



TESTING MANUAL FRONTEND - GALITROCO

Fecha: 23 de octubre de 2025

URL Local: <http://localhost:4200>

Backend: <https://render-test-php-1.onrender.com/api.php>

Estado: En progreso ⌚



CHECKLIST DE TESTING

☒ TEST 1: PÁGINA DE INICIO

URL: <http://localhost:4200/>

Verificar:

- ☐ Se carga la página sin errores
- ☐ Aparece el título "GaliTroco"
- ☐ Hay 2 botones: "Comenzar Ahora" y "Explorar Habilidades"
- ☐ Se ven las 3 cards de características
- ☐ No hay errores en la consola del navegador (F12)

Acciones:

1. Abrir navegador en <http://localhost:4200>
2. Abrir DevTools (F12) → Console
3. Verificar que no hay errores CORS
4. Click en "Explorar Habilidades" → debe ir a </habilidades>

Resultado esperado: ☒ Página carga correctamente

☒ TEST 2: LISTAR HABILIDADES (SIN LOGIN)

URL: <http://localhost:4200/habilidades>

Verificar:

- ☐ Se cargan las habilidades desde el backend
- ☐ Aparecen las 25 habilidades existentes
- ☐ Se ven los filtros: búsqueda, tipo, categoría, ubicación
- ☐ Hay paginación en la parte inferior
- ☐ Cada card muestra: título, descripción, tipo, categoría, usuario

Acciones:

1. Ir a </habilidades>
2. Verificar en DevTools → Network → ver llamada a:

```
https://render-test-php-1.onrender.com/api.php?
resource=habilidades&page=1&per_page=12
```

3. Verificar respuesta JSON con **success: true**
4. Probar filtro por tipo: "Oferta"
5. Probar búsqueda: escribir "angular"

Resultado esperado: ☒ Listado funciona, filtros operativos

Posibles errores:

- ✗ CORS: **Access-Control-Allow-Origin** → Revisar backend
- ✗ 401 Unauthorized: Endpoint requiere autenticación
- ✗ Timeout: Backend lento o caído

☒ **TEST 3: VER DETALLE DE HABILIDAD (SIN LOGIN)**

URL: <http://localhost:4200/habilidades/1>

Verificar:

- ☐ Se carga el detalle completo
- ☐ Muestra: título, descripción, tipo, categoría, duración, usuario propietario
- ☐ Hay botón "Proponer Intercambio" (puede estar deshabilitado sin login)
- ☐ Se ve información del usuario (nombre, ubicación)

Acciones:

1. Desde listado, click en una habilidad
2. Verificar llamada a: **?resource=habilidades/1**
3. Ver que se muestra toda la información

Resultado esperado: ☒ Detalle se carga correctamente

☒ **TEST 4: REGISTRO DE NUEVO USUARIO**

URL: <http://localhost:4200/register>

Verificar:

- ☐ Formulario con campos: nombre_usuario, email, password, confirmar password, ubicación
- ☐ Validaciones funcionan (email válido, password mínimo 6 caracteres)
- ☐ Botón "Registrarse" deshabilitado hasta completar correctamente

Acciones:

1. Ir a **/register**
2. Completar formulario:

```
Nombre de usuario: testfrontend_001
Email: testfrontend_001@test.com
Password: Test123456
Confirmar Password: Test123456
Ubicación: Santiago de Compostela, Galicia
```

3. Click en "Registrarse"
4. Verificar en DevTools → Network:

```
POST https://render-test-php-1.onrender.com/api.php?
resource=auth/register
Body: { nombre_usuario, email, password, ubicacion }
```

5. Si OK → Debe redirigir a [/habilidades](#) con usuario autenticado
6. Verificar en DevTools → Application → Local Storage:

```
user: { id, nombre_usuario, email, rol }
token: "xxx..."
```

Resultado esperado: ☒ Usuario creado y login automático

Posibles errores:

- ✗ 400 Bad Request: Email ya existe
- ✗ Validación de contraseñas no coinciden
- ✗ No se guarda en localStorage

☒ **TEST 5: LOGIN CON USUARIO EXISTENTE**

URL: <http://localhost:4200/login>

Datos de prueba del backend:

```
Email: test_6937@testmail.com
Password: Pass123456
```

Verificar:

- ☐ Formulario con email y password
- ☐ Validaciones funcionan
- ☐ Botón "Iniciar Sesión"
- ☐ Link a "¿Olvidaste tu contraseña?"

Acciones:

1. Ir a </login>
2. Ingresar credenciales de prueba
3. Click en "Iniciar Sesión"
4. Verificar llamada:

```
POST ?resource=auth/login
Body: { email, password }
Response: { success: true, data: { user, token } }
```

5. Debe redirigir a </habilidades>
6. Verificar que header muestra nombre de usuario
7. Verificar localStorage tiene [user](#) y [token](#)

Resultado esperado: ☒ Login exitoso y redirección

Posibles errores:

- ☒ 401 Unauthorized: Credenciales incorrectas
- ☒ No redirige tras login exitoso
- ☒ No se actualiza el header con usuario

TEST 6: CREAR HABILIDAD (REQUIERE LOGIN)

URL: <http://localhost:4200/habilidades/nueva>

PRE-REQUISITO: Estar autenticado (completar TEST 5 primero)

Verificar:

- ☐ Si no estás autenticado → redirige a </login>
- ☐ Formulario con: categoría, tipo, título, descripción, duración estimada
- ☐ Validaciones: todos los campos requeridos

Acciones:

1. Asegurarse de estar autenticado
2. Ir a </habilidades/nueva>
3. Completar formulario:

```
Categoría: Tecnología e Informática (ID: 2)
Tipo: Oferta
Título: Testing Frontend Angular + Backend PHP
Descripción: Prueba de integración completa entre Angular 19 y PHP 8.2
con PostgreSQL
Duración estimada: 120 minutos
```

4. Click en "Guardar"

5. Verificar llamada:

```
POST ?resource=habilidades
Headers: withCredentials: true (para sesión PHP)
Body: { categoria_id, tipo, titulo, descripcion, duracion_estimada }
Response: { success: true, data: { habilidad_id: X } }
```

6. Debe redirigir a [/habilidades](#) y aparecer la nueva habilidad

Resultado esperado: ☒ Habilidad creada exitosamente

Posibles errores:

- ☒ 401 Unauthorized: Sesión expirada
- ☒ 400 Bad Request: Validación de campos
- ☒ No aparece en el listado tras crear

TEST 7: EDITAR HABILIDAD PROPIA

URL: <http://localhost:4200/habilidades/{id}/editar>

PRE-REQUISITO: Haber creado una habilidad en TEST 6

Verificar:

- ☐ Solo puedes editar tus propias habilidades
- ☐ Formulario pre-cargado con datos existentes
- ☐ Puedes cambiar: título, descripción, estado, duración

Acciones:

1. Ir al listado de habilidades
2. Buscar tu habilidad recién creada
3. Click en "Editar" (si hay botón) o ir a [/habilidades/{id}/editar](#)
4. Cambiar descripción: agregar "EDITADO desde frontend"
5. Guardar cambios
6. Verificar llamada:

```
PUT ?resource=habilidades/{id}
Body: { titulo, descripcion, estado, duracion_estimada }
```

7. Verificar que se actualizó en el listado

Resultado esperado: ☒ Habilidad editada correctamente

TEST 8: ELIMINAR HABILIDAD PROPIA

URL: Desde listado o detalle

PRE-REQUISITO: Tener una habilidad propia

Verificar:

- ☐ Botón "Eliminar" solo en habilidades propias
- ☐ Dialog de confirmación antes de eliminar
- ☐ Tras eliminar, desaparece del listado

Acciones:

1. Ir a detalle de habilidad propia
2. Click en "Eliminar"
3. Confirmar eliminación
4. Verificar llamada:

```
DELETE ?resource=habilidades/{id}
Response: { success: true, message: "Habilidad eliminada correctamente" }
```

5. Verificar que ya no aparece en [/habilidades](#)

Resultado esperado: ☒ Habilidad eliminada (soft delete)

TEST 9: VER MIS INTERCAMBIOS

URL: <http://localhost:4200/intercambios>

PRE-REQUISITO: Estar autenticado

Verificar:

- ☐ Requiere autenticación (redirige a login si no)
- ☐ Muestra lista de intercambios del usuario
- ☐ Puede estar vacía si no has propuesto ninguno
- ☐ Filtros por estado: propuesto, aceptado, rechazado, completado

Acciones:

1. Login con usuario que tenga intercambios
2. Ir a [/intercambios](#)
3. Verificar llamada:

```
GET ?resource=intercambios
Response: { success: true, data: [ intercambios array ] }
```

4. Verificar que se muestran intercambios con:

- Habilidad ofrecida
- Habilidad solicitada
- Estado
- Usuario con quien intercambias
- Fecha

Resultado esperado: ☒ Lista de intercambios visible

TEST 10: PROPONER INTERCAMBIO

URL: Desde detalle de habilidad

PRE-REQUISITO:

- Estar autenticado
- Tener al menos 1 habilidad propia
- Ver una habilidad de otro usuario

Verificar:

- ☐ Botón "Proponer Intercambio" en detalle de habilidad ajena
- ☐ Abre dialog con:
 - Habilidad que solicitas (la que estás viendo)
 - Dropdown para elegir tu habilidad a ofrecer
 - Textarea para mensaje

Acciones:

1. Login con usuario A (test_6937@testmail.com)
2. Ir a una habilidad de otro usuario (ej: ID 28)
3. Click en "Proponer Intercambio"
4. Seleccionar tu habilidad a ofrecer (ej: ID 26)
5. Escribir mensaje: "Me interesa mucho tu habilidad, podemos intercambiar?"
6. Click en "Enviar Propuesta"
7. Verificar llamada:

```
POST ?resource=intercambios
Body: {
  habilidad_ofrecida_id: 26,
  habilidad_solicitada_id: 28,
  mensaje_propuesta: "...
}
Response: { success: true, data: { intercambio_id: X } }
```

8. Debe aparecer en [/intercambios](#) con estado "propuesto"

Resultado esperado: ☒ Intercambio propuesto exitosamente

Posibles errores:

- ✗ No puedes intercambiar si no tienes habilidades propias
- ✗ No puedes proponer intercambio con tu propia habilidad
- ✗ Botón no aparece (pendiente implementar)

🔒 TEST 11: ACEPTAR/RECHAZAR INTERCAMBIO

URL: <http://localhost:4200/intercambios>

PRE-REQUISITO:

- Ser el receptor de un intercambio en estado "propuesto"
- Login con usuario B (userB_6566@testing.com)

Verificar:

- ☐ Botones "Aceptar" y "Rechazar" solo para receptor
- ☐ Solo en intercambios con estado "propuesto"
- ☐ Tras aceptar → estado cambia a "aceptado"
- ☐ Tras rechazar → estado cambia a "rechazado"

Acciones:

1. Login con usuario B (receptor del intercambio)
2. Ir a [/intercambios](#)
3. Ver intercambio en estado "propuesto"
4. Click en "Aceptar"
5. Verificar llamada:

```
PUT ?resource=intercambios/{id}
Body: { estado: "aceptado" }
Response: { success: true, data: { mensaje, nuevo_estado } }
```

6. Verificar que estado cambió a "aceptado" en la UI

Resultado esperado: ☒ Intercambio aceptado

⚠ **PENDIENTE:** Botones aceptar/rechazar pueden no estar implementados aún

🔒 TEST 12: COMPLETAR INTERCAMBIO

URL: <http://localhost:4200/intercambios>

PRE-REQUISITO:

- Tener un intercambio en estado "aceptado"
- Ser proponente o receptor

Verificar:

- ☐ Botón "Marcar como Completado"
- ☐ Solo en intercambios "aceptados"
- ☐ Tras completar → estado "completado"
- ☐ Aparece opción para valorar


Acciones:

1. Login con usuario que tenga intercambio aceptado
2. Ir a [/intercambios](#)
3. Click en "Marcar como Completado"
4. Verificar llamada:

```
PUT ?resource=intercambios/{id}/completar
Response: { success: true, data: { mensaje: "Ahora puedes dejar una valoración" } }
```

5. Estado cambia a "completado"

Resultado esperado: ☒ Intercambio completado

 **PENDIENTE:** Botón completar puede no estar implementado aún

TEST 13: CREAR VALORACIÓN

URL: Desde intercambio completado

PRE-REQUISITO: Tener intercambio en estado "completado"

Verificar:

- ☐ Botón "Valorar" aparece solo en intercambios completados
- ☐ Formulario con:
 - Rating de estrellas (1-5)
 - Textarea para comentario
- ☐ Solo puedes valorar una vez por intercambio


Acciones:

1. Login con usuario A
2. Ir a intercambio completado (ID: 17)
3. Click en "Valorar"
4. Seleccionar 5 estrellas
5. Escribir comentario: "Excelente intercambio, muy profesional"
6. Enviar valoración
7. Verificar llamada:

```
POST ?resource=valoraciones
Body: {
  evaluado_id: X,
  intercambio_id: 17,
  puntuacion: 5,
  comentario: "...
}
Response: { success: true, message: "Valoración enviada correctamente" }
```

8. Verificar que ya no aparece botón "Valorar"

Resultado esperado: ☒ Valoración creada

 **PENDIENTE:** Formulario de valoración no implementado aún

TEST 14: VER PERFIL DE USUARIO

URL: <http://localhost:4200/perfil/{id}> (público) o [/perfil](http://localhost:4200/perfil) (propio)

Verificar:

- ☐ Muestra información del usuario
- ☐ Lista sus habilidades activas
- ☐ Muestra valoraciones recibidas con rating promedio
- ☐ Botón "Proponer Intercambio" si no eres tú

Acciones:

1. Ir a [/perfil/21](http://localhost:4200/perfil/21) (Usuario A)
2. Ver habilidades del usuario
3. Ver valoraciones (debe aparecer la del TEST 13)
4. Verificar rating promedio
5. Si eres otro usuario, debe haber botón para proponer intercambio

Resultado esperado: ☒ Perfil público funciona

TEST 15: PANEL ADMIN - REPORTES

URL: <http://localhost:4200/admin/reportes>

PRE-REQUISITO: Login como administrador

Datos admin:

Email: admin@galitroco.com
Password: Admin123456

Verificar:

- ☐ Solo accesible para rol "administrador"
- ☐ Lista todos los reportes del sistema
- ☐ Filtros por estado: pendiente, revisado, resuelto
- ☐ Botón "Resolver" en cada reporte
- ☐ Dialog para añadir notas de revisión

Acciones:

1. Login como admin
2. Ir a </admin/reportes>
3. Verificar llamada:

```
GET ?resource=reportes  
Response: { success: true, data: [ reportes array ] }
```

4. Click en "Resolver" de un reporte
5. Añadir notas: "Reporte revisado - Contenido apropiado"
6. Cambiar estado a "revisado"
7. Verificar llamada:

```
PUT ?resource=reportes/{id}  
Body: { estado: "revisado", notas_revision: "..."} }
```

Resultado esperado: ☒ Admin puede gestionar reportes

☒ TEST 16: LOGOUT

URL: Cualquier página autenticada

Verificar:

- ☐ Botón "Cerrar Sesión" en header/menú
- ☐ Tras logout → redirige a </login> o </>
- ☐ localStorage se limpia (user y token eliminados)
- ☐ No puede acceder a rutas protegidas

Acciones:

1. Estando autenticado, click en "Cerrar Sesión"
2. Verificar llamada:

```
POST ?resource=auth/logout  
Response: { success: true, message: "Logout exitoso" }
```

3. Verificar en DevTools → Application → Local Storage vacío
4. Intentar ir a `/intercambios` → debe redirigir a `/login`

Resultado esperado: ☒ Logout funciona correctamente

RESUMEN DE TESTS

Tests Básicos (Sin autenticación)

- ☐ TEST 1: Página de inicio
- ☐ TEST 2: Listar habilidades
- ☐ TEST 3: Ver detalle habilidad
- ☐ TEST 4: Registro
- ☐ TEST 5: Login

Tests Autenticados (Usuario)

- ☐ TEST 6: Crear habilidad
- ☐ TEST 7: Editar habilidad
- ☐ TEST 8: Eliminar habilidad
- ☐ TEST 9: Ver mis intercambios
- ☐ TEST 10: Proponer intercambio
- ☐ TEST 11: Aceptar/Rechazar intercambio ⚠
- ☐ TEST 12: Completar intercambio ⚠
- ☐ TEST 13: Crear valoración ⚠
- ☐ TEST 14: Ver perfil usuario
- ☐ TEST 16: Logout

Tests Admin

- ☐ TEST 15: Panel de reportes

ERRORES COMUNES A VERIFICAR

1. CORS (Cross-Origin Resource Sharing)

```
Error en consola:  
Access to fetch at '...' from origin 'http://localhost:4200' has been blocked  
by CORS policy
```

Solución: Verificar backend tiene headers CORS correctos

2. Sesiones PHP no funcionan

Error: 401 Unauthorized en endpoints protegidos

Causa: `withCredentials: true` no configurado o cookies bloqueadas

Solución: Verificar que `api.service.ts` tiene `withCredentials: true` en todas las peticiones

3. Datos no se actualizan en tiempo real

Habilidad creada pero no aparece en listado

Solución: Recargar componente tras crear/editar (llamar a `loadHabilidades()`)

4. Botones no aparecen

No se ve botón "Proponer Intercambio" o "Aceptar"

Estado: ⚠ Puede estar pendiente de implementar

SIGUIENTE PASO

EMPEZAR POR:

1. TEST 1: Página de inicio → Verificar que carga sin errores
2. TEST 2: Listar habilidades → Ver si conecta con backend
3. TEST 5: Login → Probar autenticación
4. TEST 6: Crear habilidad → Verificar integración completa

¿Empezamos con TEST 1? Abre el navegador en <http://localhost:4200> y cuéntame qué ves 🚀

Última actualización: 23 de octubre de 2025

Estado servidor: ☒ Angular corriendo en <http://localhost:4200>

Estado backend: ☒ Render operativo al 92%