# Configuración de API de Microservicios

## Descripción General

Este documento describe la configuración de la API para los microservicios del proyecto de inventario ITSC. El proyecto utiliza una arquitectura de microservicios basada en Laravel, donde cada servicio maneja una funcionalidad específica.

Los microservicios están ubicados en el directorio `inventario-itsc-microservices/`.

### Arquitectura del Sistema

El proyecto sigue una arquitectura de microservicios con los siguientes componentes:

- \*\*Aplicación Principal (Main App)\*\*: Aplicación Laravel principal ubicada en la raíz del proyecto, que actúa como frontend y orquestador de los microservicios.

- \*\*Auth Service\*\*: Microservicio dedicado a la autenticación y gestión de usuarios.

- \*\*Inventory Service\*\*: Microservicio para la gestión del inventario de bienes.

- \*\*Loan Service\*\*: Microservicio para gestión de préstamos (en desarrollo).

La aplicación principal consume los microservicios a través de llamadas HTTP, utilizando servicios como `InventoryService` para interactuar con el Inventory Service.

### Servicios Disponibles

- \*\*Auth Service\*\*: Maneja la autenticación de usuarios, gestión de usuarios y roles.

- \*\*Inventory Service\*\*: Gestiona el inventario de bienes, categorías, edificios y estados.

- \*\*Loan Service\*\*: Servicio para gestión de préstamos (actualmente en desarrollo).

- \*\*Main App\*\*: Aplicación principal que integra todos los servicios y proporciona la interfaz de usuario.

## Comandos para Iniciar Servicios

Para ejecutar los microservicios en los puertos especificados, utiliza los siguientes comandos en terminales separadas:

### Auth Service (Puerto 8001)

```bash

cd inventario-itsc-microservices/auth-service

php artisan serve --host=0.0.0.0 --port=8001

```

### Inventory Service (Puerto 8002)

```bash

cd inventario-itsc-microservices/inventory-service

php artisan serve --host=0.0.0.0 --port=8002

```

### Loan Service (Puerto no configurado)

El servicio de préstamos no tiene un puerto asignado actualmente, ya que está en desarrollo.

### Aplicación Principal (Main App)

```bash

php artisan serve --host=0.0.0.0 --port=8000

```

Nota: Asegúrate de tener las dependencias instaladas (`composer install`) y la base de datos configurada antes de iniciar los servicios.

### Inicio Conjunto de Servicios

Para iniciar todos los servicios activos simultáneamente, puedes usar múltiples terminales o crear un script batch/shell. Actualmente, los servicios auth e inventory están configurados para ejecutarse en terminales separadas.

Ejemplo de comandos para iniciar todos:

- Terminal 1: Auth Service (puerto 8001)

- Terminal 2: Inventory Service (puerto 8002)

- Terminal 3: Main App (puerto 8000)

## APIs Utilizadas y Flujo de Comunicación

### APIs de Microservicios Utilizadas

El proyecto utiliza dos APIs principales de microservicios que son consumidas desde la aplicación principal y comandos de consola:

#### 1. \*\*Auth Service API\*\* (Puerto 8001)

\*\*Ubicación del código fuente:\*\* `inventario-itsc-microservices/auth-service/`

\*\*Endpoints utilizados en el proyecto:\*\*

- `POST /api/login` - Endpoint para autenticación de usuarios (login)

- `GET /api/user` - Obtener información del usuario actualmente autenticado

- `GET /api/verify` - Verificar la validez de un token de autenticación

\*\*Desde dónde se consumen estos endpoints:\*\*

- `app/Console/Commands/ConsultaInventario.php` - Comando de consola que utiliza este endpoint para autenticación automática antes de consultar el inventario

\*\*Flujo de comunicación:\*\*

- El comando `ConsultaInventario` envía credenciales (username/password) al Auth Service

- Recibe un token JWT de Sanctum

- Utiliza este token para autenticar las subsiguientes llamadas al Inventory Service

#### 2. \*\*Inventory Service API\*\* (Puerto 8002)

\*\*Ubicación del código fuente:\*\* `inventario-itsc-microservices/inventory-service/`

\*\*Endpoints utilizados en el proyecto:\*\*

- `GET /api/bienes` - Obtener lista completa de bienes del inventario

- `GET /api/bienes/{id}` - Obtener detalles específicos de un bien por ID

- `GET /api/bienes-export` - Exportar datos de bienes (funcionalidad de exportación)

- `GET /api/edificios` - Obtener lista de edificios disponibles

- `GET /api/categorias` - Obtener lista de categorías de bienes

- `GET /api/estados` - Obtener lista de estados posibles de los bienes

\*\*Desde dónde se consumen estos endpoints:\*\*

- `app/Services/InventoryService.php` - Servicio principal de Laravel que encapsula todas las llamadas HTTP al Inventory Service

- `app/Http/Controllers/DashboardController.php` - Controlador que utiliza `InventoryService` para servir datos a las vistas del dashboard

- `app/Console/Commands/ConsultaInventario.php` - Comando de consola que consulta el inventario después de autenticarse

\*\*Flujo de comunicación:\*\*

- La aplicación principal (Main App) recibe peticiones en sus rutas API (`routes/api.php`)

- Estas peticiones son manejadas por controladores como `DashboardController`

- Los controladores delegan las operaciones al servicio `InventoryService`

- `InventoryService` realiza llamadas HTTP GET/POST al Inventory Service usando tokens de autenticación

- Los datos se devuelven a través de la cadena: Inventory Service → InventoryService → DashboardController → rutas API → cliente

### Configuración de URLs de Microservicios

Las URLs de los microservicios están centralizadas en el archivo de configuración `config/services.php` de la aplicación principal:

```php

'services' => [

    'inventory\_service' => [

        'url' => env('INVENTORY\_SERVICE\_URL', 'http://localhost:8002/api'),

    ],

],

```

Esta configuración permite:

- Cambiar fácilmente las URLs en diferentes entornos (desarrollo, producción)

- Centralizar la configuración de endpoints

- Facilitar el mantenimiento y escalabilidad

### Rutas API Expuestas por la Aplicación Principal

La aplicación principal expone las siguientes rutas API que internamente consumen los microservicios:

- `GET /api/bienes/{edificioId}` - Bienes filtrados por edificio (DashboardController::getBienesByEdificio)

- `GET /api/bienes/detalle/{id}` - Detalles completos de un bien (DashboardController::getBien)

- `GET /api/bienes-export` - Exportación de bienes (DashboardController::exportBienes)

- `GET /api/estadisticas` - Estadísticas del dashboard (ConsultaController::dashboardStats)

- `GET /api/usuarios-activos` - Usuarios activos (ConsultaController::usuariosActivos)

Todas estas rutas están protegidas por el middleware `auth.custom` que requiere autenticación previa.

### Comando de Consola para Consultas

Además de las llamadas HTTP desde la aplicación web, el sistema incluye un comando Artisan:

```bash

php artisan consulta:inventario {usuario} {password}

```

Este comando:

1. \*\*Autenticación\*\*: Llama al Auth Service API para obtener un token

2. \*\*Consulta\*\*: Utiliza el token para llamar al Inventory Service API

3. \*\*Salida\*\*: Muestra los resultados en formato JSON en la consola

\*\*Ubicación:\*\* `app/Console/Commands/ConsultaInventario.php`

## Configuración de Servicios

### Main App (Aplicación Principal)

- \*\*Ubicación del código\*\*: Raíz del proyecto (`/`), archivos principales en `app/`, `routes/`, `resources/`

- \*\*Puerto de ejecución\*\*: 8000 (por defecto)

- \*\*Host\*\*: 0.0.0.0 (accesible desde cualquier interfaz)

- \*\*Base de datos\*\*: SQLite (`database/database.sqlite`)

- \*\*Configuración principal\*\*:

  - APP\_ENV=local

  - APP\_DEBUG=true

  - DB\_CONNECTION=sqlite

  - Servicios externos configurados en `config/services.php`

- \*\*Archivos de configuración importantes\*\*:

  - `config/services.php`: Configuración de URLs de microservicios

    ```php

    'inventory\_service' => [

        'url' => env('INVENTORY\_SERVICE\_URL', 'http://localhost:8002/api'),

    ],

    ```

  - `routes/api.php`: Rutas de la API principal

  - `routes/web.php`: Rutas web para vistas

- \*\*Servicios generados\*\*:

  - `app/Services/InventoryService.php`: Servicio para consumir el Inventory Service vía HTTP

- \*\*Controladores\*\*:

  - `app/Http/Controllers/DashboardController.php`: Controlador para el dashboard principal

- \*\*Comandos de consola\*\*:

  - `app/Console/Commands/ConsultaInventario.php`: Comando para consultar inventario desde consola

    - Uso: `php artisan consulta:inventario {usuario} {password}`

- \*\*Vistas\*\*:

  - `resources/views/layouts/admin.blade.php`: Layout principal para vistas admin

### Auth Service

- \*\*Ubicación del código\*\*: `inventario-itsc-microservices/auth-service/`

- \*\*Puerto de ejecución\*\*: 8001

- \*\*Host\*\*: 0.0.0.0 (accesible desde cualquier interfaz)

- \*\*Base de datos\*\*: SQLite (`database/database.sqlite`)

- \*\*Configuración principal\*\*:

  - APP\_ENV=local

  - APP\_DEBUG=true

  - DB\_CONNECTION=sqlite

  - SESSION\_DRIVER=database

  - CACHE\_STORE=database

- \*\*Modelos\*\*:

  - `app/Models/User.php`: Modelo de Usuario

  - Roles gestionados a través de seeders

- \*\*Controladores\*\*:

  - `app/Http/Controllers/AuthController.php`: Controlador de autenticación

  - `app/Http/Controllers/UserController.php`: Controlador de gestión de usuarios

- \*\*Seeders\*\*:

  - `database/seeders/UsersSeeder.php`: Crea usuarios de prueba con roles

  - `database/seeders/RolesSeeder.php`: Crea roles del sistema

  - `database/seeders/DatabaseSeeder.php`: Ejecuta todos los seeders

- \*\*Endpoints de API\*\*:

  - `POST /api/login` - Inicio de sesión

  - `POST /api/logout` - Cierre de sesión (requiere autenticación)

  - `GET /api/user` - Información del usuario actual (requiere autenticación)

  - `GET /api/verify` - Verificación de token (requiere autenticación)

  - `GET /api/users` - Lista de usuarios (requiere autenticación, admin)

  - `POST /api/users` - Crear usuario (requiere autenticación, admin)

  - `GET /api/users/{id}` - Detalles de usuario (requiere autenticación, admin)

  - `PUT /api/users/{id}` - Actualizar usuario (requiere autenticación, admin)

  - `DELETE /api/users/{id}` - Eliminar usuario (requiere autenticación, admin)

  - `PUT /api/users/{id}/change-password` - Cambiar contraseña (requiere autenticación, admin)

  - `GET /api/roles` - Lista de roles (requiere autenticación, admin)

  - `POST /api/roles` - Crear rol (requiere autenticación, admin)

### Inventory Service

- \*\*Ubicación del código\*\*: `inventario-itsc-microservices/inventory-service/`

- \*\*Puerto de ejecución\*\*: 8002

- \*\*Host\*\*: 0.0.0.0 (accesible desde cualquier interfaz)

- \*\*Base de datos\*\*: SQLite compartido (`../auth-service/database/database.sqlite`)

- \*\*Configuración principal\*\*:

  - APP\_ENV=local

  - APP\_DEBUG=true

  - DB\_CONNECTION=sqlite

  - DB\_DATABASE=../auth-service/database/database.sqlite

  - SESSION\_DRIVER=database

  - CACHE\_STORE=database

- \*\*Modelos\*\*:

  - `app/Models/Bien.php`: Modelo de Bien (ítem del inventario)

  - `app/Models/Categoria.php`: Modelo de Categoría

  - `app/Models/Edificio.php`: Modelo de Edificio

  - `app/Models/Estado.php`: Modelo de Estado

  - `app/Models/User.php`: Modelo de Usuario (referencia al auth service)

- \*\*Controladores\*\*:

  - `app/Http/Controllers/BienController.php`: Controlador para gestión de bienes

  - `app/Http/Controllers/ReferenceDataController.php`: Controlador para datos de referencia (categorías, edificios, estados)

  - `app/Http/Controllers/Controller.php`: Controlador base

- \*\*Seeders\*\*:

  - `database/seeders/CategoriasSeeder.php`: Crea categorías de bienes

  - `database/seeders/EdificiosSeeder.php`: Crea edificios

  - `database/seeders/EstadosSeeder.php`: Crea estados de bienes

  - `database/seeders/DatabaseSeeder.php`: Ejecuta todos los seeders

- \*\*Endpoints de API\*\*:

  - `GET /api/bienes` - Lista de bienes

  - `POST /api/bienes` - Crear bien

  - `GET /api/bienes/{id}` - Detalles de bien

  - `PUT /api/bienes/{id}` - Actualizar bien

  - `DELETE /api/bienes/{id}` - Eliminar bien

  - `GET /api/bienes/detalle/{id}` - Detalles completos de bien

  - `GET /api/bienes-export` - Exportar bienes

  - `GET /api/edificios` - Lista de edificios

  - `GET /api/categorias` - Lista de categorías

  - `GET /api/estados` - Lista de estados

### Loan Service

- \*\*Ubicación del código\*\*: `inventario-itsc-microservices/loan-service/`

- \*\*Puerto de ejecución\*\*: No configurado (servicio no activo)

- \*\*Base de datos\*\*: SQLite (`database/database.sqlite`)

- \*\*Configuración principal\*\*:

  - APP\_ENV=local

  - APP\_DEBUG=true

  - DB\_CONNECTION=sqlite

  - APP\_KEY= (no generado)

- \*\*Modelos\*\*:

  - `app/Models/User.php`: Modelo básico de Usuario

- \*\*Endpoints de API\*\* (placeholder):

  - `GET /api/loans` - Lista de préstamos (requiere autenticación, actualmente devuelve mensaje placeholder)

## Autenticación

La mayoría de los endpoints requieren autenticación mediante tokens de Sanctum. Para acceder a endpoints protegidos:

1. Realizar login en el Auth Service para obtener un token. El login requiere `username` y `password` (no email).

2. Incluir el token en el header `Authorization: Bearer {token}` en las solicitudes subsiguientes.

## Datos Sembrados

### Usuarios (Auth Service)

Los siguientes usuarios están disponibles en la base de datos sembrada (ver `UsersSeeder.php`):

- \*\*Admin\*\*: username: `admin`, password: `admin123`

- \*\*Auditor\*\*: username: `auditor1`, password: `auditor123`

- \*\*Personal de Almacén\*\*: username: `almacen1`, password: `almacen123`

- \*\*Jefes de Edificio\*\*: username: `jefe\_edificio\_a`, password: `jefea123` (y similares para otros edificios)

### Roles (Auth Service)

Roles definidos en `RolesSeeder.php`:

- Administrador

- Auditor

- Personal de Almacén

- Jefe de Edificio

### Datos de Referencia (Inventory Service)

- \*\*Categorías\*\* (`CategoriasSeeder.php`): Electrónicos, Muebles, Herramientas, etc.

- \*\*Edificios\*\* (`EdificiosSeeder.php`): Edificio A, Edificio B, etc.

- \*\*Estados\*\* (`EstadosSeeder.php`): Nuevo, Usado, Dañado, etc.

## Comandos para Consultas

A continuación, ejemplos de comandos curl para realizar consultas a la API. Asegúrate de que los servicios estén ejecutándose en los puertos indicados.

### 1. Inicio de Sesión (Auth Service)

```bash

curl -X POST http://localhost:8001/api/login \

  -H "Content-Type: application/json" \

  -d '{"username":"admin","password":"admin123"}'

```

Respuesta esperada: JSON con token de acceso.

\*\*Nota\*\*: Este endpoint solo acepta solicitudes POST. No puede accederse directamente desde un navegador web (que envía GET por defecto). Utiliza curl, Postman, o herramientas similares para hacer la solicitud.

\*\*Solución de problemas\*\*:

- Asegúrate de incluir el header `Content-Type: application/json` y los datos JSON en el body.

- Si obtienes errores de método no permitido, ejecuta `php artisan route:clear` y `php artisan route:cache` en el directorio del servicio auth.

- Verifica que el servicio esté ejecutándose en el puerto 8001.

### 2. Verificación de Token

```bash

curl -X GET http://localhost:8001/api/verify \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

### 3. Obtener Lista de Bienes (Inventory Service)

```bash

curl -X GET http://localhost:8002/api/bienes \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

### 4. Obtener Detalles de un Bien Específico

```bash

curl -X GET http://localhost:8002/api/bienes/1 \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

### 5. Obtener Lista de Edificios

```bash

curl -X GET http://localhost:8002/api/edificios \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

### 6. Obtener Lista de Categorías

```bash

curl -X GET http://localhost:8002/api/categorias \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

### 7. Obtener Lista de Estados

```bash

curl -X GET http://localhost:8002/api/estados \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

### 8. Crear un Nuevo Bien

```bash

curl -X POST http://localhost:8002/api/bienes \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI" \

  -H "Content-Type: application/json" \

  -d '{

    "nombre": "Nuevo Bien",

    "descripcion": "Descripción del bien",

    "categoria\_id": 1,

    "edificio\_id": 1,

    "estado\_id": 1

  }'

```

### 9. Actualizar un Bien

```bash

curl -X PUT http://localhost:8002/api/bienes/1 \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI" \

  -H "Content-Type: application/json" \

  -d '{

    "nombre": "Bien Actualizado",

    "descripcion": "Descripción actualizada"

  }'

```

### 10. Eliminar un Bien

```bash

curl -X DELETE http://localhost:8002/api/bienes/1 \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

### 11. Obtener Lista de Usuarios (Auth Service, requiere admin)

```bash

curl -X GET http://localhost:8001/api/users \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

### 12. Obtener Información del Usuario Actual

```bash

curl -X GET http://localhost:8001/api/user \

  -H "Authorization: Bearer TU\_TOKEN\_AQUI"

```

Nota: Reemplaza `TU\_TOKEN\_AQUI` con el token obtenido del login. Los endpoints que requieren permisos de administrador solo funcionarán con usuarios que tengan rol de administrador.

## Comando de Consola para Consultas de Inventario

Además de las llamadas HTTP, el sistema incluye un comando de consola para consultar el inventario:

```bash

php artisan consulta:inventario {usuario} {password}

```

Este comando:

- Se autentica automáticamente con las credenciales proporcionadas

- Obtiene un token del Auth Service

- Consulta la lista de bienes del Inventory Service

- Muestra los resultados en formato JSON en la consola

Ejemplo:

```bash

php artisan consulta:inventario admin admin123

```

## Configuración de Base de Datos

Todos los servicios utilizan SQLite para simplicidad en desarrollo:

- \*\*Auth Service\*\*: `inventario-itsc-microservices/auth-service/database/database.sqlite`

- \*\*Inventory Service\*\*: Comparte la base de datos del Auth Service (`../auth-service/database/database.sqlite`)

- \*\*Main App\*\*: `database/database.sqlite`

- \*\*Loan Service\*\*: `inventario-itsc-microservices/loan-service/database/database.sqlite`

Para inicializar las bases de datos:

1. Crear los archivos SQLite: `touch database/database.sqlite` en cada directorio

2. Ejecutar migraciones: `php artisan migrate`

3. Ejecutar seeders: `php artisan db:seed`

## Variables de Entorno

Las configuraciones principales se definen en archivos `.env` en cada servicio. Variables importantes:

- `APP\_NAME`: Nombre de la aplicación

- `APP\_ENV`: Entorno (local/production)

- `APP\_DEBUG`: Habilitar/deshabilitar debug

- `APP\_URL`: URL base de la aplicación

- `DB\_CONNECTION`: Tipo de base de datos

- `DB\_DATABASE`: Ruta al archivo SQLite

Para los servicios que consumen otros microservicios, se configuran URLs en `config/services.php` o variables de entorno específicas.

Ejemplo de una consulta

Texto

El contenido generado por IA puede ser incorrecto.  
  
  
# Microservicios Desarrollados

A continuación, se listan los microservicios que has desarrollado en este proyecto:

1. \*\*auth-service\*\* - Servicio de autenticación y gestión de usuarios.

2. \*\*inventory-service\*\* - Servicio de inventario para la gestión de bienes.

3. \*\*loan-service\*\* - Servicio de préstamos (ubicado en la raíz del proyecto).

Estos servicios están organizados principalmente bajo la carpeta `inventario-itsc-microservices/`, con `loan-service` en el directorio raíz.