
  <Button
    type="button"
    variant="outline"
    onClick={onCancel}
    disabled={loading}
    className="flex-1"
```

```
  >
```

```
    Cancelar
```

```
  </Button>
```

```
  <Button
```

```
    type="submit"
```

```
    disabled={loading}
```

```
    className="flex-1 flex items-center gap-2"
```

```
  >
```

```
    {loading ? (
```

```
      <>
```

```
        <Loader2 className="animate-spin" size={16} />
```

```
        {isEditing ? 'Actualizando...' : 'Creando...'}
      </>
```

```
    </>
```



```

    ) : (
      <>
        <Save size={16} />
        {isEditing ? 'Actualizar' : 'Crear Equipo'}
      </>
    )}
  </Button>
</div>
</form>
</CardContent>
</Card>
</div>
)
}

```

🛠 Paso 3.7: Actualizar Navegación

Archivo: `src/app/(auth)/layout.tsx` (actualizar navigation array)








```

const navigation = [
  { name: 'Dashboard', href: '/', icon: '🏠', roles: ['admin', 'supervisor', 'jefe_cuadrilla', 'maestro'] },
  { name: 'Proyectos', href: '/proyectos', icon: '🏗', roles: ['admin', 'supervisor'] },
  { name: 'Pisos', href: '/pisos', icon: '🏠', roles: ['admin', 'supervisor', 'jefe_cuadrilla'] },
  { name: 'Equipos', href: '/equipos', icon: '👷', roles: ['admin', 'supervisor'] },
  { name: 'Reportes', href: '/reportes', icon: '📊', roles: ['admin'] },
]

```

✅ Criterios de Aceptación - Sprint 3:

1. ✅ **CRUD completo** para proyectos y equipos

2.  **Formularios con validación** usando react-hook-form y zod
3.  **Base de datos completa** con todas las tablas necesarias
4.  **Servicios de API** organizados y tipados
5.  **Componentes reutilizables** de formularios
6.  **Manejo de errores** y notificaciones
7.  **Estructura automática** al crear proyectos
8.  **Asignación de equipos** a proyectos

Testing Sprint 3:

Verificar compilación

```
npm run build
```

Probar funcionalidades:

1. Crear, editar y eliminar proyectos

2. Crear estructura automática de proyecto

3. Gestionar equipos y asignaciones

4. Validaciones de formularios

5. Permisos por rol

6. Responsive design en móviles

Datos de Prueba para Insertar:

-- Insertar proyecto de ejemplo

```
INSERT INTO public.projects (name, address, total_floors, units_per_floor, status,
start_date, estimated_completion)
```

```
VALUES ('Edificio Vista Hermosa', 'Av. Los Robles 2580, Las Condes', 15, 4, 'active',
'2025-08-01', '2025-12-15');
```