**📋 Documentación de API - Sistema RentaTec ITSC**

**## 🏗️ Arquitectura del Sistema**

**### Estructura General**

El sistema RentaTec está construido con **\*\*Laravel 12\*\*** y utiliza una arquitectura híbrida que combina:

- **\*\*Aplicación principal monolítica\*\*** (Laravel)

- **\*\*Microservicios especializados\*\*** (Inventario, Préstamos, Usuarios)

- **\*\*Base de datos MySQL\*\***

- **\*\*Autenticación con Laravel Sanctum\*\***

- **\*\*Sistema de notificaciones por email\*\***

**### Componentes Principales**

```

RentaTec/

├── app/                          # Aplicación principal Laravel

│   ├── Http/Controllers/         # Controladores principales

│   ├── Models/                   # Modelos Eloquent

│   ├── Services/                 # Servicios del sistema

│   └── Notifications/            # Sistema de notificaciones

├── microservicio\_inventario/     # Microservicio de inventario

├── microservicio\_prestamos/      # Microservicio de préstamos

├── microservicio\_usuarios/       # Microservicio de usuarios

├── database/                     # Migraciones y seeders

└── docker/                       # Configuración Docker

```

**## 🗄️ Base de Datos**

**### Configuración**

```env

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=rentatec

DB\_USERNAME=root

DB\_PASSWORD=1234

```

**### Tablas Principales**

**#### 👥 Tabla `users`**

```sql

- id (bigint, PK)

- name (varchar)

- email (varchar, unique)

- password (varchar, hashed)

- role (enum: 'alumno', 'docente', 'personal', 'admin')

- matricula (varchar)

- codigo\_credencial (varchar)

- oportunidades\_restantes (int, default: 3)

- bloqueado (boolean, default: false)

- fecha\_desbloqueo (datetime, nullable)

- retrasos\_totales (int, default: 0)

- created\_at, updated\_at

```

**#### 📦 Tabla `inventarios`**

```sql

- id (bigint, PK)

- nombre\_equipo (varchar)

- numero\_serie (varchar, unique)

- disponible (boolean, default: true)

- estado (enum: 'nuevo', 'funcionando', 'no\_funciona')

- estado\_prestamo (enum: 'activo', 'mantenimiento', 'guardado', 'perdido')

- categoria (varchar)

- marca (varchar, nullable)

- modelo (varchar, nullable)

- descripcion (text, nullable)

- motivo\_no\_prestamo (text, nullable)

- razon\_no\_activo (text, nullable)

- detalle\_mantenimiento (text, nullable)

- cable\_corriente (boolean, default: false)

- hdmi (boolean, default: false)

- extension (boolean, default: false)

- ultimo\_uso (datetime, nullable) # Para enfriamiento de proyectores

- created\_at, updated\_at

```

**#### 📋 Tabla `prestamos`**

```sql

- id (bigint, PK)

- user\_id (bigint, FK -> users.id)

- inventario\_id (bigint, FK -> inventarios.id)

- fecha\_entrega (date)

- fecha\_devolucion (date)

- hora\_inicio (time)

- hora\_fin (time)

- carrera (varchar, nullable)

- grado (varchar, nullable)

- grupo (varchar, nullable)

- aula (varchar, nullable)

- materia (varchar, nullable)

- nombre\_maestro (varchar, nullable)

- estado (enum: 'pendiente', 'aprobado', 'rechazado', 'devuelto', 'no\_entregado', 'solicitud\_devolucion')

- entregado\_como (varchar, nullable)

- es\_evento (boolean, default: false)

- whatsapp (varchar)

- created\_at, updated\_at

```

**## 🔗 APIs y Endpoints**

**### 🔐 Autenticación**

**#### Login**

```http

POST /api/login

Content-Type: application/json

{

    "name": "usuario",

    "password": "contraseña"

}

Response:

{

    "user": {...},

    "token": "sanctum\_token"

}

```

**### 👥 API de Usuarios**

**#### Obtener todos los usuarios (Autenticado)**

```http

GET /api/users

Authorization: Bearer {token}

```

**#### Obtener usuario por credencial**

```http

GET /api/users/{codigo\_credencial}

Authorization: Bearer {token}

```

**#### Crear usuario**

```http

POST /api/users

Authorization: Bearer {token}

Content-Type: application/json

{

    "name": "Nombre Usuario",

    "email": "usuario@itsc.edu.mx",

    "password": "contraseña",

    "role": "alumno|docente|personal|admin",

    "matricula": "20240001",

    "codigo\_credencial": "CRED001"

}

```

**### 📦 API de Inventario**

**#### Obtener inventario completo (Autenticado)**

```http

GET /api/inventarios

Authorization: Bearer {token}

```

**#### Obtener producto específico**

```http

GET /api/inventarios/{id}

Authorization: Bearer {token}

```

**#### Crear producto**

```http

POST /api/inventarios

Authorization: Bearer {token}

Content-Type: application/json

{

    "nombre\_equipo": "Proyector Epson",

    "numero\_serie": "EP001",

    "estado": "funcionando",

    "estado\_prestamo": "activo",

    "categoria": "proyector",

    "marca": "Epson",

    "modelo": "PowerLite",

    "cable\_corriente": true,

    "hdmi": true,

    "extension": false

}

```

**#### Actualizar producto**

```http

PUT /api/inventarios/{id}

Authorization: Bearer {token}

Content-Type: application/json

{

    "disponible": false,

    "estado\_prestamo": "mantenimiento",

    "detalle\_mantenimiento": "Limpieza de filtros"

}

```

**### 📋 API de Préstamos**

**#### Obtener todos los préstamos (Autenticado)**

```http

GET /api/prestamos

Authorization: Bearer {token}

```

**#### Obtener préstamos por usuario**

```http

GET /api/prestamos/user/{user\_id}

Authorization: Bearer {token}

```

**#### Obtener préstamos pendientes**

```http

GET /api/prestamos/pending

Authorization: Bearer {token}

```

**#### Crear solicitud de préstamo**

```http

POST /api/prestamos

Authorization: Bearer {token}

Content-Type: application/json

{

    "inventario\_id": 1,

    "fecha\_entrega": "2024-01-15",

    "fecha\_devolucion": "2024-01-16",

    "hora\_inicio": "08:00",

    "hora\_fin": "12:00",

    "carrera": "Ingeniería en Sistemas",

    "grado": "5",

    "grupo": "A",

    "aula": "Lab 1",

    "materia": "Programación Web",

    "nombre\_maestro": "Prof. García",

    "whatsapp": "+529331234567",

    "es\_evento": false

}

```

**#### Aprobar préstamo (Solo Admin)**

```http

POST /api/prestamos/{id}/approve

Authorization: Bearer {admin\_token}

```

**#### Rechazar préstamo (Solo Admin)**

```http

POST /api/prestamos/{id}/reject

Authorization: Bearer {admin\_token}

```

**## 🐳 Microservicios**

**### Microservicio de Inventario**

**\*\*Puerto:\*\*** 8001

**\*\*Endpoints:\*\***

```http

GET /api/inventario/disponibles          # Productos disponibles

GET /api/inventario/search?q=proyector   # Búsqueda de productos

POST /api/inventario                     # Crear producto

PUT /api/inventario/{id}                 # Actualizar producto

PATCH /api/inventario/{id}/toggle-disponible  # Cambiar disponibilidad

```

**### Microservicio de Préstamos**

**\*\*Puerto:\*\*** 8002

**\*\*Endpoints:\*\***

```http

GET /api/prestamos/pending               # Préstamos pendientes

GET /api/prestamos/retrasados           # Préstamos retrasados

GET /api/prestamos/user/{userId}        # Préstamos por usuario

POST /api/prestamos/{id}/aprobar        # Aprobar préstamo

POST /api/prestamos/{id}/devolver       # Marcar como devuelto

```

**### Microservicio de Usuarios**

**\*\*Puerto:\*\*** 8003

**\*\*Endpoints:\*\***

```http

GET /api/usuarios/search?q=nombre       # Buscar usuarios

GET /api/usuarios/bloqueados           # Usuarios bloqueados

GET /api/usuarios/{id}/sanciones       # Historial de sanciones

POST /api/usuarios/{id}/desbloquear    # Desbloquear usuario

```

**## 🔧 Configuración de Docker**

**### Docker Compose Principal**

```yaml

version: '3.8'

services:

  app:

    build: .

    container\_name: rentatec\_app

    ports:

      - "8000:8000"

    environment:

      - DB\_HOST=db

      - DB\_DATABASE=rentatec

    depends\_on:

      - db

  db:

    image: mysql:8.0

    container\_name: rentatec\_db

    environment:

      MYSQL\_DATABASE: rentatec

      MYSQL\_USER: rentatec\_user

      MYSQL\_PASSWORD: rentatec\_password

    ports:

      - "3306:3306"

  nginx:

    image: nginx:alpine

    container\_name: rentatec\_nginx

    ports:

      - "80:80"

```

**### Comandos Docker**

```bash

# Levantar todos los servicios

docker-compose up -d

# Levantar solo microservicios

docker-compose -f docker-compose.microservicios.yml up -d

# Ver logs

docker-compose logs -f app

# Ejecutar migraciones

docker-compose exec app php artisan migrate

# Acceder al contenedor

docker-compose exec app bash

```

**## 🔍 Comandos de Consulta Útiles**

**### 1. Verificar si un usuario ya hizo un préstamo**

**#### Via API**

```http

GET /api/prestamos/user/{user\_id}

Authorization: Bearer {token}

```

**#### Via Artisan Command**

```bash

php artisan user:borrowed-count {user\_id}

```

**#### Via Base de Datos**

```sql

-- Verificar préstamos activos de un usuario

SELECT p.\*, u.name, i.nombre\_equipo

FROM prestamos p

JOIN users u ON p.user\_id = u.id

JOIN inventarios i ON p.inventario\_id = i.id

WHERE u.id = 1 AND p.estado IN ('pendiente', 'aprobado');

-- Contar préstamos totales por usuario

SELECT u.name, COUNT(p.id) as total\_prestamos

FROM users u

LEFT JOIN prestamos p ON u.id = p.user\_id

WHERE u.id = 1

GROUP BY u.id;

```

**### 2. Consultar equipos disponibles**

**#### Via API**

```http

GET /api/inventarios

Authorization: Bearer {token}

```

**#### Via Base de Datos**

```sql

-- Equipos disponibles para préstamo

SELECT \* FROM inventarios

WHERE disponible = 1

AND estado\_prestamo = 'activo'

AND estado = 'funcionando';

-- Equipos con enfriamiento (proyectores)

SELECT \*,

    CASE

        WHEN categoria = 'proyector' AND ultimo\_uso IS NOT NULL

        THEN TIMESTAMPDIFF(MINUTE, ultimo\_uso, NOW())

        ELSE NULL

    END as minutos\_desde\_uso

FROM inventarios

WHERE categoria = 'proyector';

```

**### 3. Consultar préstamos pendientes**

**#### Via API**

```http

GET /api/prestamos/pending

Authorization: Bearer {token}

```

**#### Via Base de Datos**

```sql

-- Préstamos pendientes de aprobación

SELECT p.\*, u.name as usuario, i.nombre\_equipo

FROM prestamos p

JOIN users u ON p.user\_id = u.id

JOIN inventarios i ON p.inventario\_id = i.id

WHERE p.estado = 'pendiente'

ORDER BY p.created\_at DESC;

```

**### 4. Consultar préstamos retrasados**

**#### Via Base de Datos**

```sql

-- Préstamos retrasados (más de 15 minutos)

SELECT p.\*, u.name as usuario, i.nombre\_equipo,

    TIMESTAMPDIFF(MINUTE,

        CONCAT(p.fecha\_devolucion, ' ', p.hora\_fin),

        NOW()

    ) as minutos\_retraso

FROM prestamos p

JOIN users u ON p.user\_id = u.id

JOIN inventarios i ON p.inventario\_id = i.id

WHERE p.estado = 'aprobado'

AND CONCAT(p.fecha\_devolucion, ' ', p.hora\_fin) < DATE\_SUB(NOW(), INTERVAL 15 MINUTE);

```

**### 5. Consultar usuarios bloqueados**

**#### Via API**

```http

GET /api/usuarios/bloqueados

```

**#### Via Base de Datos**

```sql

-- Usuarios bloqueados o con pocas oportunidades

SELECT name, matricula, oportunidades\_restantes, retrasos\_totales,

    bloqueado, fecha\_desbloqueo

FROM users

WHERE bloqueado = 1 OR oportunidades\_restantes < 3

ORDER BY fecha\_desbloqueo DESC;

```

**### 6. Estadísticas del sistema**

```sql

-- Resumen general del sistema

SELECT

    (SELECT COUNT(\*) FROM users WHERE role != 'admin') as total\_usuarios,

    (SELECT COUNT(\*) FROM inventarios WHERE estado\_prestamo = 'activo') as equipos\_activos,

    (SELECT COUNT(\*) FROM prestamos WHERE estado = 'pendiente') as prestamos\_pendientes,

    (SELECT COUNT(\*) FROM prestamos WHERE estado = 'aprobado') as prestamos\_activos,

    (SELECT COUNT(\*) FROM users WHERE bloqueado = 1) as usuarios\_bloqueados;

-- Equipos más solicitados

SELECT i.nombre\_equipo, i.categoria, COUNT(p.id) as veces\_prestado

FROM inventarios i

LEFT JOIN prestamos p ON i.id = p.inventario\_id

GROUP BY i.id

ORDER BY veces\_prestado DESC

LIMIT 10;

-- Usuarios más activos

SELECT u.name, u.role, COUNT(p.id) as total\_prestamos

FROM users u

LEFT JOIN prestamos p ON u.id = p.user\_id

WHERE u.role != 'admin'

GROUP BY u.id

ORDER BY total\_prestamos DESC

LIMIT 10;

```

**## 🚀 Comandos de Desarrollo**

**### Instalación y Configuración**

```bash

# Clonar repositorio

git clone [repository-url]

cd RentaTec

# Instalar dependencias

composer install

npm install

# Configurar entorno

cp .env.example .env

php artisan key:generate

# Ejecutar migraciones

php artisan migrate

# Sembrar datos de prueba

php artisan db:seed

# Compilar assets

npm run dev

# Iniciar servidor

php artisan serve

```

**### Comandos Artisan Personalizados**

```bash

# Contar préstamos de usuario

php artisan user:borrowed-count {user\_id}

# Verificar retrasos automáticamente

php artisan prestamos:verificar-retrasos

# Limpiar notificaciones antiguas

php artisan notifications:cleanup

# Generar reporte mensual

php artisan reportes:mensual

```

**### Testing**

```bash

# Ejecutar tests

php artisan test

# Tests específicos

php artisan test --filter PrestamoTest

# Coverage

php artisan test --coverage

```

**## 📊 Monitoreo y Logs**

**### Logs del Sistema**

```bash

# Ver logs en tiempo real

tail -f storage/logs/laravel.log

# Logs específicos de préstamos

grep "prestamo" storage/logs/laravel.log

# Limpiar logs

php artisan log:clear

```

**### Métricas Importantes**

- Tiempo de respuesta de APIs

- Número de préstamos por día/semana/mes

- Equipos más utilizados

- Usuarios con más retrasos

- Disponibilidad del sistema

**## 🔒 Seguridad**

**### Autenticación**

- Laravel Sanctum para API tokens

- Middleware de autenticación en todas las rutas protegidas

- Roles y permisos por tipo de usuario

**### Validaciones**

- Validación de datos en todos los endpoints

- Sanitización de inputs

- Verificación de permisos por rol

**### Rate Limiting**

```php

// En RouteServiceProvider

RateLimiter::for('api', function (Request $request) {

    return Limit::perMinute(60)->by($request->user()?->id ?: $request->ip());

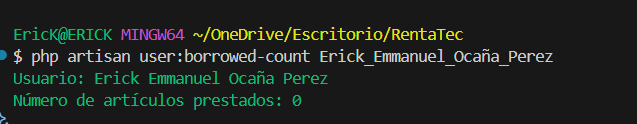
});

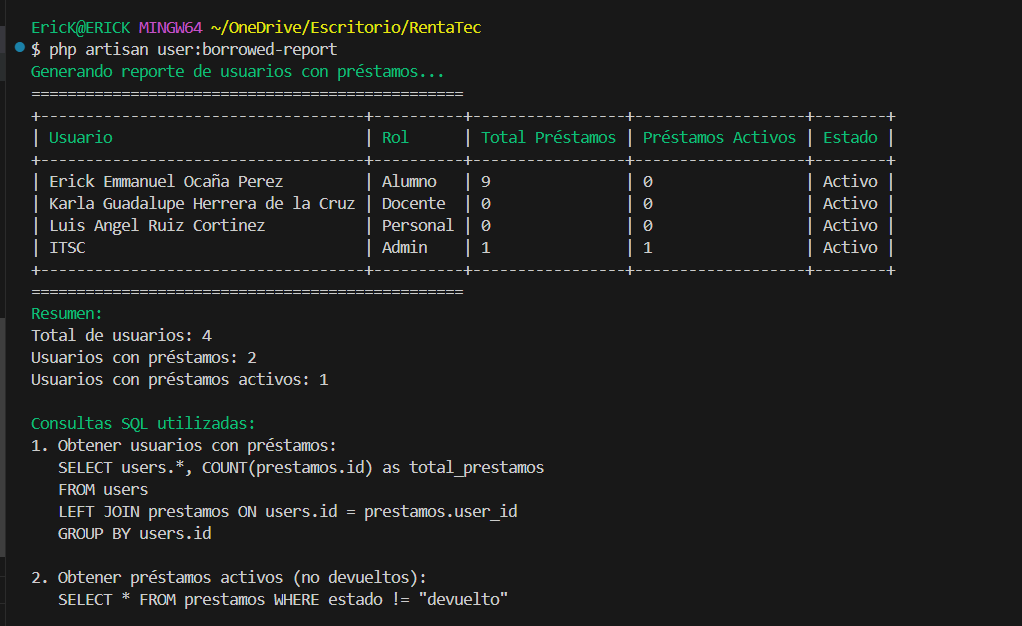
*Esta documentación fue generada automáticamente basada en el análisis del código fuente del sistema RentaTec.*

Ejemplos de consultas via API:

1. Obtener préstamos de usuario:

   GET /api/prestamos/user/{user\_id}

   Authorization: Bearer {token}  


2. Verificar equipos disponibles:  


php artisan user:borrowed-report

**# Microservicios en RentaTec**

Los microservicios identificados en el proyecto RentaTec son los siguientes:

1. **\*\*Microservicio de Usuarios\*\*** (`microservicio\_usuarios`)

   - Puerto: 8001

   - Responsabilidades: Gestión de usuarios (CRUD), autenticación, autorización, sanciones y bloqueos.

2. **\*\*Microservicio de Inventario\*\*** (`microservicio\_inventario`)

   - Puerto: 8002

   - Responsabilidades: Gestión de equipos/inventario (CRUD), control de disponibilidad, búsqueda y filtrado.

3. **\*\*Microservicio de Préstamos\*\*** (`microservicio\_prestamos`)

   - Puerto: 8003

   - Responsabilidades: Gestión de solicitudes de préstamo, aprobación/rechazo, devoluciones, sanciones y extensiones de fechas.

Estos microservicios se comunican entre sí mediante HTTP y están diseñados para funcionar de manera independiente, facilitando la escalabilidad y el mantenimiento del sistema.