



Versión: 1.0.0

Proyecto: TFM - Plataforma de Intercambio de Habilidades

- Formato de endpoints:** /api.php?resource={endpoint}



Login

Respuesta:

PROFESSEUR : M.DA ROS

Endpoints Implementados (13-14 Oct 2025)

INTERCAMBIOS

1. Listar Mis Intercambios

```
GET /api.php?resource=intercambios
GET /api.php?resource=intercambios&estado=propuesto
```

Parámetros Query (opcionales):

- **estado**: Filtrar por estado (**propuesto**, **aceptado**, **rechazado**, **completado**, **cancelado**)

Respuesta exitosa (200):

```
{
  "success": true,
  "message": "OK",
  "data": [
    {
      "id": 1,
      "estado": "propuesto",
      "mensaje_propuesta": "Hola! Me interesa tu habilidad...",
      "fecha_propuesta": "2025-10-13 10:30:00+02",
      "ultima_actualizacion": "2025-10-13 10:30:00+02",
      "fecha_completado": null,
      "habilidad_ofrecida_id": 5,
      "habilidad_ofrecida_titulo": "Clases de guitarra",
      "habilidad_ofrecida_descripcion": "Enseño guitarra española...",
      "habilidad_ofrecida_duracion": 60,
      "habilidad_ofrecida_categoria": "Arte y Creatividad",
      "habilidad_solicitada_id": 10,
      "habilidad_solicitada_titulo": "Reparación de PC",
      "habilidad_solicitada_descripcion": "Arreglo ordenadores...",
      "habilidad_solicitada_duracion": 120,
      "habilidad_solicitada_categoria": "Tecnología e Informática",
      "proponente_id": 2,
      "proponente_nombre": "usuario1",
      "proponente_ubicacion": "A Coruña, Galicia",
      "proponente_foto": null,
      "receptor_id": 3,
      "receptor_nombre": "usuario2",
      "receptor_ubicacion": "Santiago, Galicia",
      "receptor_foto": null
    }
  ]
}
```

Ejemplo PowerShell:

```
# Con sesión autenticada
$intercambios = Invoke-RestMethod -Uri "https://render-test-php-1.onrender.com/api.php?resource=intercambios" -Method GET -WebSession $session
```

Ejemplo curl:

```
curl -X GET "https://render-test-php-1.onrender.com/api.php?resource=intercambios" \
-H "Cookie: PHPSESSID=your_session_id"
```

2. 🔍 Obtener Intercambio Específico 🔒

```
GET /api.php?resource=intercambios/{id}
```

Parámetros URL:

- **id**: ID del intercambio

Respuesta exitosa (200):

```
{
  "success": true,
  "message": "OK",
  "data": {
    "id": 1,
    "estado": "propuesto",
    "mensaje_propuesta": "Hola! Me interesa intercambiar...",
    "fecha_propuesta": "2025-10-13 10:30:00+02",
    "ultima_actualizacion": "2025-10-13 10:30:00+02",
    "fecha_completado": null,
    "notas_adicionales": null,
    "habilidad_ofrecida_id": 5,
    "habilidad_ofrecida_titulo": "Clases de guitarra",
    "habilidad_ofrecida_descripcion": "Enseño guitarra española...",
    "habilidad_ofrecida_duracion": 60,
    "habilidad_ofrecida_categoria": "Arte y Creatividad",
    "habilidad_solicitada_id": 10,
    "habilidad_solicitada_titulo": "Reparación de PC",
    "habilidad_solicitada_descripcion": "Arreglo ordenadores...",
    "habilidad_solicitada_duracion": 120,
    "habilidad_solicitada_categoria": "Tecnología e Informática",
    "proponente_id": 2,
```

```
"proponente_nombre": "usuario1",
"proponente_email": "usuario1@example.com",
"proponente_ubicacion": "A Coruña, Galicia",
"proponente_foto": null,
"receptor_id": 3,
"receptor_nombre": "usuario2",
"receptor_email": "usuario2@example.com",
"receptor_ubicacion": "Santiago, Galicia",
"receptor_foto": null
}
```

Errores posibles:

- 404: Intercambio no encontrado o no tienes acceso
- 401: No autenticado

Ejemplo PowerShell:

```
$detalle = Invoke-RestMethod -Uri "https://render-test-php-1.onrender.com/api.php?resource=intercambios/7" -Method GET -WebSession $session
```

3. 🎯 Proponer Nuevo Intercambio 🔒

```
POST /api.php?resource=intercambios
Content-Type: application/json
```

Body:

```
{
  "habilidad_ofrecida_id": 5,
  "habilidad_solicitada_id": 10,
  "mensaje_propuesta": "Hola! Me interesa mucho tu habilidad. ¿Te interesaría intercambiar?"
}
```

Validaciones:

- `habilidad_ofrecida_id` debe ser una habilidad tuya y estar activa
- `habilidad_solicitada_id` debe existir y estar activa
- No puedes proponer intercambios contigo mismo
- `mensaje_propuesta` es opcional

Respuesta exitosa (201):

```
{
  "success": true,
  "message": "Intercambio propuesto exitosamente",
  "data": {
    "intercambio_id": 15,
    "mensaje": "Intercambio propuesto exitosamente"
  }
}
```

Lógica automática:

- Crea el intercambio en estado **propuesto**
- Crea una notificación automática para el receptor
- Registra la fecha de propuesta

Errores posibles:

- **400**: Campos faltantes o inválidos
- **403**: Solo puedes ofrecer tus propias habilidades
- **404**: Habilidad solicitada no existe o no está activa
- **401**: No autenticado

Ejemplo PowerShell:

```
$body = @{
  habilidad_ofrecida_id = 16
  habilidad_solicitada_id = 3
  mensaje_propuesta = "Hola! Me interesa tu servicio"
} | ConvertTo-Json

$propuesta = Invoke-WebRequest -Uri "https://render-test-php-1.onrender.com/api.php?resource=intercambios" `
  -Method POST `
  -Headers @{"Content-Type"="application/json"} `
  -Body $body `
  -WebSession $session
```

4. ☒ Aceptar/Rechazar/Cancelar Intercambio

```
PUT /api.php?resource=intercambios/{id}
Content-Type: application/json
```

Body:

```
{
  "estado": "aceptado"
}
```

Estados válidos:

- **aceptado**: El receptor acepta la propuesta
- **rechazado**: El receptor rechaza la propuesta
- **cancelado**: El proponente cancela su propuesta

Permisos:

- Solo el **receptor** puede aceptar o rechazar
- Solo el **proponente** puede cancelar
- Solo se pueden modificar intercambios en estado **propuesto**

Respuesta exitosa (200):

```
{
  "success": true,
  "message": "Intercambio actualizado exitosamente",
  "data": {
    "mensaje": "Intercambio actualizado exitosamente",
    "nuevo_estado": "aceptado"
  }
}
```

Lógica automática:

- Actualiza el estado del intercambio
- Crea notificación para el otro usuario
- Registra fecha de actualización

Errores posibles:

- **400**: Estado inválido o intercambio no está en estado **propuesto**
- **403**: No tienes permisos para modificar este intercambio
- **404**: Intercambio no encontrado

Ejemplo PowerShell:

```
$body = @{ estado = "aceptado" } | ConvertTo-Json
$result = Invoke-RestMethod -Uri "https://render-test-php-1.onrender.com/api.php?resource=intercambios/7" `
    -Method PUT `
```

```
-Headers @{ "Content-Type"="application/json" } `
-Body $body `
-WebSession $session
```

5. ✓ Marcar Intercambio como Completado

```
PUT /api.php?resource=intercambios/{id}/completar
```

Requisitos:

- El intercambio debe estar en estado **aceptado**
- Solo los participantes (proponente o receptor) pueden marcarlo como completado
- No requiere body en la petición

Respuesta exitosa (200):

```
{
  "success": true,
  "message": "Intercambio marcado como completado. Ahora puedes dejar una valoración.",
  "data": {
    "mensaje": "Intercambio marcado como completado. Ahora puedes dejar una valoración."
  }
}
```

Lógica automática:

- Cambia estado a **completado**
- Registra **fecha_completado**
- Crea notificación para el otro usuario
- **Desbloquea la posibilidad de valorar** al otro usuario

Errores posibles:

- **400**: Solo se pueden completar intercambios aceptados
- **403**: No tienes acceso a este intercambio
- **404**: Intercambio no encontrado

Ejemplo PowerShell:

```
$result = Invoke-RestMethod -Uri "https://render-test-php-1.onrender.com/api.php?resource=intercambios/7/completar" `
-Method PUT `
-WebSession $session
```

💬 CONVERSACIONES Y MENSAJES

6. 📋 Listar Mis Conversaciones 🔒

```
GET /api.php?resource=conversaciones
```

Respuesta exitosa (200):

```
{
  "success": true,
  "message": "OK",
  "data": [
    {
      "id": 1,
      "intercambio_id": 5,
      "fecha_creacion": "2025-10-10 15:20:00+02",
      "ultima_actualizacion": "2025-10-13 18:45:00+02",
      "otro_usuario_id": 3,
      "otro_usuario_nombre": "usuario2",
      "otro_usuario_foto": null,
      "ultimo_mensaje": "Perfecto! Nos vemos el sábado",
      "ultimo_mensaje_fecha": "2025-10-13 18:45:00+02",
      "mensajes_no_leidos": 2
    }
  ]
}
```

Características:

- Lista todas las conversaciones donde el usuario es participante
- Ordena por **ultima_actualizacion** (conversaciones más recientes primero)
- Incluye contador de mensajes no leídos
- Muestra preview del último mensaje
- Identifica automáticamente al "otro usuario" de la conversación

Ejemplo PowerShell:

```
$conversaciones = Invoke-RestMethod -Uri "https://render-test-php-1.onrender.com/api.php?resource=conversaciones" `
-Method GET `
-WebSession $session
```


7. 💬 Obtener Mensajes de una Conversación 🔒

GET /api.php?resource=conversaciones/{id}/mensajes

Parámetros URL:

- **id**: ID de la conversación

Respuesta exitosa (200):

```
{
  "success": true,
  "message": "OK",
  "data": [
    {
      "id": 1,
      "conversacion_id": 1,
      "emisor_id": 2,
      "emisor_nombre": "usuario1",
      "emisor_foto": null,
      "contenido": "Hola! Me interesa tu propuesta",
      "fecha_envio": "2025-10-10 15:20:00+02",
      "leido": true,
      "fecha_lectura": "2025-10-10 16:05:00+02"
    },
    {
      "id": 2,
      "conversacion_id": 1,
      "emisor_id": 3,
      "emisor_nombre": "usuario2",
      "emisor_foto": null,
      "contenido": "Perfecto! Podemos coordinar para el sábado",
      "fecha_envio": "2025-10-13 18:45:00+02",
      "leido": false,
      "fecha_lectura": null
    }
  ]
}
```

Características:

- Ordena mensajes por fecha de envío (más antiguos primero)
- Incluye información del emisor de cada mensaje
- Muestra estado de lectura de cada mensaje
- Solo puedes ver mensajes de conversaciones donde eres participante

Errores posibles:

- 403: No tienes acceso a esta conversación
- 404: Conversación no encontrada

Ejemplo PowerShell:

```
$mensajes = Invoke-RestMethod -Uri "https://render-test-php-1.onrender.com/api.php?resource=conversaciones/1/mensajes" `
-Method GET `
-WebSession $session
```

8. ✦ Crear Nueva Conversación 🔒

```
POST /api.php?resource=conversaciones
Content-Type: application/json
```

Body:

```
{
  "receptor_id": 3,
  "mensaje_inicial": "Hola! Me interesa tu habilidad de clases de inglés.
¿Podríamos coordinar?"
}
```

Campos aceptados:

- **receptor_id** o **otro_usuario_id**: ID del usuario con quien iniciar conversación
- **mensaje_inicial**: Primer mensaje de la conversación

Validaciones:

- No puedes crear conversación contigo mismo
- El receptor debe existir y estar activo
- Si ya existe una conversación entre estos usuarios, envía el mensaje ahí

Respuesta exitosa (201):

```
{
  "success": true,
  "message": "Conversación creada exitosamente",
  "data": {
    "conversacion_id": 10,
    "mensaje": "Conversación creada exitosamente"
  }
}
```

Respuesta si conversación ya existe (200):

```
{
  "success": true,
  "message": "Mensaje enviado en conversación existente",
  "data": {
    "conversacion_id": 5,
    "mensaje": "Mensaje enviado en conversación existente"
  }
}
```

Lógica automática:

- Detecta conversaciones duplicadas
- Crea la conversación y añade ambos participantes
- Envía el mensaje inicial
- Usa transacciones para garantizar consistencia
- Actualiza `ultima_actualizacion` de la conversación

Errores posibles:

- `400`: Campos faltantes o inválidos, o intento de conversación consigo mismo
- `404`: Receptor no existe o no está activo

Ejemplo PowerShell:

```
$body = @{
    receptor_id = 2
    mensaje_inicial = "Hola! Me interesa tu habilidad"
} | ConvertTo-Json

$nuevaConv = Invoke-WebRequest -Uri "https://render-test-php-1.onrender.com/api.php?resource=conversaciones" `
    -Method POST `
    -Headers @{"Content-Type"="application/json"} `
    -Body $body `
    -WebSession $session
```

9. Enviar Mensaje en Conversación Existente

```
POST /api.php?resource=conversaciones/{id}/mensaje
Content-Type: application/json
```

Parámetros URL:

- **id**: ID de la conversación

Body:

```
{
  "contenido": "Perfecto! ¿Te viene bien el sábado a las 10:00?"
}
```

Validaciones:

- Debes ser participante de la conversación
- El contenido no puede estar vacío

Respuesta exitosa (201):

```
{
  "success": true,
  "message": "Mensaje enviado exitosamente",
  "data": {
    "mensaje_id": 25,
    "mensaje": "Mensaje enviado exitosamente"
  }
}
```

Lógica automática:

- Envía el mensaje
- Marca como no leído (**leído** = **false**)
- Actualiza **ultima_actualizacion** de la conversación
- El mensaje aparecerá inmediatamente en GET /mensajes

Errores posibles:

- **400**: Contenido vacío
- **403**: No eres participante de esta conversación
- **404**: Conversación no encontrada

Ejemplo PowerShell:

```
$body = @{ contenido = "Perfecto! Nos vemos entonces" } | ConvertTo-Json
$mensaje = Invoke-WebRequest -Uri "https://render-test-php-1.onrender.com/api.php?resource=conversaciones/1/mensaje" `
  -Method POST `
  -Headers @{ "Content-Type"="application/json" } `
```

```
-Body $body `
-WebSession $session
```

10. ☒ Marcar Mensajes como Leídos

```
PUT /api.php?resource=conversaciones/{id}/marcar-leido
```

Parámetros URL:

- **id**: ID de la conversación

Respuesta exitosa (200):

```
{
  "success": true,
  "message": "Mensajes marcados como leídos",
  "data": {
    "mensaje": "Mensajes marcados como leídos"
  }
}
```

Lógica:

- Marca como leídos **todos los mensajes no leídos** de la conversación que **NO fueron enviados por ti**
- Actualiza **leido = true** y **fecha_lectura = NOW()**
- Esto permite implementar la funcionalidad de "mensajes no leídos" en el frontend

Errores posibles:

- **403**: No eres participante de esta conversación
- **404**: Conversación no encontrada

Ejemplo PowerShell:

```
$result = Invoke-RestMethod -Uri "https://render-test-php-
1.onrender.com/api.php?resource=conversaciones/1/marcar-leido" `
-Method PUT `
-WebSession $session
```

Resumen de Testing (13-14 Oct 2025)

☒ Endpoints Probados y Funcionando

- **GET /intercambios** - Lista intercambios ☒
- **POST /intercambios** - Propone intercambio ☒
- **GET /intercambios/:id** - Detalle de intercambio ☒ (arreglado telefono→ubicacion)
- **PUT /intercambios/:id** - Cancelar intercambio ☒
- **PUT /intercambios/:id** - Validación de permisos (aceptar/rechazar) ☒
- **PUT /intercambios/:id/completar** - Validación (solo si aceptado) ☒
- **GET /conversaciones** - Lista conversaciones ☒
- **GET /habilidades** - Lista habilidades ☒
- **POST /habilidades** - Crea habilidad ☒
- **GET /categorias** - Lista categorías ☒
- **POST /auth/register** - Registra usuario ☒
- **POST /auth/login** - Login ☒

Bugs Encontrados y Arreglados

1. **GET /intercambios/:id** - Error 500 (ARREGLADO ☒)

- **Problema:** Query SQL usaba columna **telefono** que no existe
- **Solución:** Cambiado a **ubicacion** en líneas 153 y 159
- **Commit:** 28efc15 - 14 Oct 2025
- **Status:** ☒ VERIFICADO - Funciona correctamente

2. **POST /conversaciones** - Error 400 (PARCIALMENTE ARREGLADO ☐)

- **Problema 1:** Endpoint esperaba **receptor_id** pero test usaba **otro_usuario_id**
- **Solución 1:** Aceptar ambos nombres + validar que receptor existe
- **Commit:** 28efc15 - 14 Oct 2025
- **Problema 2:** INSERT no incluía campo **intercambio_id** explícitamente
- **Solución 2:** Añadido **intercambio_id = NULL** en INSERT
- **Commit:** fe343d6 - 14 Oct 2025
- **Status:** ☐ PENDIENTE VERIFICACIÓN - Error persiste en transacción SQL

☐ Limitaciones de Testing Encontradas

Problema con usuarios seed: Los usuarios pre-cargados en la base de datos (IDs 1-5) tienen passwords con hash incompatible, por lo que no es posible autenticarse con ellos para testing completo. Esto afecta a:

1. **Aceptar/Rechazar intercambios:**

- ☒ Validación de permisos funciona (solo receptor puede aceptar)
- ☐ No se puede probar aceptación completa (todos los intercambios de prueba tienen a mariaglez ID:2 como receptor)

2. **Completar intercambios:**

- ☒ Validación funciona (solo si está en estado "aceptado")
- ☐ No se puede probar completación exitosa (requiere intercambio aceptado)

3. Endpoints de Conversaciones:

- ☒ GET /conversaciones funciona (devuelve array vacío)
- ☒ POST /conversaciones falla con error en transacción SQL (investigando)
- ☐ GET /conversaciones/:id/mensajes - No probado (requiere conversación existente)
- ☐ POST /conversaciones/:id/mensaje - No probado (requiere conversación existente)
- ☐ PUT /conversaciones/:id/marcar-leido - No probado (requiere conversación existente)

Recomendaciones para Testing Completo

1. **Regenerar hashes de passwords** de usuarios seed con bcrypt correcto
2. **Crear script de testing** que registre 2+ usuarios y pruebe flujo completo:
 - Usuario A crea habilidad
 - Usuario B propone intercambio
 - Usuario A acepta
 - Usuario A completa
 - Usuario B valora
3. **Debugging de POST /conversaciones:** Revisar logs de PostgreSQL en Supabase para identificar causa exacta del error en transacción

Códigos de Respuesta

Código	Significado	Cuándo se usa
200	OK	Operación exitosa (GET, PUT)
201	Created	Recurso creado exitosamente (POST)
400	Bad Request	Datos inválidos o faltantes
401	Unauthorized	No autenticado
403	Forbidden	No tienes permisos
404	Not Found	Recurso no encontrado
500	Internal Server Error	Error del servidor

Estructura de Respuestas

Respuesta Exitosa

```
{
  "success": true,
  "message": "Mensaje descriptivo",
  "data": { /* datos del recurso */ }
}
```

Respuesta de Error

```
{
  "success": false,
  "message": "Descripción del error",
  "error": "Detalles técnicos (solo en desarrollo)"
}
```

Estructura del Código Backend

```
backend/
├── api/
│   ├── index.php           # Router principal
│   ├── auth.php           # Autenticación
│   ├── usuarios.php       # Gestión de usuarios
│   ├── habilidades.php    # CRUD habilidades
│   ├── intercambios.php   #  CRUD intercambios (13 Oct 2025)
│   ├── conversaciones.php #  CRUD conversaciones (13 Oct 2025)
│   ├── valoraciones.php   # Valoraciones
│   ├── notificaciones.php # Notificaciones
│   ├── categorias.php     # Categorías
│   └── reportes.php       # Reportes admin
├── config/
│   ├── database.php       # Conexión PostgreSQL/Supabase
│   └── cors.php           # Configuración CORS
└── utils/
    ├── Auth.php          # Clase de autenticación
    ├── Response.php       # Respuestas estandarizadas
    └── EmailService.php   # Envío de emails
```

Notas de Implementación

Sesiones PHP

- El sistema usa sesiones PHP para mantener autenticación
- La sesión se almacena con una cookie `PHPSESSID`
- El `user_id` se guarda en `$_SESSION['user_id']`

Transacciones

Los siguientes endpoints usan transacciones para garantizar consistencia:

- `POST /intercambios` - Crea intercambio + notificación
- `POST /conversaciones` - Crea conversación + participantes + mensaje
- `PUT /intercambios/:id` - Actualiza estado + crea notificación

Notificaciones Automáticas

Se crean automáticamente en:

- Nueva propuesta de intercambio → Notifica al receptor
- Intercambio aceptado/rechazado → Notifica al proponente
- Intercambio completado → Notifica al otro usuario
- Nuevo mensaje → (Pendiente implementar)

Validaciones de Negocio

- **Intercambios:** Solo puedes ofrecer habilidades tuyas y activas
- **Conversaciones:** No puedes crear conversación contigo mismo
- **Acceso:** Solo puedes ver/modificar tus propios recursos

Deploy en Render

URL Producción: <https://render-test-php-1.onrender.com>

Auto-deploy: Activado desde branch `main`

Última actualización: 14 de Octubre de 2025, 00:15 CEST

Información del Proyecto

TFM - Universitat Oberta de Catalunya (UOC)

Máster: Desarrollo de Sitios y Aplicaciones Web


Alumno: Toni Kampos

Email: toni.vendecasa@gmail.com

GitHub: <https://github.com/tonikampos/render-test-php>

Historial de Cambios

14 de Octubre de 2025

- ☒ Implementación completa de `intercambios.php` (524 líneas)
- ☒ Implementación completa de `conversaciones.php` (393 líneas)
- ☒ Testing de endpoints principales
- 🐛 Bug fix: Query SQL en `GET /intercambios/:id`
- 🐛 Bug fix: Validación en `POST /conversaciones`
- ☒ Deploy en Render.com
-  Documentación API completa

13 de Octubre de 2025

- ☒ Creación del esquema completo en Supabase (14 tablas)
- ☒ Carga de datos seed (5 usuarios, 15 habilidades, 5 intercambios, etc.)

- ☒ Implementación inicial de endpoints

Fin de la documentación 