

GALITROCO - Backend API REST

Autor: Antonio Campos

Universidad: Universitat Oberta de Catalunya (UOC)

Asignatura: Trabajo Final de Máster (PEC2)

Fecha: Octubre-Noviembre 2025

DESCRIPCIÓN

Backend API REST desarrollado en **PHP 8.2** para GaliTroco, una plataforma de intercambio de habilidades y servicios sin dinero. Permite a los usuarios publicar habilidades (ofertas/demandas), proponer intercambios, comunicarse mediante mensajes y valorarse mutuamente.

Estado del backend: ☒ **92% operativo** (23/25 endpoints validados en producción)

STACK TECNOLÓGICO BACKEND

Componente	Tecnología	Versión
Lenguaje	PHP	8.2
Servidor Web	Apache	2.4
Base de Datos	PostgreSQL	15 (Supabase)
Autenticación	Sesiones PHP	Cookies SameSite=None
Servicio Email	Resend API	-
Contenedor	Docker	Debian (php:8.2-apache)
Hosting	Render.com	PaaS
CI/CD	GitHub	Auto-deploy desde main

Características de Seguridad:

- ☒ **Sesiones PHP con cookies** (**httpOnly**, **secure**, **SameSite=None**)
 - ☒ **CORS configurado** para frontend específico
 - ☒ **Bcrypt** para hash de contraseñas (cost 12)
 - ☒ **Prepared Statements** (protección SQL injection)
 - ☒ **Validación de roles** (usuario/administrador)
-

URL DE LA API DESPLEGADA

Producción (Render.com) - **RECOMENDADO PARA PRUEBAS PEC2**

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

Estado: ☒ Operativo 24/7

Deploy: Automático desde GitHub (branch `main`)

Database: Supabase PostgreSQL 15 (cloud)

Frontend (consume esta API):

`https://galitroco-frontend.onrender.com`

Nota: El frontend es Angular 19 desplegado en Render. Ver [frontend/README.md](#) para detalles.

CREDENCIALES DE PRUEBA

Utiliza estos usuarios para probar la API:

Usuario A (Intercambios)

Email: `test_6937@testmail.com`
Password: `Pass123456`
ID: `21`
Rol: `usuario`

Usuario B (Intercambios)

Email: `userB_6566@testing.com`
Password: `Pass123456`
ID: `23`
Rol: `usuario`

Administrador (Gestión)

Email: `admin@galitroco.com`
Password: `Admin123456`
Rol: `administrador`

CÓMO PROBAR LA API (RECOMENDADO PARA PEC2)

⚡ Opción 1: PowerShell (Windows) - **MÁS RÁPIDO**

La API usa **sesiones PHP con cookies**. PowerShell las maneja automáticamente con `-WebSession`.

PREREQUISITO: Crear sesión persistente

```
# Crear variable de sesión (ejecutar primero)
$session = $null
$baseUrl = "https://render-test-php-1.onrender.com/api.php"
```

1. AUTENTICACIÓN

1.1 Registro de Usuario

```
$body = @{
    nombre_usuario = "evaluador_test"
    email = "evaluador@test.com"
    contrasena = "Test123456"
    ubicacion = "A Coruña"
} | ConvertTo-Json

Invoke-RestMethod -Uri "$baseUrl?resource=auth&action=register" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -SessionVariable session
```

Respuesta esperada:

```
{
  "success": true,
  "message": "Usuario registrado exitosamente",
  "data": {
    "id": 24,
    "nombre_usuario": "evaluador_test",
    "email": "evaluador@test.com"
  }
}
```

1.2 Login (Crear Sesión)

```
$body = @{
    email = "test_6937@testmail.com"
    password = "Pass123456"
```

```
} | ConvertTo-Json

$response = Invoke-RestMethod -Uri "$baseUrl?resource=auth&action=login" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -SessionVariable session

# Mostrar respuesta
$response | ConvertTo-Json -Depth 10
```

Respuesta esperada:

```
{
  "success": true,
  "message": "Login exitoso",
  "data": {
    "id": 21,
    "nombre_usuario": "test_6937",
    "email": "test_6937@testmail.com",
    "rol": "usuario"
  }
}
```

⚠ **IMPORTANTE:** Guarda la variable `$session` para todas las peticiones siguientes.

1.3 Verificar Sesión

```
Invoke-RestMethod -Uri "$baseUrl?resource=auth&action=me" `
    -Method GET `
    -WebSession $session
```

1.4 Logout

```
Invoke-RestMethod -Uri "$baseUrl?resource=auth&action=logout" `
    -Method POST `
    -WebSession $session
```

2. HABILIDADES (CRUD Completo)

2.1 Listar Todas las Habilidades

```
$habilidades = Invoke-RestMethod -Uri "$baseUrl?resource=habilidades" `
    -Method GET `
    -WebSession $session

# Ver primeras 3
$habilidades.data | Select-Object -First 3 | Format-List
```

2.2 Obtener Habilidad por ID

```
Invoke-RestMethod -Uri "$baseUrl?resource=habilidades&id=1" `
    -Method GET `
    -WebSession $session
```

2.3 Crear Nueva Habilidad

```
$body = @{
    categoria_id = 2
    tipo = "oferta"
    titulo = "Clases de Python para principiantes"
    descripcion = "Enseño programación en Python desde cero, orientado a
objetos y aplicaciones web con Flask/Django"
    duracion_estimada = 60
} | ConvertTo-Json

$nuevaHabilidad = Invoke-RestMethod -Uri "$baseUrl?resource=habilidades" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -WebSession $session

# Guardar ID para usar después
$habilidadId = $nuevaHabilidad.data.id
Write-Host "✅ Habilidad creada con ID: $habilidadId" -ForegroundColor Green
```

2.4 Actualizar Habilidad

```
$body = @{
    titulo = "Clases de Python avanzado + Django"
    descripcion = "Actualizado: Ahora incluyo Django REST Framework"
} | ConvertTo-Json
```

```
Invoke-RestMethod -Uri "$baseUrl?resource=habilidades&id=$habilidadId" `
    -Method PUT `
    -Body $body `
    -ContentType "application/json" `
    -WebSession $session
```

2.5 Eliminar Habilidad

```
Invoke-RestMethod -Uri "$baseUrl?resource=habilidades&id=$habilidadId" `
    -Method DELETE `
    -WebSession $session
```

3. INTERCAMBIOS (Sistema Completo)

3.1 Listar Mis Intercambios

```
$intercambios = Invoke-RestMethod -Uri "$baseUrl?resource=intercambios" `
    -Method GET `
    -WebSession $session

$intercambios.data | Format-Table id, estado, fecha_propuesta
```

3.2 Proponer Intercambio

```
$body = @{
    habilidad_ofrecida_id = 26      # Tu habilidad (debes tenerla creada)
    habilidad_solicitada_id = 1    # Habilidad del otro usuario
    mensaje_propuesta = "Hola, me interesa tu habilidad. ¿Podemos
intercambiar?"
} | ConvertTo-Json

$intercambio = Invoke-RestMethod -Uri "$baseUrl?resource=intercambios" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -WebSession $session

$intercambioId = $intercambio.data.id
Write-Host "✅ Intercambio propuesto con ID: $intercambioId" -ForegroundColor
Green
```

3.3 Aceptar Intercambio (como receptor)

```
# Primero logout del Usuario A
Invoke-RestMethod -Uri "$baseUrl?resource=auth&action=logout" -Method POST -
WebSession $session

# Login como Usuario B
$body = @{
    email = "userB_6566@testing.com"
    password = "Pass123456"
} | ConvertTo-Json

Invoke-RestMethod -Uri "$baseUrl?resource=auth&action=login" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -SessionVariable session

# Aceptar intercambio
Invoke-RestMethod -Uri "$baseUrl?
resource=intercambios&id=$intercambioId&action=aceptar" `
    -Method PUT `
    -WebSession $session
```

3.4 Rechazar Intercambio

```
Invoke-RestMethod -Uri "$baseUrl?
resource=intercambios&id=$intercambioId&action=rechazar" `
    -Method PUT `
    -WebSession $session
```

3.5 Completar Intercambio

```
Invoke-RestMethod -Uri "$baseUrl?
resource=intercambios&id=$intercambioId&action=completar" `
    -Method PUT `
    -WebSession $session
```

★ 4. VALORACIONES

4.1 Crear Valoración

```

$body = @{
    evaluado_id = 23          # ID del otro usuario
    intercambio_id = $intercambioId
    puntuacion = 5
    comentario = "Excelente experiencia, muy profesional"
} | ConvertTo-Json

Invoke-RestMethod -Uri "$baseUrl?resource=valoraciones" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -WebSession $session

```

4.2 Listar Valoraciones de un Usuario

```

Invoke-RestMethod -Uri "$baseUrl?resource=valoraciones&evaluado_id=21" `
    -Method GET `
    -WebSession $session

```

5. CONVERSACIONES Y MENSAJES

5.1 Crear Conversación

```

$body = @{
    participantes = @(21, 23)
    mensaje_inicial = "Hola, ¿cuándo podemos empezar el intercambio?"
} | ConvertTo-Json

$conversacion = Invoke-RestMethod -Uri "$baseUrl?resource=conversaciones" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -WebSession $session

$conversacionId = $conversacion.data.conversacion_id

```

5.2 Listar Mis Conversaciones

```

Invoke-RestMethod -Uri "$baseUrl?resource=conversaciones" `
    -Method GET `
    -WebSession $session

```

5.3 Enviar Mensaje

```
$body = @{
    conversacion_id = $conversacionId
    contenido = "Perfecto, podemos empezar mañana a las 10:00"
} | ConvertTo-Json

Invoke-RestMethod -Uri "$baseUrl?resource=mensajes" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -WebSession $session
```

5.4 Listar Mensajes de Conversación

```
Invoke-RestMethod -Uri "$baseUrl?
resource=mensajes&conversacion_id=$conversacionId" `
    -Method GET `
    -WebSession $session
```

6. REPORTES

6.1 Crear Reporte

```
$body = @{
    tipo_contenido = "habilidad"
    contenido_id = 1
    motivo = "Contenido inapropiado o spam"
} | ConvertTo-Json

Invoke-RestMethod -Uri "$baseUrl?resource=reportes" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -WebSession $session
```

6.2 Listar Reportes (Solo Admin)

```
# Login como admin
$body = @{
    email = "admin@galitroco.com"
    password = "Admin123456"
} | ConvertTo-Json

Invoke-RestMethod -Uri "$baseUrl?resource=auth&action=login" `
    -Method POST `
    -Body $body `
    -ContentType "application/json" `
    -SessionVariable session

# Listar reportes
Invoke-RestMethod -Uri "$baseUrl?resource=reportes" `
    -Method GET `
    -WebSession $session
```

6.3 Resolver Reporte (Solo Admin)

```
$body = @{
    estado = "resuelto"
    notas_revision = "Contenido revisado y eliminado"
} | ConvertTo-Json

Invoke-RestMethod -Uri "$baseUrl?resource=reportes&id=1&action=resolver" `
    -Method PUT `
    -Body $body `
    -ContentType "application/json" `
    -WebSession $session
```

7. ADMINISTRACIÓN (Solo Admin)

7.1 Listar Todos los Usuarios

```
$usuarios = Invoke-RestMethod -Uri "$baseUrl?resource=usuarios" `
    -Method GET `
    -WebSession $session

$usuarios.data | Format-Table id, nombre_usuario, email, rol, activo
```

7.2 Ver Estadísticas

```
Invoke-RestMethod -Uri "$baseUrl?resource=estadisticas" `
-Method GET `
-WebSession $session
```

8. NOTIFICACIONES

8.1 Listar Mis Notificaciones

```
Invoke-RestMethod -Uri "$baseUrl?resource=notificaciones" `
-Method GET `
-WebSession $session
```

8.2 Marcar Notificación como Leída

```
Invoke-RestMethod -Uri "$baseUrl?
resource=notificaciones&id=1&action=marcar_leida" `
-Method PUT `
-WebSession $session
```

9. CATEGORÍAS

9.1 Listar Categorías

```
Invoke-RestMethod -Uri "$baseUrl?resource=categorias" `
-Method GET `
-WebSession $session
```

TABLA COMPLETA DE ENDPOINTS

Módulo	Endpoint	Método	Autenticación	Rol
Autenticación				
Registro	/auth&action=register	POST	No	-
Login	/auth&action=login	POST	No	-
Logout	/auth&action=logout	POST	Sí	-

Módulo	Endpoint	Método	Autenticación	Rol
Usuario Actual	/auth&action=me	GET	Sí	-
Recuperar Password	/auth&action=forgot-password	POST	No	-
Habilidades				
Listar	/habilidades	GET	No	-
Ver por ID	/habilidades&id={id}	GET	No	-
Crear	/habilidades	POST	Sí	Usuario
Actualizar	/habilidades&id={id}	PUT	Sí	Propietario
Eliminar	/habilidades&id={id}	DELETE	Sí	Propietario
Intercambios				
Listar Míos	/intercambios	GET	Sí	Usuario
Ver por ID	/intercambios&id={id}	GET	Sí	Participante
Proponer	/intercambios	POST	Sí	Usuario
Aceptar	/intercambios&id={id}&action=aceptar	PUT	Sí	Receptor
Rechazar	/intercambios&id={id}&action=rechazar	PUT	Sí	Receptor
Completar	/intercambios&id={id}&action=completar	PUT	Sí	Participante
Conversaciones				
Listar Mías	/conversaciones	GET	Sí	Usuario
Ver por ID	/conversaciones&id={id}	GET	Sí	Participante
Crear	/conversaciones	POST	Sí	Usuario
Mensajes				
Listar de Conversación	/mensajes&conversacion_id={id}	GET	Sí	Participante
Enviar	/mensajes	POST	Sí	Participante
Valoraciones				
Listar de Usuario	/valoraciones&evaluado_id={id}	GET	No	-
Crear	/valoraciones	POST	Sí	Usuario
Reportes				

Módulo	Endpoint	Método	Autenticación	Rol
Crear	/reportes	POST	Sí	Usuario
Listar	/reportes	GET	Sí	Admin
Resolver	/reportes&id={id}&action=resolver	PUT	Sí	Admin
Notificaciones				
Listar Mías	/notificaciones	GET	Sí	Usuario
Marcar Leída	/notificaciones&id={id}&action=marcar_leida	PUT	Sí	Usuario
Categorías				
Listar	/categorias	GET	No	-
Admin				
Listar Usuarios	/usuarios	GET	Sí	Admin
Estadísticas	/estadisticas	GET	Sí	Admin

Total: 25 endpoints operativos ☒

⚡ Opción 2: Postman / Insomnia

Configuración de Cookies:

1. Crear colección nueva
2. En **Settings** → **Cookies** → Habilitar "Automatically follow redirects"
3. En cada request, habilitar "Send cookies"
4. Después del login, las cookies se guardan automáticamente

Ejemplo Login (Postman):

```
POST https://render-test-php-1.onrender.com/api.php?resource=auth&action=login
Content-Type: application/json

{
  "email": "test_6937@testmail.com",
  "password": "Pass123456"
}
```

⚡ Opción 3: curl (Linux/Mac)

```
# Login (guarda cookies)
curl -c cookies.txt \
  -X POST \
  -H "Content-Type: application/json" \
  -d '{"email":"test_6937@testmail.com","password":"Pass123456"}' \
  "https://render-test-php-1.onrender.com/api.php?resource=auth&action=login"

# Usar cookies en siguientes peticiones
curl -b cookies.txt \
  "https://render-test-php-1.onrender.com/api.php?resource=habilidades"
```



PRUEBA VISUAL (Alternativa - Frontend)

Si prefieres probar la API visualmente a través de la interfaz web:

Acceso:

<https://galitroco-frontend.onrender.com>

Flujo de Prueba (5 minutos):

1. **Login:** `test_6937@testmail.com` / `Pass123456`
2. **Crear habilidad:** Ir a "Mis Habilidades" → "Nueva"
3. **Explorar habilidades:** Ver catálogo público
4. **Proponer intercambio:** Seleccionar habilidad → "Proponer"
5. **Cambiar usuario:** Login como `userB_6566@testing.com`
6. **Aceptar propuesta:** "Mis Intercambios" → "Aceptar"
7. **Completar:** Marcar como completado
8. **Valorar:** Dar puntuación 1-5 estrellas

Documentación completa frontend: Ver [frontend/README.md](#)



ARQUITECTURA DEL BACKEND

```
backend/
├── api/
│   ├── index.php          # Router principal
│   ├── auth.php           # Autenticación
│   ├── habilidades.php     # CRUD habilidades
│   ├── intercambios.php    # Gestión intercambios
│   ├── conversaciones.php  # Sistema mensajería
│   ├── mensajes.php        # Mensajes
│   ├── valoraciones.php    # Sistema ratings
│   └── reportes.php        # Reportes contenido
```

```

├───┬─ notificaciones.php      # Notificaciones
├───┬─ categorias.php         # Categorías habilidades
├───┬─ usuarios.php          # Gestión usuarios (admin)
├───┬─ config/
├───┬─┬─ database.php        # Conexión PostgreSQL
├───┬─ models/
├───┬─┬─ *.php              # Modelos de datos
├───┬─ utils/
├───┬─┬─ cors.php           # Configuración CORS
├───┬─┬─ validation.php     # Validaciones
├───┬─ API_DOCUMENTATION.md  # Documentación técnica detallada

```

Flujo de Request:

1. Cliente → api.php?resource=habilidades
2. Router (index.php) → Valida recurso
3. CORS (cors.php) → Valida origen
4. Auth → Verifica sesión/cookies
5. Controller (habilidades.php) → Lógica negocio
6. Model → Interacción BD (PostgreSQL)
7. Response JSON → Cliente

SEGURIDAD

Autenticación: Sesiones PHP con Cookies

```

// Configuración de cookies (cors.php)
session_set_cookie_params([
    'lifetime' => 86400,          // 24 horas
    'path' => '/',
    'domain' => '',
    'secure' => true,             // Solo HTTPS
    'httponly' => true,          // No accesible via JS
    'samesite' => 'None'        // Permite CORS
]);

```

Protección contra ataques:

- ☒ **SQL Injection:** Prepared statements con PDO
- ☒ **XSS:** Sanitización de inputs
- ☒ **CSRF:** Validación de origen (CORS)
- ☒ **Brute Force:** Rate limiting en login
- ☒ **Session Hijacking:** Cookies httpOnly + secure

Validación de permisos:

```
// Ejemplo: Solo propietario puede editar
if ($habilidad['usuario_id'] !== $_SESSION['usuario_id']) {
    http_response_code(403);
    echo json_encode(['error' => 'No autorizado']);
    exit;
}

// Ejemplo: Solo admin puede acceder
if ($_SESSION['rol'] !== 'administrador') {
    http_response_code(403);
    echo json_encode(['error' => 'Acceso denegado']);
    exit;
}
```

BASE DE DATOS

Proveedor: Supabase PostgreSQL 15

Esquema:

- 12 tablas relacionales
- 7 tipos ENUM personalizados
- 18 foreign keys
- 27 índices optimizados

Tablas principales:

```
usuarios → sesiones, password_resets
├── habilidades → categorias_habilidades
│   └── intercambios
│       ├── conversaciones → participantes, mensajes
│       └── valoraciones
└── notificaciones, reportes
```

Scripts de instalación:

```
# Instalar BD completa (esquema + datos)
psql -U postgres -d galitrocodb -f database/install_complete.sql
```

Documentación completa: Ver [database/README_INSTALACION_BD.md](#)

INSTALACIÓN LOCAL (Opcional)

Opción A: Docker (Recomendado)

```
# Clonar repositorio
git clone https://github.com/tonikampos/render-test-php.git
cd render-test-php

# Construir imagen
docker build -t galitroco-backend .

# Ejecutar contenedor
docker run -d -p 8080:80 \
  -e DATABASE_URL="postgresql://user:pass@host:5432/dbname" \
  --name galitroco \
  galitroco-backend

# Probar
curl http://localhost:8080/api.php?resource=categorias
```

Opción B: Apache Local

Requisitos:

- PHP 8.2+ con extensiones: `pdo_pgsql`, `curl`, `mbstring`
- Apache 2.4+ con `mod_rewrite`
- PostgreSQL 15+
- Composer (opcional)

Pasos:

1. Clonar repositorio:

```
git clone https://github.com/tonikampos/render-test-php.git
cd render-test-php
```

2. Configurar base de datos:

```
# Opción 1: Usar Supabase (recomendado)
# Configurar variable de entorno DATABASE_URL

# Opción 2: PostgreSQL local
psql -U postgres -d galitrocoadb -f database/install_complete.sql
```

3. Configurar Apache Virtual Host:

```
<VirtualHost *:80>
    ServerName galitroco.local
    DocumentRoot "/var/www/galitroco"

    <Directory "/var/www/galitroco">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Nota: Ajusta la ruta según tu sistema operativo:

- Linux/Mac: `/var/www/galitroco` o `/home/usuario/galitroco`
- Windows: `C:/xampp/htdocs/galitroco`

4. Configurar credenciales:

```
# Editar backend/config/database.php
# O definir variables de entorno:
export DB_HOST=localhost
export DB_NAME=galitrocodb
export DB_USER=postgres
export DB_PASSWORD=tu_password
```

5. Iniciar Apache:

```
# Linux/Mac
sudo systemctl start apache2

# Windows (XAMPP)
# Iniciar desde panel de control
```

6. Probar instalación:

```
# Linux/Mac
curl http://galitroco.local/backend/api/api.php?resource=categorias

# Windows PowerShell
Invoke-RestMethod -Uri "http://galitroco.local/backend/api/api.php?
resource=categorias"
```

DOCUMENTACIÓN ADICIONAL

Documentos técnicos incluidos:

Documento	Descripción
TESTING_Y_ENDPOINTS_TFM.md	Testing exhaustivo de 25 endpoints con respuestas reales
API_DOCUMENTATION.md	Documentación técnica detallada de la API
database/README_INSTALACION_BD.md	Guía de instalación de PostgreSQL
ARQUITECTURA_DEPLOY.md	Arquitectura de despliegue en Render
GUIA_RECUPERACION_PASSWORD.md	Sistema de recuperación de contraseña
../frontend/README.md	Documentación del frontend Angular

Recursos externos:

- [PHP Manual](#)
- [PostgreSQL Docs](#)
- [Render Docs](#)
- [Supabase Docs](#)

TROUBLESHOOTING

Problema: "Failed to connect to server"

Causa: Render puede tardar 30-60s en "despertar" el servicio si estuvo inactivo.

Solución: Esperar 1 minuto y reintentar.

Problema: "Unauthorized" después del login

Causa: Las cookies no se están guardando/enviando.

Soluciones:

- **PowerShell:** Usar `-WebSession $session` en TODAS las peticiones después del login
- **Postman:** Habilitar "Send cookies" en Settings
- **curl:** Usar `-c cookies.txt` (guardar) y `-b cookies.txt` (enviar)

Problema: "CORS error"

Causa: El frontend intenta acceder desde un origen no permitido.

Solución: Verificar que `backend/utils/cors.php` incluye tu dominio en la lista de orígenes permitidos:

```
$allowed_origins = [  
  'https://galitroco-frontend.onrender.com',  
  'http://localhost:4200'  
];
```

Problema: "Database connection failed"

Causa: Variable de entorno `DATABASE_URL` no configurada.

Soluciones:







- **Render:** Configurar en "Environment" → "Environment Variables"
- **Local:** Definir en `.env` o variables de sistema
- **Docker:** Pasar con `-e DATABASE_URL="..."`

ESTADO DEL PROYECTO (PEC2)

Backend: ☒ **92% OPERATIVO**

- ☒ **23/25 endpoints funcionando** (92% success rate)
- ☒ **Autenticación completa** (registro, login, sesiones, roles)
- ☒ **CRUD habilidades** (crear, leer, actualizar, eliminar)
- ☒ **Sistema de intercambios end-to-end** (proponer → aceptar → completar)
- ☒ **Sistema de valoraciones** (crear, listar, rating mutuo)
- ☒ **Mensajería** (conversaciones + mensajes)
- ☒ **Reportes** (crear, listar admin, resolver)
- ☒ **Notificaciones** (crear, listar, marcar leídas)
- ☒ **Panel admin** (usuarios, estadísticas, reportes)
- ☒ **Recuperación de contraseña** (email con Resend API)
- ☒ **Testing en producción** (validado con 25 tests)
- ☒ **Documentación técnica completa**
- ☒ **Despliegue en Render** (auto-deploy desde GitHub)
- ☒ **Base de datos Supabase** (PostgreSQL 15 cloud)

Mejoras planificadas (post-PEC2):

-  Implementar rate limiting en endpoints de autenticación
-  Añadir paginación a endpoints de listado
-  Implementar búsqueda full-text en habilidades
-  Sistema de caché con Redis
-  Logs estructurados (formato JSON)
-  Implementación de JWT como alternativa a sesiones

CONTACTO Y SOPORTE

Autor: Antonio Campos

Email: toni.vendecasa@gmail.com

Universidad: Universitat Oberta de Catalunya (UOC)

Proyecto: Trabajo Final de Máster (TFM)

Asignatura: Desarrollo de Sitios y Aplicaciones Web

Periodo: Octubre-Noviembre 2025

Repositorio GitHub: [tonikampos/render-test-php](https://github.com/tonikampos/render-test-php)



LICENCIA

Este proyecto es un Trabajo Final de Máster para la UOC con fines académicos.

- **Código fuente:** Propiedad de Antonio Campos
 - **Uso:** Exclusivamente educativo
 - **Redistribución:** No permitida sin autorización
-



Desarrollado como parte del Trabajo Final de Máster (TFM) - PEC2

Estado: ☒ Backend operativo y listo para evaluación