🏠 InmoHouse - Sistema de Gestión Inmobiliaria

InmoHouse es una aplicación web moderna que digitaliza la gestión de propiedades, agentes y clientes para una empresa inmobiliaria. Desarrollada con una arquitectura limpia y escalable, ofrece seguridad, control por roles y una experiencia de usuario optimizada.

🚀 Tecnologías Utilizadas

- Angular 17.3.0 (Standalone Components + Angular Material)
- SCSS + Metodología BEM
- Spring Boot (Java 17)
- MySQL en Railway
- JWT para autenticación y autorización
- Vercel para frontend | Railway para backend y base de datos
- ngx-charts para gráficas dinámicas
- Exportación de datos a Excel y PDF

📦 Arquitectura del Proyecto

La estructura sigue los principios de Clean Architecture adaptados, basada en capas desacopladas:

- `@domain`: define modelos, interfaces, contratos y gateways.
- `@application`: gestiona los casos de uso (`listar-clientes.usecase.ts`, `crear-propiedad.usecase.ts`, etc.).
- `@infrastructure`: implementa adaptadores HTTP y almacenamiento (`AuthStorageAdapter`, `UsuarioHttpService`, etc.).
- `@components`: contiene componentes standalone divididos por roles (admin, agente, cliente).

🔐 Seguridad y Roles

- Redirección automática post-login según el rol
- `RoleGuard` protege rutas específicas
- Token JWT decodificado localmente para control de acceso
- Edición/eliminación de recursos condicionada por permisos
- Pantallas personalizadas para errores 403 y 404

📊 Funcionalidades Destacadas

- Dashboards separados para administrador, agente y cliente
- CRUD completo de propiedades y usuarios
- Gestión de clientes por parte de agentes
- Vista de propiedades para clientes sin edición
- Estadísticas de propiedades por agente y tipo

- Exportación de datos (Excel, PDF)
- Diseño responsive para desktop, tablet y móvil

Estadísticas Inteligentes

- Visualización con `ngx-charts`
- Cálculo de porcentajes por tipo y agente
- Etiquetas dinámicas como: `12 • 23.1%`
- Información resumen: tipo con mayor participación, total global

---

Credenciales de Prueba

- **Administrador:** oscar@mail.com / 12345
- **Agente:** agente@mail.com / 12345
- **Cliente:** cliente@mail.com / 12345

Backend (Spring Boot + Railway)

```
### Frontend (Angular - Vercel)

git clone https://github.com/colombianet/back-users
cd back-users
# Configura tus variables en application.properties
./mvnw spring-boot:run

## Documentación Técnica

---

### Arquitectura basada en Clean Architecture

El proyecto sigue un enfoque desacoplado y modular, dividido en capas:

- `@domain`: Modelos, interfaces, gateways y contratos.
- `@application`: Casos de uso divididos por contexto (`crear-
propiedad.usecase.ts`, `listar-usuarios.usecase.ts`, etc.).
- `@infrastructure`: Adapta servicios HTTP y almacenamiento local
(`AuthStorageAdapter`, `UsuarioHttpService`, etc.).
- `@presentation/components`: Componentes standalone organizados por rol (admin,
agente, cliente).

---

### Componentes standalone

Cada dashboard está encapsulado y contiene sus propios módulos:
```

- `AdminDashboardComponent`
- `AgenteDashboardComponent`
- `ClienteDashboardComponent`

Todos utilizan `inject()` para acceder a dependencias, eliminando constructores largos y facilitando testeo.

🗂 Modularización de Angular Material

Los módulos de Angular Material se agrupan en `MaterialModule`:

```ts
import { MaterialModule } from '@shared/material.module';
```

Esto reduce decenas de líneas de importación por componente. Exporta:

- `MatButtonModule`
- `MatCardModule`
- `MatDialogModule`
- `MatIconModule`
- `MatInputModule`
- `MatPaginatorModule`
- `MatProgressSpinnerModule`
- `MatSelectModule`
- `MatSnackBarModule`
- `MatTableModule`
- `MatTabsModule`
- `MatTooltipModule`
- `MatFormFieldModule`
- `MatMenuModule`
- `MatCheckboxModule`
- `MatRadioModule`
- `MatSlideToggleModule`
- `MatDatepickerModule`
- `MatNativeDateModule`
- `MatExpansionModule`

🪄 Inyección simplificada con `inject()` + `providers`

Para reducir aún más líneas, se centralizan los providers en un archivo externo: