









Sistema de Cotizaciones

Un sistema web completo para la gestión y creación de cotizaciones de proyectos, desarrollado con React y TypeScript.

Características

-  Gestión completa de proyectos con datos básicos
-  Work Packages con deliverables y steps detallados
-  Gestión de perfiles profesionales y salarios por país
-  Sistema de licencias y asignación a steps
-  Gestión de costes no operacionales (viajes, subcontrata, IT, otros)
-  Resumen financiero con métricas y márgenes
-  Sistema de autenticación
-  Soporte para proyectos multipaís y multianuales

Estructura del Proyecto

src/

- └─ components/
 - └─ auth/
 - └─ LoginForm.tsx
 - └─ AuthGuard.tsx
 - └─ layout/
 - └─ Header.tsx
 - └─ TabNavigation.tsx
 - └─ Layout.tsx
 - └─ forms/
 - └─ ProjectDataForm.tsx
 - └─ CountrySelector.tsx
 - └─ MultiyearFields.tsx
 - └─ workpackages/
 - └─ WorkPackagesList.tsx
 - └─ WorkPackageCard.tsx
 - └─ DeliverableCard.tsx
 - └─ StepsTable.tsx
 - └─ StepLicensesModal.tsx
 - └─ profiles/
 - └─ ProfilesList.tsx
 - └─ ProfileTable.tsx
 - └─ ProfileForm.tsx
 - └─ licenses/
 - └─ LicensesList.tsx
 - └─ LicenseTable.tsx
 - └─ LicenseForm.tsx
 - └─ costs/
 - └─ TravelCosts.tsx
 - └─ SubcontractCosts.tsx
 - └─ ITCosts.tsx
 - └─ OtherCosts.tsx
 - └─ CostTable.tsx
 - └─ summary/
 - └─ ProjectSummary.tsx
 - └─ FinancialSummary.tsx
 - └─ StatisticsCard.tsx
 - └─ modals/
 - └─ AddClientModal.tsx
 - └─ AddActivityModal.tsx
 - └─ ConfirmDialog.tsx
 - └─ Modal.tsx
 - └─ ui/
 - └─ Button.tsx
 - └─ Input.tsx
 - └─ Select.tsx

```
| | |— Table.tsx
| | |— Card.tsx
| | |— LoadingSpinner.tsx
| | |— EmptyState.tsx
| |— tabs/
| |   |— ProjectDataTab.tsx
| |   |— ProfilesTab.tsx
| |   |— WorkPackagesTab.tsx
| |   |— CostsTab.tsx
| |   |— SummaryTab.tsx
|— types/
|   |— project.ts
|   |— profile.ts
|   |— license.ts
|   |— costs.ts
|   |— auth.ts
|   |— common.ts
|— hooks/
|   |— useAuth.ts
|   |— useProject.ts
|   |— useWorkPackages.ts
|   |— useProfiles.ts
|   |— useLicenses.ts
|   |— useCosts.ts
|   |— useModal.ts
|   |— useLocalStorage.ts
|— stores/
|   |— authStore.ts
|   |— projectStore.ts
|   |— uiStore.ts
|— utils/
|   |— calculations.ts
|   |— formatters.ts
|   |— validators.ts
|   |— constants.ts
|   |— helpers.ts
|— styles/
|   |— globals.css
|   |— components.css
|   |— variables.css
```

Tipos Principales

 **types/project.ts**

Define las interfaces principales para la gestión de proyectos:


```
interface ProjectData {  
  title: string;  
  crmCode: string;  
  client: string;  
  activity: string;  
  startDate: string;  
  businessManager: string;  
  businessUnit: string;  
  opsDomain: string;  
  country: string;  
  scope: 'local' | 'transnational';  
  additionalCountries: string[];  
  multiyear: boolean;  
  startYear?: number;  
  endYear?: number;  
  iqp: string;  
  segmentation: 'New Business' | 'Recurrent';  
  description?: string;  
}
```

```
interface WorkPackage {  
  id: number;  
  name: string;  
  deliverables: Deliverable[];  
}
```

```
interface Deliverable {  
  id: number;  
  name: string;  
  margin: number;  
  yearlyQuantities: Record<number, number>;  
  steps: Step[];  
}
```

```
interface Step {  
  id: number;  
  name: string;  
  profile: string;  
  country: string;  
  processTime: number;  
  timeUnits: 'horas' | 'días' | 'meses';  
  office: boolean;  
  it: boolean;  
  mngPercent: number;  
  nptPercent: number;
```

```
licenses: number[];  
}
```



types/profile.ts

Gestión de perfiles profesionales:

```
typescript  
  
interface Profile {  
  id: number;  
  name: string;  
  salaries: Record<string, number>; // country -> salary  
}
```



types/license.ts

Sistema de licencias:

```
typescript  
  
interface License {  
  id: number;  
  name: string;  
  cost: number;  
  fullProjectCost: boolean;  
}
```



types/costs.ts

Costes no operacionales:

```
typescript
```

```
interface TravelCost {  
  id: number;  
  category: 'vuelos' | 'hoteles' | 'dietas' | 'trenes' | 'otros';  
  quantity: number;  
  unitCost: number;  
  refactorable: boolean;  
}
```

```
interface SubcontractCost {  
  id: number;  
  mode: string;  
  provider: string;  
  unitCost: number;  
  quantity: number;  
  refactorable: boolean;  
}
```

```
interface ITCost {  
  id: number;  
  type: string;  
  name: string;  
  unitCost: number;  
  quantity: number;  
  refactorable: boolean;  
}
```

```
interface OtherCost {  
  id: number;  
  concept: string;  
  unitCost: number;  
  quantity: number;  
  refactorable: boolean;  
}
```



types/auth.ts

Autenticación:

typescript

```
interface LoginCredentials {
  username: string;
  password: string;
}

interface AuthState {
  isLoggedIn: boolean;
  user?: string;
}
```



types/common.ts

Tipos comunes y utilidades:

```
typescript

type TabName = 'project-data' | 'profiles' | 'work-packages' | 'non-operational-costs' | 'summary';

interface ModalProps {
  isOpen: boolean;
  onClose: () => void;
}

interface Country {
  code: string;
  name: string;
}
```

Componentes Clave

Layout Components

- **Layout.tsx**: Componente principal que envuelve toda la aplicación
- **Header.tsx**: Cabecera con título y controles de sesión
- **TabNavigation.tsx**: Navegación entre las diferentes secciones



Form Components

- **ProjectDataForm.tsx**: Formulario principal con datos básicos del proyecto
- **CountrySelector.tsx**: Selector inteligente de países (simple/múltiple)
- **MultiyearFields.tsx**: Campos específicos para proyectos multianuales



Work Package Components

- **WorkPackagesList.tsx**: Lista principal de work packages

- **WorkPackageCard.tsx**: Tarjeta expandible para cada WP
- **DeliverableCard.tsx**: Gestión de deliverables con steps
- **StepsTable.tsx**: Tabla detallada de steps con todas sus propiedades
- **StepLicensesModal.tsx**: Modal para asignar licencias a steps específicos

Profile Components

- **ProfilesList.tsx**: Gestión de perfiles profesionales
- **ProfileTable.tsx**: Tabla con salarios por país
- **ProfileForm.tsx**: Formulario para crear/editar perfiles

License Components

- **LicensesList.tsx**: Lista de licencias disponibles
- **LicenseTable.tsx**: Tabla con detalles de licencias
- **LicenseForm.tsx**: Formulario para gestión de licencias

Cost Components

- **TravelCosts.tsx**: Gestión de gastos de viaje
- **SubcontractCosts.tsx**: Costes de subcontratación
- **ITCosts.tsx**: Costes de infraestructura IT
- **OtherCosts.tsx**: Otros gastos diversos
- **CostTable.tsx**: Componente base reutilizable para tablas de costes

Summary Components

- **ProjectSummary.tsx**: Resumen completo del proyecto
- **FinancialSummary.tsx**: Métricas financieras y márgenes
- **StatisticsCard.tsx**: Tarjetas de estadísticas reutilizables

Modal Components

- **AddClientModal.tsx**: Modal para añadir nuevos clientes
- **AddActivityModal.tsx**: Modal para añadir nuevas actividades
- **ConfirmDialog.tsx**: Diálogo de confirmación genérico
- **Modal.tsx**: Componente base para todos los modales

UI Components

Componentes base reutilizables:

- **Button.tsx**: Botones con variantes (primary, secondary, danger, etc.)

- **Input.tsx**: Inputs con validación y estilos consistentes
- **Select.tsx**: Dropdowns con funcionalidad extendida
- **Table.tsx**: Tablas responsivas con ordenación
- **Card.tsx**: Contenedores con estilos base
- **LoadingSpinner.tsx**: Indicador de carga
- **EmptyState.tsx**: Estados vacíos informativos

Hooks Personalizados

useAuth.ts

Gestión completa de autenticación:

```
typescript  
  
const { isLoggedIn, login, logout, user } = useAuth();
```

useProject.ts

Estado principal del proyecto:

```
typescript  
  
const { projectData, updateProject, saveProject } = useProject();
```

useWorkPackages.ts

Gestión de work packages:

```
typescript  
  
const {  
  workPackages,  
  addWorkPackage,  
  updateWorkPackage,  
  removeWorkPackage,  
  addDeliverable,  
  addStep  
} = useWorkPackages();
```

useProfiles.ts

Gestión de perfiles:

```
typescript
```

```
const { profiles, addProfile, updateProfile, removeProfile } = useProfiles();
```



useLicenses.ts

Gestión de licencias:

typescript

```
const { licenses, addLicense, updateLicense, removeLicense } = useLicenses();
```



useCosts.ts

Gestión de todos los tipos de costes:

typescript

```
const {  
  travelCosts,  
  subcontractCosts,  
  itCosts,  
  otherCosts,  
  addCost,  
  updateCost,  
  removeCost  
} = useCosts();
```



useModal.ts

Control de estado de modales:

typescript

```
const { isOpen, openModal, closeModal } = useModal();
```



useLocalStorage.ts

Persistencia automática:

typescript

```
const [data, setData] = useLocalStorage('projectData', defaultValue);
```



Stores (Estado Global)



authStore.ts

Estado de autenticación global con Zustand



Estado completo del proyecto para compartir entre componentes



Estado de la interfaz (tabs activas, modales abiertos, etc.)

Utilidades



Funciones para cálculos financieros:

- Cálculo de márgenes
- Costes totales por país
- Métricas de proyecto



Formateo de datos:

- Monedas
- Fechas
- Porcentajes
- Números



Validaciones de formularios:

- Validación de emails
- Rangos de fechas
- Campos requeridos



Constantes de la aplicación:

- Listas de países
- Business managers
- Categorías de costes

Instalación y Uso

bash

Instalar dependencias

npm install

Ejecutar en desarrollo

npm run dev

Construir para producción

npm run build

Ejecutar tests

npm run test



Credenciales de Prueba

- **Usuario:** admin
- **Contraseña:** admin



Tecnologías Utilizadas

- **React 18** con TypeScript
- **Hooks personalizados** para lógica de negocio
- **Zustand** para estado global
- **CSS Modules** o **Styled Components** para estilos
- **React Hook Form** para gestión de formularios
- **Zod** para validación de esquemas



Próximas Mejoras

- ☐ Exportación a PDF/Excel
- ☐ Integración con APIs externas
- ☐ Sistema de roles y permisos
- ☐ Historial de versiones de cotizaciones
- ☐ Dashboard con analytics
- ☐ Plantillas de proyectos
- ☐ Notificaciones en tiempo real



Contribuir

1. Fork el proyecto
2. Crea una rama para tu feature (git checkout -b feature/AmazingFeature)
3. Commit tus cambios (git commit -m 'Add some AmazingFeature')

4. Push a la rama (`git push origin feature/AmazingFeature`)

5. Abre un Pull Request



Licencia

Este proyecto está bajo la Licencia MIT - ver el archivo [LICENSE.md](#) para detalles.

Desarrollado con ❤️ para la gestión eficiente de cotizaciones de proyectos