

# Library Management API

---

RESTful API para gestión de biblioteca con autenticación JWT, construida con Laravel 5.1, PostgreSQL y Docker.

## Tabla de Contenidos

- [Características](#)
- [Tecnologías](#)
- [Requisitos Previos](#)
- [Instalación](#)
- [Configuración](#)
- [Uso de la API](#)
- [Endpoints](#)
- [Arquitectura](#)
- [Testing](#)

## Características

- Autenticación JWT (JSON Web Tokens)
- CRUD completo de Usuarios, Autores y Libros
- Sistema de Events y Jobs para actualización automática de contadores
- Exportación de datos a Excel (XLSX)
- Soft Deletes con auditoría (created\_by, updated\_by, deleted\_by)
- Validaciones con Form Requests
- Respuestas JSON consistentes
- Arquitectura escalable con Services y Base Classes
- Docker para desarrollo y producción
- PostgreSQL como base de datos

## Tecnologías

- **Backend:** Laravel 5.1
- **Base de Datos:** PostgreSQL 13
- **Autenticación:** JWT (tymon/jwt-auth 0.5)
- **Servidor Web:** Nginx
- **Contenedores:** Docker & Docker Compose
- **PHP:** 7.1-fpm
- **Excel Export:** Maatwebsite Excel 2.1

## Requisitos Previos

- Docker Desktop instalado
- Git
- Cliente HTTP (Postman, Insomnia, o cURL)

## Instalación

## 1. Clonar el repositorio

```
git clone https://github.com/CristobalRodriguezCh/library-management-api.git  
cd library-management-api
```

## 2. Levantar los contenedores Docker

```
docker-compose up -d --build
```

Esto construirá e iniciará los siguientes servicios:

- **app:** Aplicación Laravel (PHP 7.1-FPM)
- **nginx:** Servidor web (puerto 8000)
- **db:** PostgreSQL (puerto 5433)

## 3. Instalar dependencias de Composer

```
docker-compose exec app composer install
```

## 4. Configurar el archivo .env

```
docker-compose exec app cp .env.example .env
```

Verifica que el `.env` tenga estas configuraciones:

```
APP_NAME=LibraryManagementAPI  
APP_ENV=local  
APP_KEY=  
APP_DEBUG=true  
APP_URL=http://localhost:8000
```

```
DB_CONNECTION=pgsql  
DB_HOST=db  
DB_PORT=5432  
DB_DATABASE=library_db  
DB_USERNAME=library_user  
DB_PASSWORD=library_pass
```

```
CACHE_DRIVER=file  
SESSION_DRIVER=file  
QUEUE_DRIVER=sync
```

## 5. Generar la clave de la aplicación

```
docker-compose exec app php artisan key:generate
```

## 6. Generar la clave JWT

```
docker-compose exec app php artisan jwt:generate
```

## 7. Ejecutar las migraciones

```
docker-compose exec app php artisan migrate
```

## 8. Configurar permisos

```
docker-compose exec app chmod -R 775 storage bootstrap/cache  
docker-compose exec app chown -R www-data:www-data storage bootstrap/cache
```

## 9. Verificar instalación

Abre tu navegador en <http://localhost:8000> - Deberías ver la página de bienvenida de Laravel.

## ⚙️ Configuración

### Variables de Entorno

Variable	Descripción	Valor por Defecto
APP_URL	URL de la aplicación	<a href="http://localhost:8000">http://localhost:8000</a>
DB_HOST	Host de PostgreSQL	db
DB_PORT	Puerto de PostgreSQL	5432
DB_DATABASE	Nombre de la BD	library_db
DB_USERNAME	Usuario de BD	library_user
DB_PASSWORD	Contraseña de BD	library_pass

### Puertos Expuestos

- **8000:** Nginx (Aplicación web)
- **5433:** PostgreSQL (Base de datos - puerto externo)

## 📖 Uso de la API

### Base URL

```
http://localhost:8000/api/v1
```

### Autenticación

La API utiliza JWT (JSON Web Tokens) para autenticación. Incluye el token en el header de cada petición:

```
Authorization: Bearer {tu_token_jwt}
```

### Flujo de Autenticación

1. **Registrar usuario:** POST /api/v1/register
2. **Login:** POST /api/v1/login → Obtener token
3. **Usar el token** en todas las peticiones protegidas

## 🔗 Endpoints

### Autenticación

#### Registro de Usuario

```
POST /api/v1/register
Content-Type: application/json

{
  "name": "Juan Pérez",
  "email": "juan@example.com",
  "password": "password123",
  "password_confirmation": "password123",
  "birth_date": "1990-01-15",
  "role": "user"
}
```

#### Respuesta exitosa (201):

```
{
  "success": true,
  "message": "User registered successfully",
  "data": {
    "user": {
      "id": 1,
      "name": "Juan Pérez",
```

```
"email": "juan@example.com",
"birth_date": "1990-01-15",
"role": "user",
"created_at": "2026-01-29 00:00:00",
"updated_at": "2026-01-29 00:00:00"
},
"token": "eyJ0eXAiOiJKV1QiLCJhbGc..."}
```

## Login

```
POST /api/v1/login
Content-Type: application/json

{
  "email": "juan@example.com",
  "password": "password123"
}
```

### Respuesta exitosa (200):

```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "user": {
      "id": 1,
      "name": "Juan Pérez",
      "email": "juan@example.com"
    },
    "token": "eyJ0eXAiOiJKV1QiLCJhbGc..."
  }
}
```

## Obtener Usuario Autenticado

```
GET /api/v1/me
Authorization: Bearer {token}
```

## Logout

```
POST /api/v1/logout
Authorization: Bearer {token}
```

## Usuarios

### Listar Usuarios

```
GET /api/v1/users
Authorization: Bearer {token}
```

### Crear Usuario

```
POST /api/v1/users
Authorization: Bearer {token}
Content-Type: application/json

{
  "name": "María García",
  "email": "maria@example.com",
  "password": "password123",
  "password_confirmation": "password123",
  "birth_date": "1985-05-20",
  "role": "author"
}
```

### Ver Usuario

```
GET /api/v1/users/{id}
Authorization: Bearer {token}
```

### Actualizar Usuario

```
PUT /api/v1/users/{id}
Authorization: Bearer {token}
Content-Type: application/json

{
  "name": "María García Actualizado",
  "email": "maria.nueva@example.com"
}
```

### Eliminar Usuario (Soft Delete)

```
DELETE /api/v1/users/{id}
Authorization: Bearer {token}
```

## Exportar Usuarios a Excel

```
GET /api/v1/users/export
Authorization: Bearer {token}
```

## Autores

### Listar Autores

```
GET /api/v1/authors
Authorization: Bearer {token}
```

### Crear Autor (Opción 1: Con user\_id existente)

```
POST /api/v1/authors
Authorization: Bearer {token}
Content-Type: application/json

{
  "user_id": 2,
  "biography": "Reconocido autor de novelas históricas"
}
```

### Crear Autor (Opción 2: Creando usuario nuevo)

```
POST /api/v1/authors
Authorization: Bearer {token}
Content-Type: application/json

{
  "user": {
    "name": "Gabriel García Márquez",
    "email": "gabo@example.com",
    "password": "password123",
    "birth_date": "1927-03-06"
  },
  "biography": "Premio Nobel de Literatura 1982"
}
```

## Ver Autor

```
GET /api/v1/authors/{id}
Authorization: Bearer {token}
```

## Actualizar Autor

```
PUT /api/v1/authors/{id}
Authorization: Bearer {token}
Content-Type: application/json

{
  "biography": "Biografía actualizada del autor"
}
```

## Eliminar Autor

```
DELETE /api/v1/authors/{id}
Authorization: Bearer {token}
```

## Exportar Autores a Excel

```
GET /api/v1/authors/export
Authorization: Bearer {token}
```

---

## Libros

### Listar Libros

```
GET /api/v1/books
Authorization: Bearer {token}
```

### Crear Libro

```
POST /api/v1/books
Authorization: Bearer {token}
```

```
Content-Type: application/json

{
  "title": "Cien Años de Soledad",
  "description": "Obra maestra del realismo mágico",
  "published_date": "1967-05-30",
  "isbn": "978-0-06-088328-7",
  "author_id": 1
}
```

**Nota:** Al crear un libro, automáticamente se dispara un Event que ejecuta un Job para actualizar el campo `books_count` del autor.

## Ver Libro

```
GET /api/v1/books/{id}
Authorization: Bearer {token}
```

## Actualizar Libro

```
PUT /api/v1/books/{id}
Authorization: Bearer {token}
Content-Type: application/json

{
  "title": "Cien Años de Soledad - Edición Especial",
  "description": "Descripción actualizada"
}
```

## Eliminar Libro

```
DELETE /api/v1/books/{id}
Authorization: Bearer {token}
```

## Exportar Libros a Excel

```
GET /api/v1/books/export
Authorization: Bearer {token}
```

---

# Ξ Arquitectura

## Estructura del Proyecto

```
library-management-api/
├── app/
│   ├── Events/
│   │   └── BookCreated.php          # Evento al crear libro
│   ├── Http/
│   │   ├── Controllers/
│   │   │   └── Api/
│   │   │       ├── BaseController.php
│   │   │       ├── AuthController.php
│   │   │       ├── UserController.php
│   │   │       ├── AuthorController.php
│   │   │       └── BookController.php
│   │   ├── Requests/                # Form Requests
│   │   └── Middleware/
│   ├── Jobs/
│   │   └── UpdateAuthorBooksCount.php # Job para actualizar contador
│   ├── Listeners/
│   │   └── UpdateAuthorBooksCountListener.php
│   ├── Services/
│   │   ├── BaseService.php
│   │   ├── AuthService.php
│   │   ├── UserService.php
│   │   ├── AuthorService.php
│   │   └── BookService.php
│   ├── Author.php                  # Modelo Author
│   ├── Book.php                   # Modelo Book
│   └── User.php                   # Modelo User
├── database/
│   └── migrations/
└── docker/
    └── nginx/
        └── default.conf
├── Dockerfile
└── docker-compose.yml
└── README.md
```

## Patrones y Principios

- **Service Layer:** Lógica de negocio separada de controladores
- **Repository Pattern (implícito):** A través de Eloquent ORM
- **Single Responsibility:** Cada clase tiene una responsabilidad clara
- **DRY (Don't Repeat Yourself):** BaseService y BaseController reutilizables
- **Dependency Injection:** Servicios injectados en controladores
- **Event-Driven Architecture:** Events y Jobs para acciones asíncronas

## Base de Datos

### Relaciones

```
users (1) —— (1) authors (1) —— (*) books
```

- Un **Usuario** puede ser un **Autor**
- Un **Autor** pertenece a un **Usuario**
- Un **Autor** tiene muchos **Libros**
- Un **Libro** pertenece a un **Autor**

## Auditoría

Todas las tablas incluyen:

- `created_at, updated_at` (timestamps)
- `deleted_at` (soft delete)
- `created_by, updated_by, deleted_by` (auditoría de usuario)

## ✍ Testing

(Opcional - Si decides agregar tests básicos)

```
docker-compose exec app php artisan test
```

## ⚡ Troubleshooting

### Error de permisos

```
docker-compose exec app chmod -R 775 storage/bootstrap/cache  
docker-compose exec app chown -R www-data:www-data storage/bootstrap/cache
```

### Limpiar caché

```
docker-compose exec app php artisan cache:clear  
docker-compose exec app php artisan config:clear  
docker-compose exec app php artisan view:clear
```

### Recrear contenedores

```
docker-compose down  
docker-compose up -d --build
```

### Ver logs

```
docker-compose logs -f app
docker-compose logs -f nginx
docker-compose logs -f db
```

## Notas de Desarrollo

- El proyecto usa Laravel 5.1 por requerimiento de la prueba técnica
- JWT configurado con tymon/jwt-auth 0.5 (compatible con Laravel 5.1)
- PostgreSQL en puerto externo 5433 (puerto interno 5432)
- Los exports de Excel se descargan directamente al hacer GET

## Autor

### Cristobal Canto

- GitHub: [@tu-usuario](#)
- Email: cantoniorodriguez2307@gmail.com

---

Desarrollado con ❤ usando Laravel, Docker y PostgreSQL BY CR