



ALCALDÍA MAYOR
DE BOGOTÁ D.C.
CULTURA RECREACIÓN Y DEPORTE
Instituto Distrital de las Artes

GESTIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN

INSTRUCTIVO DOCUMENTACION API

GTI-INS-06

V.1

14/10/2026

CONTENIDO

OBJETIVO	3
ALCANCE	3
1. Aplicación del instructivo	3
2. Información General y Metadatos de la API	3
3. Esquema Global de Autenticación y Seguridad	4
3.2. Controles Mínimos Obligatorios	5
3.3. Criterios de Validación de Paquetes y Componentes de Seguridad	6
4. Estructura Estándar de Respuestas y Control de Errores	6
4.1. Esquema JSON de Respuesta Exitosa (200 OK / 201 Created)	6
4.2. Esquema JSON de Error de Validación (422 Unprocessable Entity)	6
4.3. Tabla de Códigos de Estado HTTP Institucionales	7
5. Plantilla de Especificación por Endpoint	8
6. Estándar de Documentación en Código (PHP Laravel + L5-Swagger)	9
6.1. Anotación Base del Controlador Principal (app/Http/Controllers/Controller.php)	9
6.2. Anotación de Endpoint en Controlador (ArtistasController.php)	9
7. Arquitectura y Acceso a Base de Datos	10
8. Checklist de Calidad y Criterios de Aceptación	10

Tablas e Ilustraciones

Tabla 1. Información General y Metadatos de la API	3
Tabla 2. Perfil de Seguridad de APIs	4
Tabla 3. Tabla de Códigos de Estado HTTP Institucionales.....	7
Tabla 4. Plantilla de Especificación por Endpoint.....	8
Tabla 5. Checklist de Calidad y Criterios de Aceptación	10
Ilustración 1. Esquema JSON de Respuesta Exitosa	6
Ilustración 2. Esquema JSON de Error de Validación	6
Ilustración 3. Anotación Base del Controlador Principal	9
Ilustración 4. Anotación de Endpoint en Controlador.....	9
Ilustración 5. Arquitectura y Acceso a Base de Datos	10

OBJETIVO

Definir los estándares técnicos, de ciberseguridad y de documentación técnica de las API's en Idartes, asegurando la interoperabilidad con los distintos sistemas de información de la entidad y de otras entidades externas, la homogeneidad en el manejo de respuestas y errores, y el cumplimiento de los requisitos de calidad requeridos para su despliegue en producción.

ALCANCE

Este instructivo aplica para la gestión integral de todas las APIs desarrolladas para o por el Instituto Distrital de las Artes - Idartes.

El alcance inicia con la identificación de la necesidad, levantamiento de requerimientos y diseño técnico de la API; continúa con las actividades de desarrollo, documentación, validaciones de ciberseguridad y pruebas de calidad; y finaliza con la aprobación, despliegue exitoso en el ambiente de producción y entrega formal para su operación.

Aplica a los profesionales de arquitectura de software, desarrollo, ciberseguridad y calidad de la Oficina Asesora de Planeación y Tecnologías de la Información (OAPTI), otras áreas que realicen actividades de desarrollo, así como a proveedores y firmas desarrolladoras externas que interactúan con el ecosistema tecnológico de la entidad.

Nota: Este documento debe hacer parte de la documentación técnica del sistema de información o aplicativo, por tanto, debe mantenerse actualizado a su última versión.

1. Aplicación del instructivo

El presente instructivo aplica para el diligenciamiento del formato GTI-F-26-Formato_Estandar_Documentacion_API en la versión vigente publicada en el mapa de procesos.

2. Información General y Metadatos de la API

Esta sección define el contexto institucional, versión del servicio y entornos de ejecución para garantizar la interoperabilidad de los sistemas de Idartes con las entidades del Distrito Capital. El cumplimiento de este estándar evita la acumulación de deuda técnica y facilita la adopción por parte de nuevos equipos de desarrollo.

Tabla 1. Información General y Metadatos de la API

Nombre de la API / Servicio	[Nombre del Sistema o Microservicio] (Ej. API de Registro de Artistas)
Versión de la API	v1.0.0 (Prefijo obligatorio de ruta: /api/v1/)
Estándar de Especificación	OpenAPI 3.1.0 (Compatible con JSON Schema Draft 2020-12)
Ruta de Documentación Viva	/api/documentation (Autogenerada mediante L5-Swagger)
Entornos / Servidores (servers)	• Producción: https://api.idartes.gov.co/v1 • Staging/Pruebas: https://fakeapi.idartes.gov.co/v1 • Desarrollo: http://localhost:8000/api/v1

Equipo Responsable	Oficina Asesora de Planeación y Tecnologías de la Información/ Equipo de Arquitectura y Desarrollo de Software — Idartes
---------------------------	--

Fuente: Elaboración Propia OAPT/

3. Esquema Global de Autenticación y Seguridad

Todas las APIs expuestas por Idartes deben implementar los controles de seguridad obligatorios estipulados en las políticas de interoperabilidad del Distrito:

- **Autenticación JWT / OAuth 2.0:** Implementación obligatoria mediante Laravel Passport (o Sanctum para servicios internos simples). Todas las rutas sensibles deben estar protegidas bajo el middleware auth:api.
- **Encabezado de Autorización:** Formato estándar Authorization: Bearer <token_jwt> en cada petición HTTP.
- **Rate Limiting (Control de Tasa):** Configuración de throttle:api con un límite máximo de 100 solicitudes por minuto por usuario o IP para prevenir abusos y ataques de denegación de servicio (DDoS).
- **Cifrado y Seguridad de Red:** Uso estricto de HTTPS en entornos de producción (ForceHttps). Implementación de encabezados de seguridad mediante el paquete bepsvpt/secure-headers (HSTS, CSP, X-Frame-Options, X-XSS-Protection).
- **Control de Orígenes (CORS):** Restricción explícita de ALLOWED_ORIGINS en el archivo de configuración .env para responder únicamente a dominios autorizados de la entidad.

3.1. Perfil de Seguridad de APIs

El Perfil de Seguridad de APIs es el documento de referencia definido y aprobado por el área de Ciberseguridad (Seguridad de la Información) de Idartes. Establece, según el nivel de exposición y la clasificación de la información, los controles obligatorios, sus parámetros mínimos y el inventario de componentes tecnológicos autorizados para implementarlos. Cada API debe declarar en su documentación el perfil aplicado:

Tabla 2. Perfil de Seguridad de APIs

Perfil de Seguridad aplicado	[Código y nombre del perfil] — Versión [x.y]
Nivel de exposición	☐ Pública (Internet) ☐ Interinstitucional (entidades del Distrito) ☐ Interna (red institucional)
Clasificación de la información	☐ Pública ☐ Pública clasificada ☐ Pública reservada (Ley 1712 de 2014)
Aprobado por	Área de Ciberseguridad / Oficial de Seguridad de la Información — Idartes

Fecha de aprobación	[dd/mm/aaaa]
Revisión del perfil	Mínimo una vez al año y de forma extraordinaria ante un cambio de versión mayor del framework, el fin de soporte de un componente autorizado o la publicación de una vulnerabilidad crítica.

Fuente: Elaboración Propia OAPTI

3.2. Controles Mínimos Obligatorios

Los siguientes controles son de cumplimiento obligatorio. Los valores indicados son mínimos de referencia; si el Perfil de Seguridad aprobado establece parámetros distintos, prevalece el perfil:

- **Autenticación JWT / OAuth 2.0:** Implementación obligatoria mediante Laravel Passport (o Sanctum para servicios internos simples). Todas las rutas sensibles deben estar protegidas bajo el middleware auth:api. Obligatoria en todas las rutas sensibles mediante OAuth 2.0 con tokens JWT firmados, cuyo algoritmo de firma, tiempo de expiración y mecanismo de revocación se definen en el Perfil de Seguridad. Implementación de referencia: Laravel Passport (servicios externos o interinstitucionales) o Laravel Sanctum (servicios internos simples), con las rutas protegidas bajo el middleware de autenticación (auth:api), sujeta a la validación del numeral 2.3.
- **Encabezado de Autorización:** Formato estándar Authorization: Bearer <token_jwt> en cada petición HTTP. El token nunca debe enviarse en parámetros de la URL ni registrarse en logs.
- **Rate Limiting (Control de Tasa):** Configuración de throttle:api con un límite máximo de 100 solicitudes por minuto por usuario o IP para prevenir abusos y ataques de denegación de servicio (DDoS). Límite de solicitudes por usuario, cliente o IP definido en el Perfil de Seguridad según el nivel de exposición de la API (valor de referencia: 100 solicitudes por minuto), con respuesta HTTP 429 al superarse, para prevenir abusos y ataques de denegación de servicio (DDoS). Puede aplicarse en el API Gateway / WAF o en la aplicación (throttle:api).
- **Cifrado y Seguridad de Red**[1] [2] : Uso estricto de HTTPS en entornos de producción (ForceHttps). Implementación de encabezados de seguridad mediante el paquete bepsvpt/secure-headers (HSTS, CSP, X-Frame-Options, X-XSS-Protection). Uso obligatorio de HTTPS con TLS 1.2 o superior en todos los entornos expuestos (ForceHttps en la aplicación). Los encabezados de seguridad (HSTS, Content-Security-Policy, X-Content-Type-Options, X-Frame-Options o frame-ancestors y Referrer-Policy) se configuran en el punto que defina el Perfil de Seguridad, preferiblemente de forma centralizada en el proxy inverso o API Gateway de entrada; su implementación dentro de la aplicación solo procede con un componente validado según el numeral 2.3. El encabezado X-XSS-Protection se considera obsoleto y no debe habilitarse.
- **Control de Orígenes (CORS):** Restricción explícita de ALLOWED_ORIGINS en el archivo de configuración .env para responder únicamente a dominios autorizados de la entidad. No se permite el comodín (*) en APIs que requieran autenticación.
- **Gestión de Secretos:** Llaves, credenciales, certificados y client secrets se almacenan fuera del código fuente y del repositorio (variables de entorno o gestor de secretos autorizado), conforme al Perfil de Seguridad.

3.3. Criterios de Validación de Paquetes y Componentes de Seguridad

Este estándar no obliga el uso de un paquete específico. Todo paquete, librería o componente de terceros utilizado para cumplir un control de seguridad debe contar con concepto favorable de Ciberseguridad y figurar en el inventario de componentes autorizados del Perfil de Seguridad, previa verificación de los siguientes criterios:

- **Soporte activo:** versiones publicadas o mantenimiento activo en los últimos 12 meses y sin declaratoria de abandono (abandoned) en Packagist o en su repositorio oficial.
- **Compatibilidad:** con las versiones de PHP y Laravel adoptadas por la entidad, dentro de su ciclo de soporte vigente.
- **Vulnerabilidades:** sin vulnerabilidades críticas o altas pendientes de corrección (verificación con composer audit y bases de datos CVE / GitHub Advisory).
- **Licencia:** compatible con el uso institucional (p. ej. MIT, Apache 2.0, BSD).
- **Alternativa nativa:** se prefieren las capacidades nativas del framework o los controles de infraestructura (proxy inverso, API Gateway, WAF) cuando cubran el mismo control.

Si un componente autorizado pierde soporte o presenta una vulnerabilidad crítica sin parche disponible, el equipo responsable debe registrar un plan de reemplazo aprobado por Ciberseguridad.

4. Estructura Estándar de Respuestas y Control de Errores

Para garantizar un contrato de integración predecible, todas las respuestas devueltas por los servicios web deben ceñirse a una estructura JSON unificada, procesada de forma centralizada mediante la clase app/Exceptions/Handler.php.

4.1. Esquema JSON de Respuesta Exitosa (200 OK / 201 Created)

Ilustración 1. Esquema JSON de Respuesta Exitosa

```
{
  "code": "00",
  "message": "TRANSACCIÓN EXITOSA",
  "correlationId": "76119c42-3488-4ced-8ba1-ac9cb14cdd14",
  "data": {
    "id": 105,
    "nombre": "Agrupación Filarmónica de Idartes",
    "disciplina": "Música",
    "estado": "ACTIVO"
  }
}
```

Fuente: Elaboración Propia OAPT/

4.2. Esquema JSON de Error de Validación (422 Unprocessable Entity)

Ilustración 2. Esquema JSON de Error de Validación

```
{
  "code": "05",
```

```

"message": "ERROR DE VALIDACIÓN DE DATOS",
"correlationId": "af54c139-2a3f-4a4b-bb4d-054ef0428ced",
"errors": {
  "email": [
    "El campo correo electrónico es obligatorio y debe ser válido."
  ],
  "telefono": [
    "El formato del número telefónico es inválido."
  ]
}
}

```

Fuente: Elaboración Propia OAPTI

4.3. Tabla de Códigos de Estado HTTP Institucionales

Tabla 3. Tabla de Códigos de Estado HTTP Institucionales

Código	Escenario	Descripción y acción
200 OK	Consulta o actualización exitosa	La solicitud se procesó y se retorna la información solicitada o el recurso actualizado.
201 Created	Creación exitosa	El recurso se creó. Incluye el encabezado Location y el recurso en el cuerpo.
202 Accepted	Procesamiento asíncrono	La solicitud se recibió y se procesa en segundo plano (colas o jobs). Retorna cómo consultar el estado.
204 No Content	Eliminación exitosa	La operación salió bien y no se retorna cuerpo.
400 Bad Request	Solicitud malformada	No se puede interpretar la petición: JSON inválido, parámetros de consulta con formato incorrecto.
401 Unauthorized	Sin autenticación	Token JWT ausente, inválido o vencido en el encabezado Authorization.
403 Forbidden	Sin autorización	El usuario autenticado no tiene los roles o permisos requeridos.
404 Not Found	Recurso no encontrado	El recurso no existe, o existe pero el usuario no debe conocer su existencia.
405 Method Not Allowed	Método no permitido	El verbo HTTP no está soportado en el endpoint. Incluye el encabezado Allow.

Código	Escenario	Descripción y acción
409 Conflict	Conflicto de estado	Registro duplicado (NIT, correo) o el recurso está en un estado que impide la operación.
415 Unsupported Media Type	Formato no soportado	El Content-Type no es application/json o el tipo de archivo enviado no es válido.
422 Unprocessable Content	Error de validación	La petición se interpretó, pero los datos no cumplen las reglas de la Form Request. Retorna el detalle por campo.
429 Too Many Requests	Límite de peticiones	Se superó el rate limit. Incluye el encabezado Retry-After.
500 Internal Server Error	Error interno	Excepción no controlada. Se registra en logs con un identificador; nunca se expone el stack trace.
503 Service Unavailable	Servicio no disponible	Mantenimiento programado o dependencia caída. Incluye Retry-After cuando se conoce.

Fuente: Elaboración Propia OAPTI

5. Plantilla de Especificación por Endpoint

Esta plantilla debe ser diligenciada de manera formal para cada recurso expuesto en el catálogo de servicios de Idartes:

Tabla 4. Plantilla de Especificación por Endpoint

Método HTTP & Ruta Endpoint	POST /api/v1/artistas
Resumen y Descripción	Crear y registrar un nuevo artista en la plataforma institucional.
Módulo / Etiqueta (tags)	Artistas
Esquema de Seguridad	Bearer Token JWT (auth:api) — Requerido
Parámetros de Entrada	• Path Parameters: N/A • Query Parameters: N/A
Cuerpo de la Solicitud (Request Body)	Clase: CreateArtistaRequest (Form Request dedicado) Formato: application/json Campos: - nombre (string, requerido, max:100) - email (string, requerido, email, único) - disciplina_id (integer, requerido, existe en v_disciplinas)
Respuestas Estándar	• 201 Created: ArtistaResource (Objeto creado)

(Responses)	• 401 Unauthorized: Token ausente o expirado • 422 Unprocessable Entity: Detalle de errores de validación
--------------------	--

Fuente: Elaboración Propia OAPTI

6. Estándar de Documentación en Código (PHP Laravel + L5-Swagger)

Para evitar discrepancias entre la documentación técnica y la implementación real, Idartes exige el uso del paquete darkaonline/l5-swagger. La documentación viva se genera automáticamente mediante el comando `php artisan l5-swagger:generate` leyendo las anotaciones PHP en los controladores.

6.1. Anotación Base del Controlador Principal (app/Http/Controllers/Controller.php)

Ilustración 3. Anotación Base del Controlador Principal

```
<?php

namespace App\Http\Controllers;

/**
 * @OA\Info(
 *   title="Idartes API REST",
 *   version="1.0.0",
 *   description="API REST para la interoperabilidad de los sistemas del Instituto Distrital de las Artes"
 * )
 * @OA\Server(
 *   url="https://api.idartes.gov.co/v1",
 *   description="Servidor de Producción"
 * )
 * @OA\Server(
 *   url="https://fakeapi.idartes.gov.co/v1",
 *   description="Servidor de Pruebas / Staging"
 * )
 * @OA\SecurityScheme(
 *   securityScheme="bearerAuth",
 *   type="http",
 *   scheme="bearer",
 *   bearerFormat="JWT"
 * )
 */
abstract class Controller
{
}
```

Fuente: Elaboración Propia OAPTI

6.2. Anotación de Endpoint en Controlador (ArtistasController.php)

Ilustración 4. Anotación de Endpoint en Controlador

```
/**
 * @OA\Post(
 *   path="/api/v1/artistas",
 *   summary="Crear nuevo artista",
 *   tags={"Artistas"},
 *   security={{"bearerAuth":{}}},

```

```

* @OA\RequestBody(
*   required=true,
*   @OA\JsonContent(ref="#/components/schemas/CreateArtistaRequest")
* ),
* @OA\Response(
*   response=201,
*   description="Artista creado exitosamente",
*   @OA\JsonContent(ref="#/components/schemas/ArtistaResource")
* ),
* @OA\Response(response=401, description="Token no suministrado o inválido"),
* @OA\Response(response=422, description="Error de validación de datos")
* )
*/
public function store(CreateArtistaRequest $request): ArtistaResource
{
    $artista = $this->service->create($request->validated());
    return new ArtistaResource($artista);
}

```

Fuente: Elaboración Propia OAPTI

7. Arquitectura y Acceso a Base de Datos

Las APIs en Idartes se construyen siguiendo los principios de Clean Architecture para garantizar bajo acoplamiento y alta testabilidad:

Ilustración 5. Arquitectura y Acceso a Base de Datos

Capa de Arquitectura	Componentes Laravel	Responsabilidad Principal
Presentación (API)	Controllers, Form Requests, API Resources	Recepción de solicitudes HTTP, validación declarativa y transformación de respuestas JSON.
Aplicación	Services (Lógica de Negocio)	Orquestación de la lógica de negocio y llamadas a repositorios.
Dominio	Models (Eloquent)	Representación de entidades del dominio y definición de relaciones.
Infraestructura	Repositories, Middleware, Oracle Driver	Acceso a base de datos (Oracle yajra/laravel-oci8), caché en Redis e intercepción de peticiones.

Fuente: Elaboración Propia OAPTI

8. Checklist de Calidad y Criterios de Aceptación

Antes de autorizar el despliegue de cualquier API a entornos de producción en Idartes, el proyecto debe cumplir con criterios de validación de los DESARROLLOS DE LOS SISTEMAS DE INFORMACIÓN Y APLICATIVOS:

Tabla 5. Checklist de Calidad y Criterios de Aceptación

Categoría	Criterio de Verificación	Estado
Documentación	Especificación OpenAPI viva accesible en /api/documentation y actualizada.	Obligatorio
Seguridad	Autenticación y autorización implementadas conforme al	Obligatorio

Categoría	Criterio de Verificación	Estado
	Perfil de Seguridad aprobado por Ciberseguridad (numerales 2.1 y 2.2).	
Seguridad	Rate Limiting configurado según el Perfil de Seguridad (valor de referencia: 100 req/min).	Obligatorio
Seguridad	HTTPS (TLS 1.2 o superior) y encabezados de seguridad configurados según el Perfil de Seguridad.	Obligatorio
Seguridad	Paquetes y componentes de seguridad validados y autorizados (numeral 2.3); composer audit sin vulnerabilidades críticas o altas.	Opcional
Calidad de Código	Validación estricta de peticiones con clases Form Request dedicadas.	Obligatorio
Calidad de Código	Pruebas unitarias e integración con cobertura superior al 80% (php artisan test --coverage).	Obligatorio
Operaciones	Logging estructurado JSON con Monolog y sin exposición de stack traces.	Obligatorio

Fuente: Elaboración Propia OAPTl



ALCALDÍA MAYOR
DE BOGOTÁ D.C.
CULTURA RECREACIÓN Y DEPORTE
Instituto Distrital de las Artes

CONTROL DE CAMBIOS

VERSIÓN	FECHA DE APROBACIÓN	DESCRIPCIÓN DE CAMBIOS REALIZADOS
1	2026-10-14	Emisión inicial

CONTROL DE APROBACIÓN

ESTADO	FECHA	NOMBRE	CARGO
ELABORÓ	2026-10-14	JONATHAN GONZALEZ BOLANOS	ENLACE MIPG
REVISÓ	2026-10-14	MARYURY FORERO BOHORQUEZ	REFERENTE MIPG
APROBÓ	2026-10-14	DANIEL SANCHEZ ROJAS	LIDER DE PROCESO
AVALÓ	2026-10-14	DANIEL SANCHEZ ROJAS	JEFE DE LA OFICINA ASESORA DE PLANEACIÓN Y TECNOLOGÍAS DE LA INFORMACIÓN

COLABORADORES

NOMBRE
CARLOS ANDRES MENDEZ ESPITIA