

DOCUMENTACIÓN TÉCNICA API

DIVIPOLA

Sistema de Información Geográfica y Poblacional

Observatorio Amazonas - Gobernación del Amazonas

Autor: Julián Andrés Ramírez
Versión: 1.0 | Fecha: Diciembre 2025
Estado: Implementación Activo

Tabla de contenido

Tabla de contenido.....	2
1. INTRODUCCIÓN Y MARCO INSTITUCIONAL	4
1.1 Propósito del Documento.....	4
1.2 Marco Normativo	4
1.3 Alcance Técnico.....	4
1.4 Configuración del Entorno Productivo	4
2. ARQUITECTURA TÉCNICA DEL SISTEMA	4
2.1 Patrón Arquitectónico	4
2.2 Esquema de Base de Datos	5
3. API DE DEPARTAMENTOS.....	5
3.1 Endpoints Disponibles	5
3.2 Especificación Detallada.....	6
4. API DE CENTROS POBLADOS	6
4.1 Endpoints Principales	6
4.2 Endpoint Especializado: Coordenadas Geográficas	7
4.3 Parámetros de Búsqueda Avanzada.....	8
5. MODELOS DE DATOS Y VALIDACIONES	8
5.1 Modelo CentroPoblado	8
6. FLUJO DE CAPTURA Y PROCESAMIENTO	9
6.1 Diagrama de Flujo del Procesamiento.....	9
6.2 Transformación de Datos	9
7. MANEJO DE ERRORES Y RESPUESTAS	10
7.1 Códigos de Estado HTTP Implementados.....	10
7.2 Estructura Estándar de Respuestas	10
8.1 Medidas de Seguridad Implementadas	11
8.2 Optimizaciones de Rendimiento	11

9. EJEMPLOS DE IMPLEMENTACIÓN	12
9.1 Consulta de Centros Poblados por Municipio	12
9.2 Creación de Nuevo Centro Poblado	12
10. APÉNDICES TÉCNICOS.....	12
10.1 Variables de Entorno Requeridas	12
10.2 Scripts de Instalación y Configuración	13
10.3 Contacto Técnico	13

1. INTRODUCCIÓN Y MARCO INSTITUCIONAL

1.1 Propósito del Documento

El presente documento constituye la documentación técnica oficial del Sistema de Información DIVIPOLA (División Político-Administrativa) del Observatorio Amazonas, desarrollado para la Gobernación del Departamento de Amazonas, Colombia.

1.2 Marco Normativo

El sistema está alineado con las disposiciones del DANE (Departamento Administrativo Nacional de Estadística) para la codificación territorial colombiana y cumple con los estándares técnicos establecidos por el Marco de Arquitectura Empresarial para la Gestión de TI del Estado Colombiano.

1.3 Alcance Técnico

- Gestión integral de datos territoriales del Amazonas
- APIs REST para consulta y manipulación de datos geográficos
- Integración con sistemas de información geográfica
- Servicios web para aplicaciones gubernamentales

1.4 Configuración del Entorno Productivo

Servidor: `http://localhost:4000`
Base de Datos Principal: `observatorio`
Base de Datos DIVIPOLA: `divipola`
Servidor MySQL: `localhost:3306`
Codificación: `UTF-8 (utf8mb4)`
Protocolo: `HTTPS (Producción)`

2. ARQUITECTURA TÉCNICA DEL SISTEMA

2.1 Patrón Arquitectónico

El sistema implementa una arquitectura de capas (Layered Architecture) siguiendo los principios de Domain-Driven Design (DDD):

Capa	Responsabilidad	Tecnología	Archivos
Presentación	Endpoints HTTP y validación de entrada	Express.js	/routes/*.js
Controladores	Orquestación de peticiones	Node.js	/controllers/*.js
Servicios	Lógica de negocio	JavaScript ES6+	/services/*.js
Repositorios	Acceso a datos	MySQL2	/repositories/*.js
Modelos	Entidades de dominio	Clases JS	/models/*.js

2.2 Esquema de Base de Datos

Tabla	Propósito	Campos Principales	Relaciones
departamentos	División administrativa nivel 1	codigo_departamento, nombre_departamento	1:N municipios
municipios	División administrativa nivel 2	codigo_municipio, nombre_municipio, codigo_departamento	N:1 departamentos, 1:N centros_poblados
centro_poblado	Asentamientos humanos	codigo_centro_poblado, nombre_centro_poblado, latitud, longitud	N:1 municipios

3. API DE DEPARTAMENTOS

3.1 Endpoints Disponibles

Método	Endpoint	Descripción	Parámetros	Estado
GET	/api/departamentos	Listar departamentos	orderBy, order, limit, offset	200
GET	/api/departamentos/:codigo	Obtener por código	codigo (path)	200/404
GET	/api/departamentos/search	Búsqueda por texto	q, limit, offset	200

POST	/api/departamentos	Crear departamento	Body JSON	201/400
PUT	/api/departamentos/:codigo	Actualizar departamento	codigo (path), Body JSON	200/404
DELETE	/api/departamentos/:codigo	Eliminar departamento	codigo (path)	204/404

3.2 Especificación Detallada

GET /api/departamentos

Propósito: Obtiene lista paginada y ordenada de todos los departamentos colombianos.

Parámetros de Consulta (Query Parameters):

- orderBy: Campo de ordenamiento (default: 'nombre_departamento')
- order: Dirección ASC|DESC (default: 'ASC')
- limit: Registros por página (1-100, default: 50)
- offset: Desplazamiento para paginación (default: 0)

Ejemplo de Petición:

GET /api/departamentos?orderBy=nombre_departamento&order=ASC&limit=10&offset=0

Respuesta Exitosa (HTTP 200):

```
{
  "success": true,
  "message": "Departamentos obtenidos correctamente",
  "data": [
    {
      "codigoDepartamento": "91",
      "nombreDepartamento": "AMAZONAS",
      "timestamps": {
        "creado": "2024-01-15T10:30:00.000Z",
        "actualizado": "2024-01-15T10:30:00.000Z"
      }
    }
  ],
  "pagination": {
    "total": 32,
    "limit": 10,
    "offset": 0,
    "hasNext": true,
    "hasPrevious": false,
    "totalPages": 4,
    "currentPage": 1
  },
  "timestamp": "2024-12-14T19:45:30.123Z"
}
```

4. API DE CENTROS POBLADOS

4.1 Endpoints Principales

Método	Endpoint	Funcionalidad	Casos de Uso
--------	----------	---------------	--------------

GET	/api/centros-poblados	Listado completo	Consultas generales, reportes
GET	/api/centros-poblados/:codigo	Consulta individual	Detalles específicos
GET	/api/centros-poblados/municipio/:codigo	Por municipio	Consultas territoriales
GET	/api/centros-poblados/search	Búsqueda avanzada	Filtros múltiples
GET	/api/centros-poblados/coordenadas	Con datos geográficos	Sistemas GIS, mapas
GET	/api/centros-poblados/estadísticas	Métricas del sistema	Dashboards, monitoreo

4.2 Endpoint Especializado: Coordenadas Geográficas

GET /api/centros-poblados/coordenadas

Descripción: Retorna exclusivamente los centros poblados que poseen coordenadas geográficas válidas, optimizado para aplicaciones de mapas y sistemas GIS.

Ejemplo de Respuesta:

```
{
  "success": true,
  "message": "Centros poblados con coordenadas obtenidos correctamente",
  "data": [
    {
      "codigo": 91001023,
      "nombre": "ASENTAMIENTO HUMANO TAKANA KM 11",
      "tipo": "CP",
      "coordenadas": {
        "latitud": -4.099833,
        "longitud": -69.9402
      },
      "municipio": {
        "codigo": 91001,
        "nombre": "LETICIA"
      },
      "timestamps": {
        "creado": "2024-01-15T08:15:00.000Z",
        "actualizado": "2024-12-14T14:30:00.000Z"
      }
    }
  ],
  "metadata": {
    "totalConCoordenadas": 58,
    "totalSinCoordenadas": 142,
    "porcentajeCompleto": 29.0
  }
}
```

4.3 Parámetros de Búsqueda Avanzada

GET /api/centros-poblados/search

Parámetro	Tipo	Descripción	Ejemplo
q	String	Término de búsqueda en nombre	"leticia"
municipio	String	Código del municipio	"91001"
tipo	String	Tipo de centro poblado	"CP", "CAS", "CORP"
conCoordenadas	Boolean	Solo con coordenadas	true, false

5. MODELOS DE DATOS Y VALIDACIONES

5.1 Modelo CentroPoblado

```
class CentroPoblado {
  constructor(data = {}) {
    this.codigo_centro_poblado = data.codigo_centro_poblado || null;
    this.nombre_centro_poblado = data.nombre_centro_poblado || '';
    this.tipo = data.tipo || null;
    this.latitud = data.latitud || null;
    this.longitud = data.longitud || null;
    this.codigo_municipio = data.codigo_municipio || null;
    this.created_at = data.created_at || null;
    this.updated_at = data.updated_at || null;
  }

  // Validación de código
  static isValidCodigo(codigo) {
    return codigo && !isNaN(codigo) && parseInt(codigo) > 0;
  }

  // Validación de nombre
  static isValidNombre(nombre) {
    return nombre &&
      typeof nombre === 'string' &&
      nombre.trim().length >= 2 &&
      nombre.trim().length <= 200;
  }

  // Validación de coordenadas
  static isValidLatitud(latitud) {
    return !isNaN(latitud) && latitud >= -90 && latitud <= 90;
  }

  static isValidLongitud(longitud) {
    return !isNaN(longitud) && longitud >= -180 && longitud <= 180;
  }
}
```

5.2 Reglas de Validación por Campo

Campo	Tipo	Restricciones	Observaciones
codigo_centro_poblado	INTEGER	Positivo, único	Clave primaria
nombre_centro_poblado	VARCHAR(200)	2-200 caracteres, no nulo	Texto en mayúsculas
tipo	VARCHAR(5)	CP, CAS, CORP, INSP, VER	Según clasificación DANE
latitud	DECIMAL(10,8)	-90.0 a 90.0	Grados decimales (WGS84)
longitud	DECIMAL(11,8)	-180.0 a 180.0	Grados decimales (WGS84)
codigo_municipio	INTEGER	Debe existir en tabla municipios	Clave foránea

6. FLUJO DE CAPTURA Y PROCESAMIENTO

6.1 Diagrama de Flujo del Procesamiento

1. **Recepción de Petición:** El controlador recibe la petición HTTP
2. **Validación de Parámetros:** Se verifican parámetros de entrada y autorización
3. **Instanciación de Modelo:** Se crea objeto del modelo correspondiente
4. **Validación de Datos:** El modelo ejecuta validaciones de negocio
5. **Transformación:** Los datos se adaptan al formato de base de datos
6. **Operación en Repositorio:** Se ejecuta la operación CRUD correspondiente
7. **Formateo de Respuesta:** El controlador estructura la respuesta JSON
8. **Envío de Respuesta:** Se retorna la respuesta HTTP con código apropiado
- 9.

6.2 Transformación de Datos

```
// Ejemplo: Procesamiento de Centro Poblado
async procesarCentroPoblado(datosEntrada) {
  try {
    // 1. Instanciación del modelo
    const centroPoblado = new CentroPoblado(datosEntrada);
```

```
// 2. Validaciones de negocio
const erroresValidacion = centroPoblado.validate();
if (erroresValidacion.length > 0) {
  throw new ValidationError('Datos inválidos', erroresValidacion);
}

// 3. Transformación para BD
const objetoBD = centroPoblado.toDatabaseObject();

// 4. Persistencia
const resultado = await this.repository.create(objetoBD);

// 5. Transformación de respuesta
return centroPoblado.toResponseObject(resultado);

} catch (error) {
  this.handleError(error);
}
```

7. MANEJO DE ERRORES Y RESPUESTAS

7.1 Códigos de Estado HTTP Implementados

Código	Estado	Descripción	Casos de Aplicación
200	OK	Operación exitosa	Consultas, actualizaciones exitosas
201	Created	Recurso creado	Creación de nuevos registros
204	No Content	Eliminación exitosa	Operaciones DELETE exitosas
400	Bad Request	Petición malformada	Parámetros inválidos, JSON malformado
404	Not Found	Recurso no encontrado	Entidad inexistente, endpoint inválido
409	Conflict	Conflicto de recursos	Duplicación de códigos únicos
422	Unprocessable Entity	Errores de validación	Datos que no cumplen reglas de negocio
500	Internal Server Error	Error interno	Fallos de conexión, errores no controlados

7.2 Estructura Estándar de Respuestas

Respuesta Exitosa:

```
{
  "success": true,
  "message": "Descripción de la operación realizada",
  "data": { /* Datos de respuesta */ },
  "pagination": { /* Solo en listados paginados */ },
  "metadata": { /* Información adicional */ },
  "timestamp": "2024-12-14T15:45:30.123Z"
}
```

Respuesta de Error:

```
{
  "success": false,
  "error": "Tipo de error general",
  "message": "Descripción detallada del error",
  "errors": [
    {
      "field": "nombre_campo",
      "code": "VALIDATION_ERROR",
      "message": "Descripción específica del error"
    }
  ],
  "code": "ERROR_CODE_SYSTEM",
  "timestamp": "2024-12-14T15:45:30.123Z",
  "requestId": "req_1234567890abcdef"
}
```

8. SEGURIDAD Y CONSIDERACIONES TÉCNICAS

8.1 Medidas de Seguridad Implementadas

- **Validación de Entrada:** Todos los datos de entrada son validados antes del procesamiento
- **Sanitización de Datos:** Se eliminan caracteres peligrosos y se normalizan los datos
- **Prepared Statements:** Se utilizan consultas parametrizadas para prevenir inyección SQL
- **CORS Configurado:** Control de acceso desde orígenes autorizados
- **Rate Limiting:** Límites de velocidad para prevenir abuso de APIs
- **Logging de Auditoría:** Registro detallado de operaciones críticas

8.2 Optimizaciones de Rendimiento

Área	Implementación	Beneficio	Métrica
Paginación	Límite máximo de 500 registros	Reduce carga de red	Tiempo de respuesta < 2s
Índices de BD	En códigos y nombres	Acelera consultas	Búsquedas < 100ms
Connection Pool	Pool de conexiones MySQL	Reutilización eficiente	50 conexiones concurrentes

Caché	Cache en memoria para consultas frecuentes	Reduce acceso a BD	Cache TTL: 300s
-------	--	--------------------	-----------------

9. EJEMPLOS DE IMPLEMENTACIÓN

9.1 Consulta de Centros Poblados por Municipio

```
// Cliente JavaScript
async function obtenerCentrosPobladosLeticia() {
  try {
    const response = await fetch(
      'http://localhost:4000/api/centros-poblados/municipio/91001?limit=20&orderBy=nombre_centro_poblado'
    );

    if (!response.ok) {
      throw new Error(`Error HTTP: ${response.status}`);
    }

    const data = await response.json();

    if (data.success) {
      console.log(`Se encontraron ${data.data.length} centros poblados`);
      return data.data;
    } else {
      console.error('Error en la respuesta:', data.message);
      return [];
    }
  } catch (error) {
    console.error('Error en la petición:', error.message);
    return [];
  }
}
```

9.2 Creación de Nuevo Centro Poblado

```
// Ejemplo con cURL
curl -X POST http://localhost:4000/api/centros-poblados \
  -H "Content-Type: application/json" \
  -d '{
    "codigo": "91001999",
    "nombre": "NUEVA COMUNIDAD INDÍGENA",
    "tipo": "CP",
    "latitud": -4.1234,
    "longitud": -69.8765,
    "codigo_municipio": "91001"
  }'
```

10. APÉNDICES TÉCNICOS

10.1 Variables de Entorno Requeridas

```
# Configuración de Base de Datos Principal
DB_HOST=localhost
```

```
DB_USER=admin
DB_PASSWORD=admin
DB_NAME=observatorio
DB_PORT=3306
DB_CHARSET=utf8mb4
```

```
# Configuración de Base de Datos DIVIPOLA
DB_DIVPOLA_HOST=localhost
DB_DIVPOLA_USER=admin
DB_DIVPOLA_PASSWORD=admin
DB_DIVPOLA_NAME=divipola
DB_DIVPOLA_PORT=3306
DB_DIVPOLA_CHARSET=utf8mb4
```

```
# Configuración de API
VITE_API_BASE=http://localhost:4000
NODE_ENV=development
JWT_SECRET=clave_secret_jwt
CORS_ORIGIN=http://localhost:3000
```

10.2 Scripts de Instalación y Configuración

```
# Instalación de dependencias
npm install express cors mysql2 dotenv multer jsonwebtoken bcryptjs

# Configuración de base de datos
mysql -u root -p < database/create_divipola_tables.sql
mysql -u root -p < database/create_centro_poblado_table.sql

# Ejecución del servidor
npm start
```

10.3 Contacto Técnico

- **Entidad:** Gobernación del Departamento de Amazonas
- **Sistema:** Observatorio Amazonas - DIVIPOLA
- **Versión:** 1.0 (Diciembre 2024)
- **Estado:** Implementación Activa
- **Soporte:** Dirección de Ciencia, Tecnología e Innovación

Este documento constituye la especificación técnica oficial del Sistema DIVIPOLA del Observatorio Amazonas. La información contenida es de carácter técnico y está destinada al personal especializado en desarrollo e implementación de sistemas de información geográfica gubernamentales.

Documento clasificación: Privado | Versión: 1.0 | Última actualización: Diciembre 2025

CONTROL DE DOCUMENTO

Versión	Fecha	Autor	Descripción
1.0	Diciembre 2025	Julian Andres Ramirez 	Creación inicial del documento