📋 DOCUMENTACIÓN COMPLETA - FRONTEND ANGULAR GALITROCO

1. RESUMEN EJECUTIVO

Fecha: 2 de octubre de 2025
Proyecto: GaliTroco - Sistema de Intercambio de Habilidades/Servicios
Fase: Frontend Angular + Preparación Deploy

Objetivos completados:

- ☑ Creación de proyecto Angular 19.2 con arquitectura modular
- ☑ Implementación de servicios core conectados a API backend
- ☑ Sistema de autenticación completo (login/register)
- ☑ Componentes de habilidades con filtros y paginación
- ☑ Layout responsive con Material Design
- ☑ Guards y interceptors para seguridad

---

2. STACK TECNOLÓGICO FRONTEND

Tecnologías principales:

- **Angular:** 19.2.0 (última versión estable)
- **TypeScript:** 5.x
- **Angular Material:** 19.2.19 (UI components)
- **RxJS:** Para programación reactiva
- **SCSS:** Preprocesador CSS

Herramientas de desarrollo:

- **Angular CLI:** 19.2.0
- **Node.js:** 22.17.0
- **npm:** 10.1.0
- **VS Code:** Editor con extensiones Angular

---

3. ARQUITECTURA DEL PROYECTO

3.1 Estructura de carpetas creada:

```
frontend/
├── src/
│   ├── app/
│   │   ├── core/                    # Servicios singleton y utilidades core
│   │   │   ├── services/
```

```
│   │   │   │   ├── api.service.ts         # Cliente HTTP base
│   │   │   │   ├── auth.service.ts        # Gestión autenticación
│   │   │   │   ├── storage.service.ts     # LocalStorage wrapper
│   │   │   │   ├── habilidades.service.ts # CRUD habilidades
│   │   │   │   └── categorias.service.ts  # Listado categorías
│   │   │   ├── guards/
│   │   │   │   ├── auth.guard.ts          # Proteger rutas privadas
│   │   │   │   └── admin.guard.ts         # Proteger rutas admin
│   │   │   └── interceptors/
│   │   │       └── auth.interceptor.ts    # Manejo global errores HTTP
│   │   ├── shared/                  # Componentes y modelos compartidos
│   │   │   ├── models/
│   │   │   │   ├── user.model.ts          # Interfaces User, Auth
│   │   │   │   ├── habilidad.model.ts     # Interfaces Habilidad
│   │   │   │   ├── categoria.model.ts     # Interface Categoria
│   │   │   │   ├── api-response.model.ts  # Interfaces respuestas API
│   │   │   │   └── index.ts               # Barrel export
│   │   │   └── components/                # (vacío - futuro)
│   │   ├── features/                # Módulos funcionales por feature
│   │   │   ├── auth/
│   │   │   │   ├── login/
│   │   │   │   │   ├── login.component.ts
│   │   │   │   │   ├── login.component.html
│   │   │   │   │   └── login.component.scss
│   │   │   │   └── register/
│   │   │   │       ├── register.component.ts
│   │   │   │       ├── register.component.html
│   │   │   │       └── register.component.scss
│   │   │   ├── home/
│   │   │   │   ├── home.component.ts
│   │   │   │   ├── home.component.html
│   │   │   │   └── home.component.scss
│   │   │   ├── habilidades/
│   │   │   │   ├── habilidades-list/
│   │   │   │   │   ├── habilidades-list.component.ts
│   │   │   │   │   ├── habilidades-list.component.html
│   │   │   │   │   └── habilidades-list.component.scss
│   │   │   │   ├── habilidad-detail/
│   │   │   │   │   ├── habilidad-detail.component.ts
│   │   │   │   │   ├── habilidad-detail.component.html
│   │   │   │   │   └── habilidad-detail.component.scss
│   │   │   │   └── habilidad-form/
│   │   │   │       ├── habilidad-form.component.ts
│   │   │   │       ├── habilidad-form.component.html
│   │   │   │       └── habilidad-form.component.scss
│   │   │   └── perfil/
│   │   │       └── perfil.component.ts
│   │   ├── layout/                  # (futuro - header/footer separados)
│   │   ├── app.component.ts         # Componente raíz con layout
│   │   ├── app.component.html       # Template con header/footer
│   │   ├── app.component.scss       # Estilos globales layout
│   │   ├── app.config.ts            # Configuración providers
│   │   └── app.routes.ts            # Definición de rutas
```

```
│   ├── environments/
│   │   ├── environment.ts          # Configuración desarrollo
│   │   └── environment.prod.ts     # Configuración producción
│   ├── index.html                  # HTML principal
│   ├── main.ts                     # Bootstrap Angular
│   └── styles.scss                 # Estilos globales
├── angular.json                    # Configuración Angular CLI
├── package.json                    # Dependencias npm
├── tsconfig.json                   # Configuración TypeScript
└── README.md                       # Documentación proyecto
```

Total archivos creados: 32 archivos TypeScript + HTML + SCSS

---

4. MODELOS DE DATOS (TypeScript Interfaces)

4.1 User Model

```typescript
export interface User {
  id: number;
  nombre_usuario: string;
  email: string;
  rol: 'usuario' | 'administrador';
  ubicacion: string;
  biografia?: string;
  foto_url?: string;
  activo: boolean;
  fecha_registro?: string;
  ultima_conexion?: string;
}

export interface LoginRequest {
  email: string;
  password: string;
}

export interface RegisterRequest {
  nombre_usuario: string;
  email: string;
  password: string;
  ubicacion: string;
}

export interface AuthResponse {
  success: boolean;
  message: string;
  data: {
    user: User;
    token: string;
```

```
  };
}
```

4.2 Habilidad Model

```
export interface Habilidad {
  id: number;
  usuario_id: number;
  categoria_id: number;
  tipo: 'oferta' | 'demanda';
  titulo: string;
  descripcion: string;
  estado: 'activa' | 'pausada' | 'intercambiada';
  duracion_estimada?: number;
  fecha_publicacion: string;
  // Datos JOIN
  categoria?: string;
  categoria_icono?: string;
  usuario_nombre?: string;
  usuario_ubicacion?: string;
}
```

---

5. SERVICIOS CORE

5.1 ApiService

Propósito: Cliente HTTP base para todas las peticiones

Métodos:

- `get<T>(resource, params?): Observable<T>`
- `post<T>(resource, body): Observable<T>`
- `put<T>(resource, body): Observable<T>`
- `delete<T>(resource): Observable<T>`

Característica clave:

```
{ withCredentials: true }  // Para sesiones PHP
```

5.2 AuthService

Propósito: Gestión de autenticación con estado reactivo

Métodos:

- `login(credentials): Observable<AuthResponse>`

- `register(data): Observable<AuthResponse>`
- `logout(): Observable<ApiResponse<any>>`
- `getCurrentUser(): Observable<ApiResponse<User>>`
- `isAuthenticated(): boolean`
- `isAdmin(): boolean`

Estado reactivo:

```
public currentUser$: Observable<User | null>
```

5.3 HabilidadesService

Métodos:

- `list(params?): Observable<PaginatedResponse<Habilidad>>`
- `getById(id): Observable<Habilidad>`
- `create(data): Observable<Habilidad>`
- `update(id, data): Observable<Habilidad>`
- `delete(id): Observable<any>`

---

6. COMPONENTES IMPLEMENTADOS

6.1 LoginComponent

- Formulario reactivo con validaciones
- Email + password
- Toggle mostrar/ocultar contraseña
- Redirección a `returnUrl` después de login

6.2 RegisterComponent

- Formulario con 5 campos
- Validador custom: contraseñas deben coincidir
- Auto-login después de registro exitoso

6.3 HomeComponent

- Landing page con gradiente
- 3 secciones: Hero, Features, CTA
- Responsive design

6.4 HabilidadesListComponent

- **Filtros reactivos:** búsqueda, tipo, categoría, ubicación
- **Paginación:** 6/12/24/48 items por página
- **Grid responsive:** Tarjetas Material Design

- **Loading state** y **empty state**

6.5 Otros componentes (stubs)

- HabilidadDetailComponent
- HabilidadFormComponent
- PerfilComponent

---

7. SEGURIDAD

7.1 AuthGuard

Protege rutas que requieren autenticación:

```
canActivate: [authGuard]
```

7.2 AuthInterceptor

Manejo global de errores:

- **401:** Limpia sesión → redirige a login
- **403:** Redirige a home

---

8. INTEGRACIÓN CON BACKEND

8.1 URL API

```
https://render-test-php-1.onrender.com/api.php
```

8.2 Patrón de peticiones

```
GET  /api.php?resource=habilidades&page=1
POST /api.php?resource=auth/login
PUT  /api.php?resource=habilidades/5
```

8.3 Sesiones PHP

Crítico: Todas las peticiones usan `withCredentials: true` para enviar cookie `PHPSESSID`.

---

9. COMANDOS ÚTILES

Desarrollo

```
cd frontend
ng serve                  # Puerto 4200
ng serve --open           # Con auto-open
ng serve --port 4300      # Puerto custom
```

Build

```
ng build                              # Desarrollo
ng build --configuration production   # Producción
```

Output: `dist/frontend/browser/`

Generar componentes

```
ng generate component features/nombre
ng generate service core/services/nombre
ng generate guard core/guards/nombre
```

---

10. PRÓXIMOS PASOS

Desarrollo pendiente:

- ⚠ Implementar HabilidadDetailComponent completo
- ⚠ Implementar HabilidadFormComponent (crear/editar)
- ⚠ Implementar PerfilComponent
- ⚠ Sistema de intercambios
- ⚠ Mensajería
- ⚠ Valoraciones

Deploy en Render:

1. Crear Dockerfile para frontend
2. Build de Angular
3. Servir con nginx
4. Crear servicio en Render
5. Conectar GitHub auto-deploy

---

11. MÉTRICAS

Archivos creados: 32 archivos
Líneas de código: ~2.500 líneas

Componentes: 7 componentes funcionales
Servicios: 5 servicios core
Guards: 2 guards
Interceptors: 1 interceptor
Modelos/Interfaces: 12 interfaces TypeScript

---

12. DECISIONES TÉCNICAS

¿Por qué Standalone Components?

- No necesita NgModules
- Más ligero y tree-shakeable
- Estándar en Angular 15+

¿Por qué BehaviorSubject para Auth?

- Estado inicial desde localStorage
- Emite último valor a nuevos suscriptores
- Reactivo: componentes escuchan cambios

¿Por qué Lazy Loading?

- Bundle inicial más pequeño
- Cada ruta carga bajo demanda
- Mejor performance inicial

FIN DE LA DOCUMENTACIÓN

Autor: GitHub Copilot
Proyecto: GaliTroco TFM
Fecha: 2 octubre 2025