# API REST PHP

## Documentación Técnica Completa

Sistema de Gestión de Autonomix de Profesionales

Versión 1.0 - Febrero 2026

## Tabla de Contenidos

## 1. Resumen Ejecutivo

API REST desarrollada en PHP para gestión de perfiles profesionales con arquitectura RESTful, base de datos MySQL y respuestas JSON. Incluye 12 endpoints para operaciones CRUD sobre usuarios, skills y categorías.

| | |
|---|---|
| **Lenguaje** | PHP 7.4+ |
| **Base de Datos** | MySQL 5.7+ |
| **Arquitectura** | REST API |
| **Formato** | JSON (UTF-8) |
| **Endpoints** | 12 endpoints |
| **Tablas** | 5 tablas |

## 2. Introducción

### 2.1 Descripción

API REST que proporciona funcionalidad completa para gestionar perfiles de desarrolladores y profesionales tecnológicos, incluyendo sus habilidades, categorías y datos personales.

### 2.2 Características

- Arquitectura RESTful con métodos HTTP estándar
- Respuestas JSON con codificación UTF-8
- Consultas preparadas (SQL injection prevention)
- Relaciones many-to-many para skills y categorías
- Gestión completa de usuarios (CRUD)
- Sistema modular y extensible

# 3. Arquitectura del Sistema

## 3.1 Estructura de Directorios

```
backend/
└── api/
    ├── conexion.php
    ├── obtener_usuarios.php
    ├── obtener_usuario.php
    ├── obtener_profesionales.php
    ├── obtener_skills.php
    ├── obtener_categorias.php
    ├── registrar_usuario.php
    ├── actualizar_usuario.php
    ├── eliminar_usuario.php
    ├── enviar_contacto.php
    └── reenviar_contactos.php
```

## 3.2 Flujo de Peticiones

1. Cliente → Petición HTTP
2. Endpoint PHP → Validación
3. Conexión MySQL → Consulta
4. Procesamiento → JSON
5. Respuesta → Cliente

## 4. Base de Datos

Base de datos relacional MySQL con 5 tablas principales.

### 4.1 Tabla: users

| Campo | Tipo | Descripción |
|---|---|---|
| **id** | int(11) PK | ID único |
| **nombre** | varchar(100) | Nombre completo |
| **email** | varchar(150) UNIQUE | Email único |
| **password** | varchar(255) | Hash de contraseña |
| **ciudad** | varchar(100) | Ciudad |
| **provincia** | varchar(100) | Provincia |
| **modalidad** | Enum/String | presencial/online/mixto |
| **backend** | Enum/String | Nivel 0-5 |
| **frontend** | Enum/String | Nivel 0-5 |
| **especializacion** | varchar(250) | Título profesional |
| **descripcion** | text | Biografía |
| **enlaces** | varchar(500) | URLs redes sociales |
| **avatar3D** | varchar(255) | URL avatar 3D |
| **avatar2D** | varchar(255) | URL avatar 2D |
| **created_at** | timestamp | Fecha registro |

### 4.2 Tabla: skills

Catálogo de habilidades técnicas.

| Campo | Tipo | Descripción |
|---|---|---|
| **id** | int(11) PK | ID único |
| **nombre** | varchar(100) UNIQUE | Nombre skill |

### 4.3 Tabla: categories

Categorías profesionales.

| Campo | Tipo | Descripción |
|---|---|---|
| **id** | int(11) PK | ID único |
| **nombre** | varchar(50) UNIQUE | Nombre categoría |

### 4.4 Tablas de Relación

user_skills y user_categories implementan relaciones many-to-many.

## 5. API Endpoints

| Endpoint | Método | Descripción |
|---|---|---|
| **/api/obtener_usuarios.php** | GET | Lista todos los usuarios |
| **/api/obtener_usuario.php?id=N** | GET | Obtiene usuario específico |
| **/api/obtener_profesionales.php** | GET | Profesionales completos |
| **/api/obtener_skills.php** | GET | Lista de skills |
| **/api/obtener_categorias.php** | GET | Lista de categorías |
| **/api/registrar_usuario.php** | POST | Crea nuevo usuario |
| **/api/actualizar_usuario.php** | POST | Actualiza usuario |
| **/api/eliminar_usuario.php** | POST | Elimina usuario |
| **/api/enviar_contacto.php** | POST | Procesa contacto |

### 5.1 Obtener Usuario

```
GET /api/obtener_usuario.php?id=20
```

Respuesta:


```
{
  "success": true,
  "data": {
    "id": 20,
    "nombre": "Andrea Collazo",
    "email": "acollazocacho@gmail.com",
    "enlaces": "https://www.linkedin.com/",
    "descripcion": "Programadora Backend, API, Spring, Microservicios",
    "ciudad": "Vigo",
    "provincia": "Pontevedra",
    "modalidad": "online",
    "backend": "4",
    "frontend": "2",
    "avatar3D": null,
    "avatar2D": "uploads/avatars/6983163d1542a_avatar.png",
    "especializacion": "Backend, Bases de datos",
    "skills": [
      "HTML",
      "Java",
      "JavaScript",
      "Laravel",
      "SpringBoot",
      "WordPress"
    ],
    "categorias": [
      "APIs & Integraciones",
      "Backend",
      "Fullstack"
    ]
  }
}
```


### 5.2 Registrar Usuario

```
POST /api/registrar_usuario.php
Content-Type: application/json

{
```

```
  "nombre": "Ana López",
  "email": "ana@example.com",
  "password": "secure123",
  "ciudad": "Barcelona",
  "skills": [1, 2],
  "categorias": [1]
}
```

# 6. Ejemplos de Implementación

## 6.1 JavaScript (Fetch)

```
// Obtener usuarios
async function getUsuarios() {
  const response = await fetch('{BaseURL}/obtener_usuarios.php');
  const data = await response.json();
  if (data.success) {
    console.log(data.data);
  }
}

// Registrar usuario
async function registrar(usuario) {
  const response = await fetch('{BaseURL}/registrar_usuario.php', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(usuario)
  });
  return await response.json();
}
```

## 6.3 cURL

```
# GET
curl {BaseURL}/obtener_usuarios.php

# POST
curl -X POST {BaseURL}/registrar_usuario.php \
  -H "Content-Type: application/json" \
  -d '{"nombre":"Test","email":"test@example.com"}'
```

# 7. Configuración

## 7.1 Requisitos

- PHP 7.4+
- MySQL 5.7+
- Apache/Nginx
- Extensión mysqli

1. Configurar conexion.php

```
<?php
$host = "sql7.freesqldatabase.com";
$user = "sql7815048";
$pass = "F7k37eg2fG";
$db   = "sql7815048";
$port = 3306;


$conn = new mysqli($host, $user, $pass, $db, $port);
?>
```

## 8. Seguridad

- Consultas preparadas (prepared statements)
- Validación de entrada de datos
- Hashing de contraseñas con password_hash() (Sin uso al no haber autenticación)
- Headers de seguridad apropiados
- Codificación UTF-8 en respuestas
- CORS configurado correctamente

Recomendaciones adicionales:

- Implementar autenticación JWT
- Añadir rate limiting
- Usar HTTPS en producción
- Mantener logs de acceso
- Backups automáticos de BD

## 9. Conclusión

Esta API REST en PHP proporciona una solución completa y profesional para la gestión de perfiles de desarrolladores. Con arquitectura RESTful, base de datos optimizada, 12 endpoints funcionales y documentación exhaustiva, el sistema está listo para producción y futuras expansiones.

Características destacadas:

- ✓ 12 endpoints RESTful
- ✓ Base de datos con 5 tablas
- ✓ Relaciones many-to-many
- ✓ Consultas preparadas
- ✓ JSON con UTF-8
- ✓ Ejemplos en 3 lenguajes
- ✓ Documentación completa