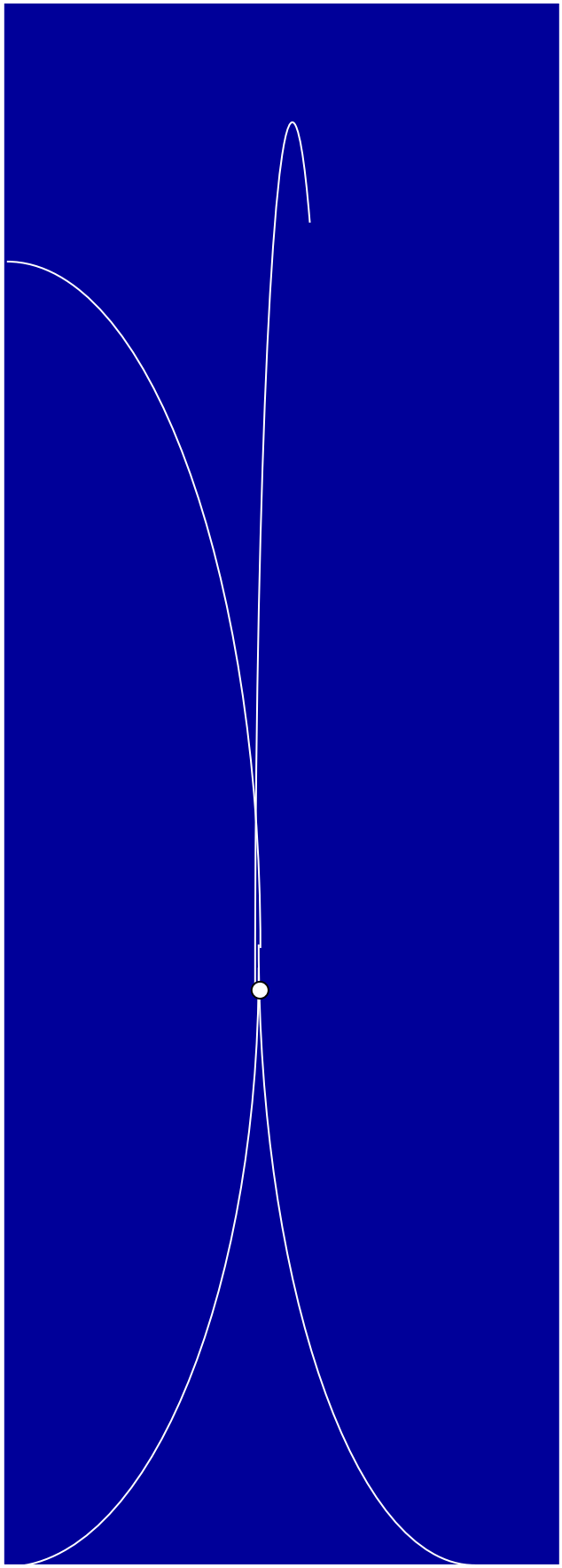


Programa educativo
INGENIERÍA EN SISTEMAS COMPUTACIONALES

Grupo
8avo

Nombre de la materia
ARQUITECTURA DE SERVICIOS

Nombre de los integrantes
ALVARADO GODÍNEZ ULISES
BAÑALES GUZMÁN JUAN HUMBERTO
RAMÍREZ GODÍNEZ JESÚS

Número y nombre del trabajo
ACTIVIDAD 2. 2DO AVANCE DEL PROYECTO

Unidad # 2
TECNOLOGÍAS PARA EL DESARROLLO DE SERVICIOS.

Nombre del Profesor
MIS ROBERTO SUÁREZ ZINZÚN

Fecha
14 de marzo de 2025

1. Diseño del modelo de datos

Se ha optado por el uso de una base de datos NoSQL, ya que consideramos que es la opción más adecuada para gestionar la información relacionada con los cultivos y las diversas acciones que se realizan sobre ellos. A diferencia de las bases de datos relacionales, en este caso no es necesario manejar transacciones complejas que requieran una estructura rígida de tablas y relaciones estrictas.

Además, nuestra experiencia previa con bases de datos NoSQL, en particular con MongoDB durante el semestre pasado, ha sido un factor determinante en esta decisión. Queremos consolidar los conocimientos adquiridos y profundizar aún más en el manejo de este tipo de bases de datos, explorando sus ventajas en términos de escalabilidad, flexibilidad en el modelado de datos y eficiencia en la consulta y almacenamiento de información.

A continuación, se presenta el modelo documental de la base de datos, diseñado con base en las entidades definidas en la Unidad 1:

Colección Cultivos

Cultivos.json > ...

```json
{
    "_id": "ObjectId()",
    "nomCultivo": "String",
    "fechaSiembra": "Date",
    "fechaCosechaEst": "Date",
    "fechaCosechaReal": "Date",
    "areaCultivo": "number",
    "tipoSuelo": "String",
    "estadoActual": "String",
    "idUsuario": "ref(Usuarios._id)",
    "aplicacionesInsumo":[
        {
            "fechaAplicacion": "Date",
            "cantAplicada": "number",
            "metodoAplicacion": "String",
            "estadoAplicacion": "String",
            "idInsumo": "ref(Insumos._id)",
            "idUsuario": "ref(Usuarios._id)"
        }
    ],
    "riegos": [
        {
            "fechaRiego": "Date",
            "cantAgua": "number",
            "metodoRiego": "String",
            "duracionRiego": "number",
            "idUsuario": "ref(Usuarios._id)"
        }
    ],
    "ubicacion": [
        {
        "nombreUbicacion": "String",
        "coordenadas": {
            "latitud": "number",
            "longitud": "number"
        },
        "superficie": "number",
        "clima": "String",
        "tipoSuelo": "String",
        "accesoAgua": "boolean"
    }
    ]
}
```

Colección Seguimiento_Cultivo

Seguimiento_Cultivos.json > ...

```json
{
    "_id": "ObjectId()",
    "fechaRevision": "Date",
    "estadoCultivo": "String",
    "observaciones": ["String"],
    "recomendaciones": ["String"],
    "idCultivo": "ref(Cultivos._id)",
    "idUsuario": "ref(Usuarios._id)",
    "datosClimaticos": [
        {
            "fechaMedicion": "Date",
            "temperatura": "String",
            "humedad": "String",
            "precipitacion": "String",
            "viento": "String"
        }
    ],
    "alertas": [
        {
            "tipoAlerta": "String",
            "descripcionAlerta": "String",
            "fechaGenerada": "Date",
            "horaGenerada": "String",
            "estadoAlerta": "String"
        }
    ]
}
```

Colección Usuarios

Usuarios.json > ...

```json
{
    "_id": "ObjectId()",
    "nombre": "String",
    "telefono": "String",
    "estatus": "boolean",
    "email": "String",
    "password": "String",
    "rol": "String // Valores posibles: 'Agricultor', 'Ingeniero', 'Personal'"
}
```

Colección Actividades_Usuario

ActividadesUsuarios.json > ...

```json
{
    "_id": "ObjectId()",
    "actividad": "String",
    "fechaActividad": "Date",
    "estatus": "boolean",
    "idCultivo": "ref(Cultivos._id)",
    "idUsuario": "ref(Usuarios._id)"
}
```

Colección Insumos

Insumos.json > ...

```json
{
    "_id": "ObjectId()",
    "nombreInsumo": "String",
    "tipoInsumo": "String",
    "cantDisponible": "number",
    "unidadMedida": "String"
}
```

Colección Historial_Suelo

Historial_Suelo.json > ...

```json
{
    "_id": "ObjectId()",
    "fechaMedicion": "Date",
    "pH": "number",
    "nutrientes": [
        {
            "nutriente": "String",
            "valor": "number",
            "unidad": "String"
        }
    ],
    "observaciones": ["String"],
    "idCultivo": "ref(Cultivos._id)",
    "idUsuario": "ref(Usuarios._id)"
}
```

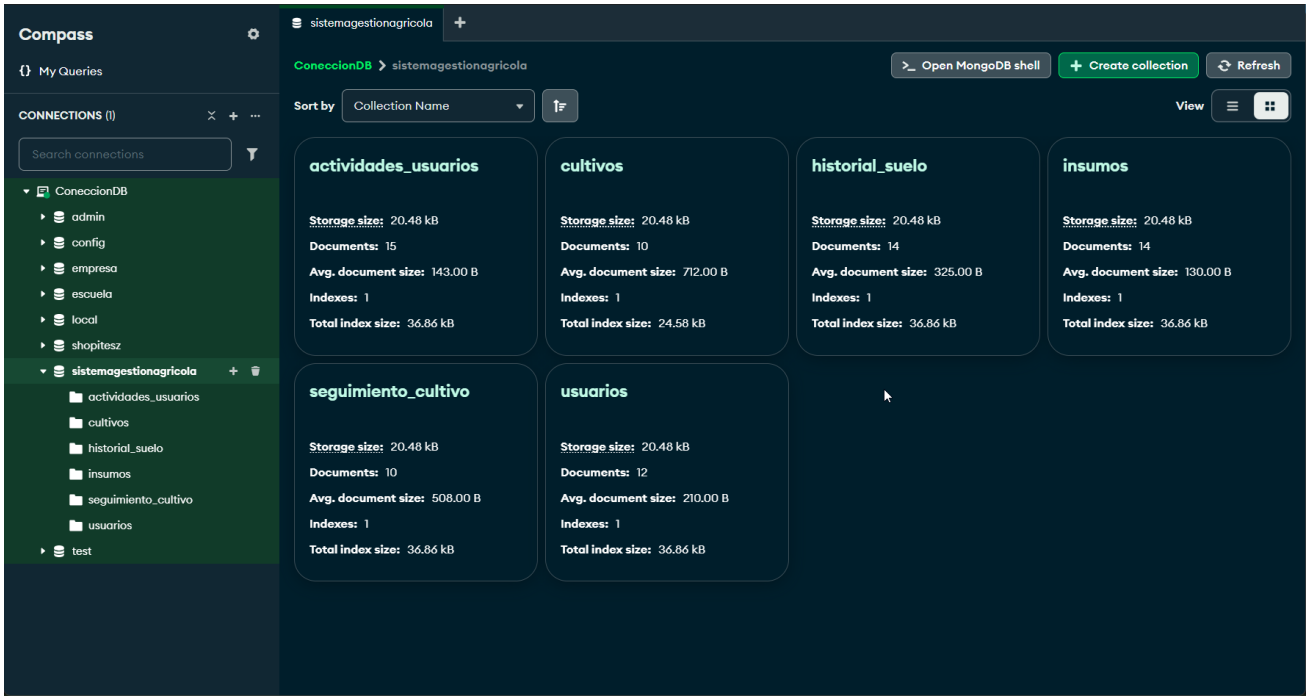
2. Script de la BD

A continuación, se presenta la evidencia de que la base de datos ha sido creada en MongoDB, así como los comandos utilizados para crearla, todo en base a las colecciones mostradas anteriormente:

```
//Comando para crear base de datos
use sistemagestionagricola

//Comandos para crear las colecciones una vez que esté en la bd de sistemagestionagricola
db.createCollection("usuarios")
db.createCollection("actividades_usuarios")
db.createCollection("cultivos")
db.createCollection("historial_suelo")
db.createCollection("insumos")
db.createCollection("seguimiento_cultivo")
```

A continuación, se muestra una captura de la base de datos creada en mongo con las colecciones y datos ya ingresados:



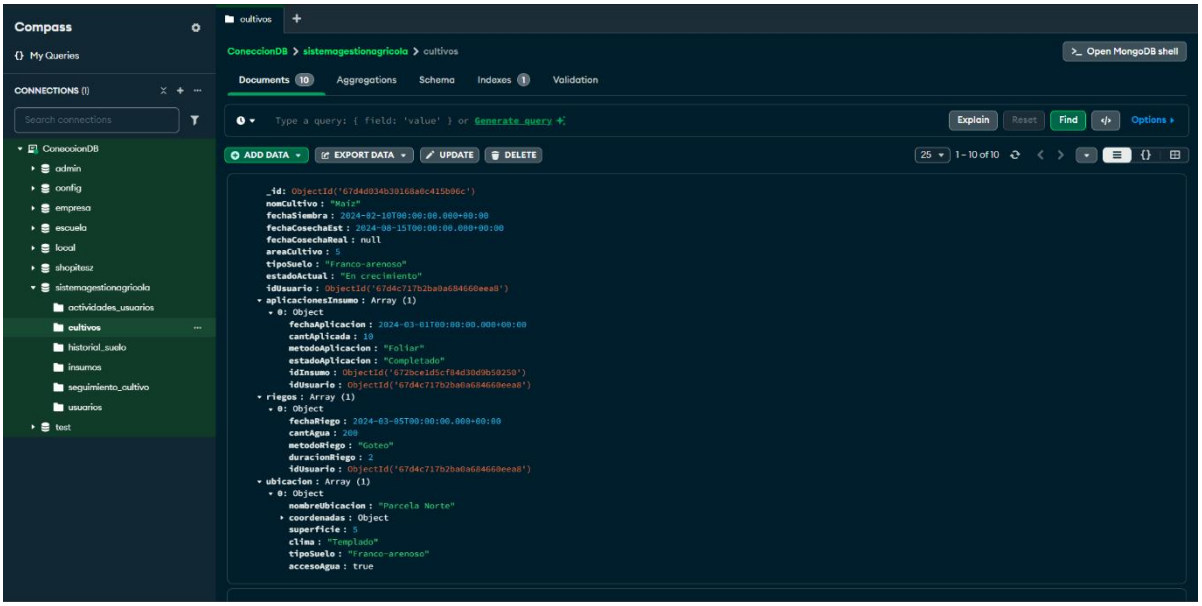
3. Carga inicial de datos

A continuación, se muestra evidencia de que se han cargado datos a las colecciones en MongoDB, ingresando 10 documentos a cada colección.

Inserción de 10 documentos a la colección cultivos:

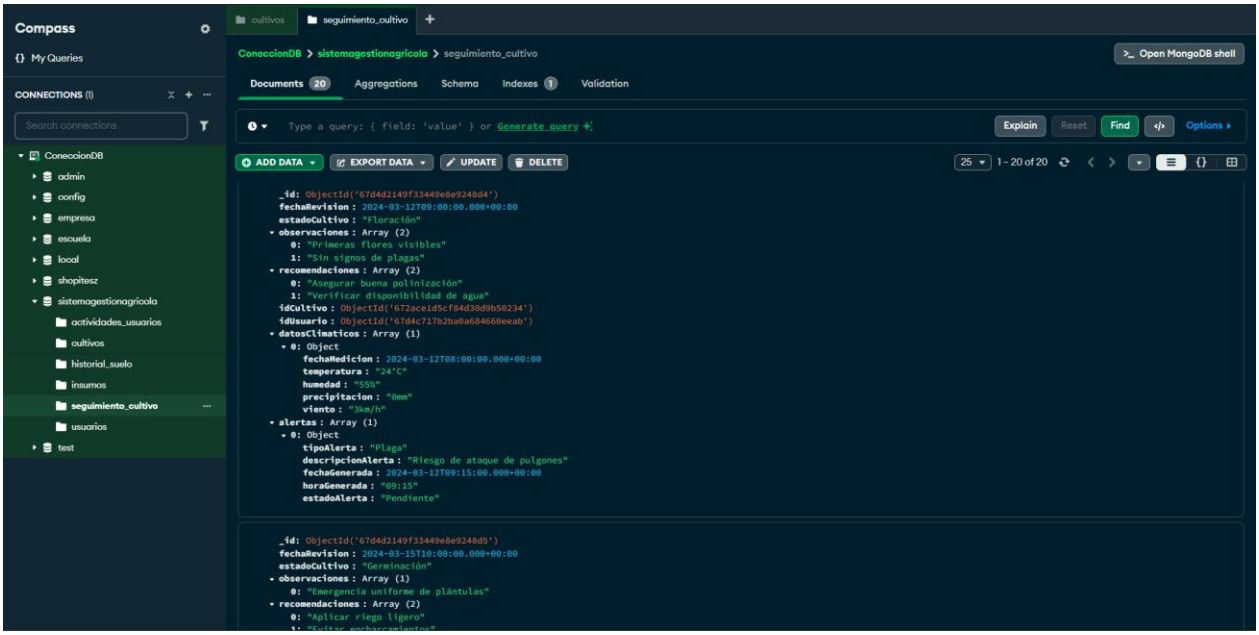
```
                    "latitud": 19.8765,
                    "longitud": -98.7654
                },
                "superficie": 4.0,
                "clima": "Templado seco",
                "tipoSuelo": "Arenoso",
                "accesoAgua": false
            }
        ]
    }

]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d4d034b30168a0c415b06c'),
    '1': ObjectId('67d4d034b30168a0c415b06d'),
    '2': ObjectId('67d4d034b30168a0c415b06e'),
    '3': ObjectId('67d4d034b30168a0c415b06f'),
    '4': ObjectId('67d4d034b30168a0c415b070'),
    '5': ObjectId('67d4d034b30168a0c415b071'),
    '6': ObjectId('67d4d034b30168a0c415b072'),
    '7': ObjectId('67d4d034b30168a0c415b073'),
    '8': ObjectId('67d4d034b30168a0c415b074'),
    '9': ObjectId('67d4d034b30168a0c415b075')
  }
}
```



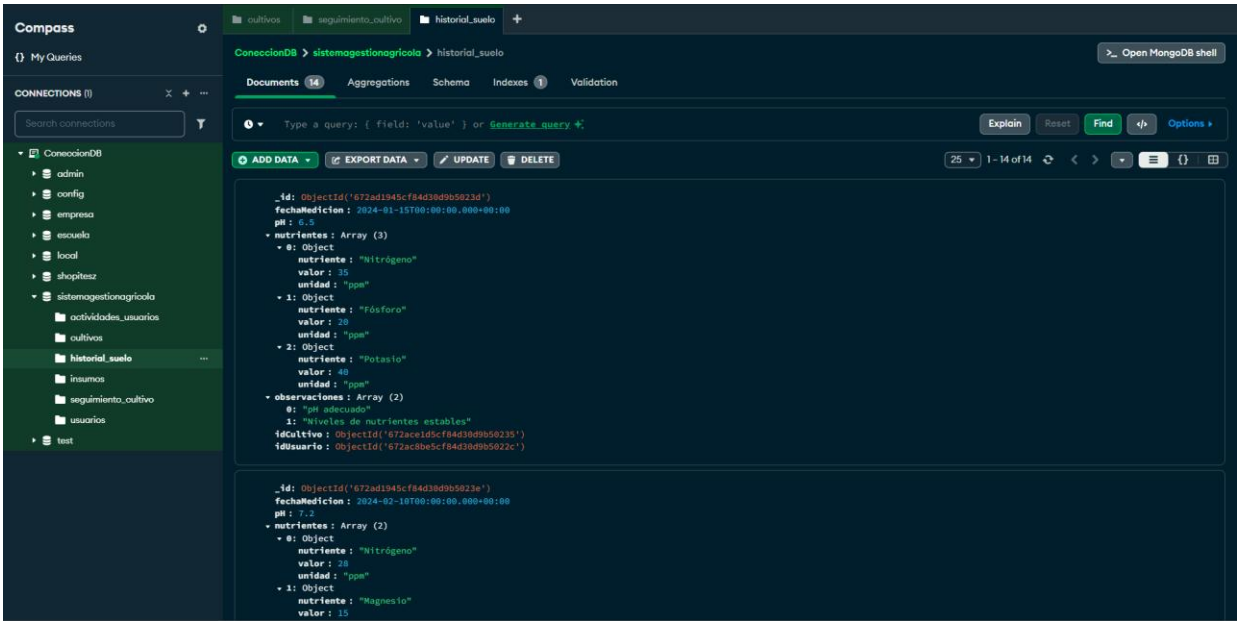
Inserción de 10 documentos a la colección seguimiento_cultivo:

```
        ],
        "alertas": [
            {
                "tipoAlerta": "Suelo",
                "descripcionAlerta": "Necesidad de recuperación de nutrientes",
                "fechaGenerada": ISODate("2024-04-02T12:00:00Z"),
                "horaGenerada": "12:00",
                "estadoAlerta": "Pendiente"
            }
        ]
    }
]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d4d2149f33449e8e9248d3'),
    '1': ObjectId('67d4d2149f33449e8e9248d4'),
    '2': ObjectId('67d4d2149f33449e8e9248d5'),
    '3': ObjectId('67d4d2149f33449e8e9248d6'),
    '4': ObjectId('67d4d2149f33449e8e9248d7'),
    '5': ObjectId('67d4d2149f33449e8e9248d8'),
    '6': ObjectId('67d4d2149f33449e8e9248d9'),
    '7': ObjectId('67d4d2149f33449e8e9248da'),
    '8': ObjectId('67d4d2149f33449e8e9248db'),
    '9': ObjectId('67d4d2149f33449e8e9248dc')
  }
}
```



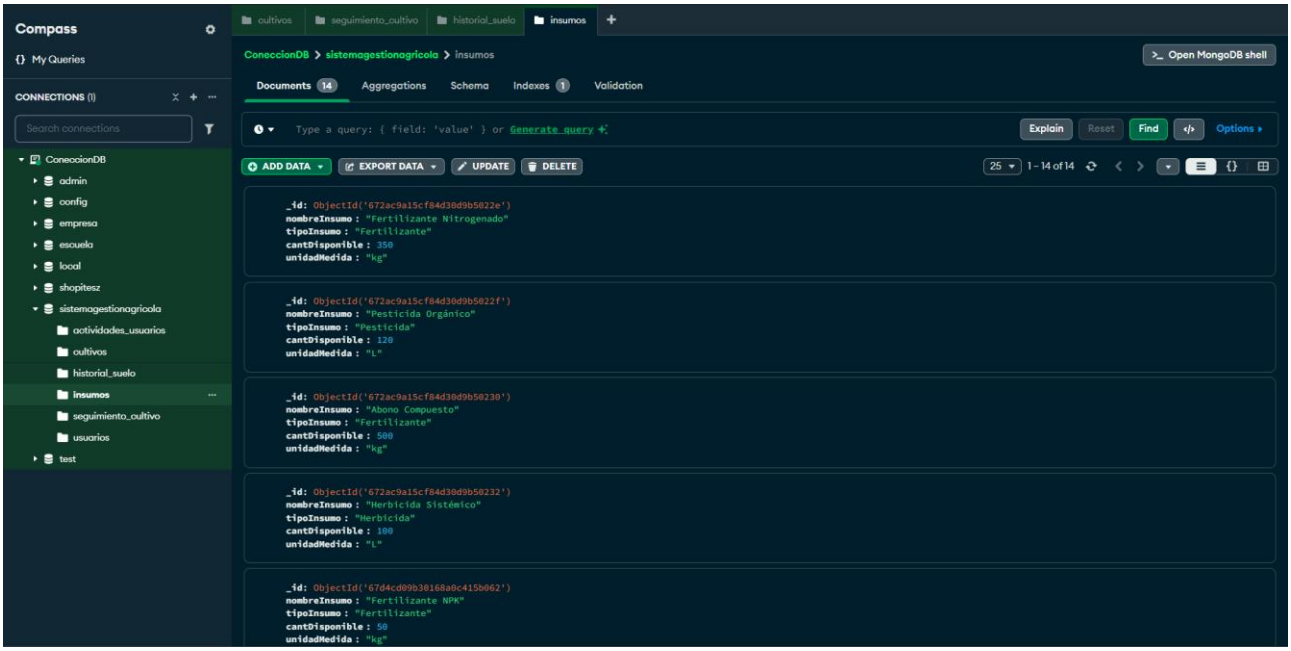
Inserción de 10 documentos a la colección historial_suelo:

```
    {
        "fechaMedicion": new Date("2025-03-18T12:00:00Z"),
        "pH": 6.7,
        "nutrientes": [
            { "nutriente": "Calcio", "valor": 20, "unidad": "ppm" },
            { "nutriente": "Magnesio", "valor": 12, "unidad": "ppm" }
        ],
        "observaciones": ["Los nutrientes son adecuados para el cultivo."],
        "idCultivo": ObjectId("672ace1d5cf84d30d9b50234"),
        "idUsuario": ObjectId("67d4c717b2ba0a684660eeab")
    }
]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d4ca33336e60d1020aa67c'),
    '1': ObjectId('67d4ca33336e60d1020aa67d'),
    '2': ObjectId('67d4ca33336e60d1020aa67e'),
    '3': ObjectId('67d4ca33336e60d1020aa67f'),
    '4': ObjectId('67d4ca33336e60d1020aa680'),
    '5': ObjectId('67d4ca33336e60d1020aa681'),
    '6': ObjectId('67d4ca33336e60d1020aa682'),
    '7': ObjectId('67d4ca33336e60d1020aa683'),
    '8': ObjectId('67d4ca33336e60d1020aa684'),
    '9': ObjectId('67d4ca33336e60d1020aa685')
  }
}
```



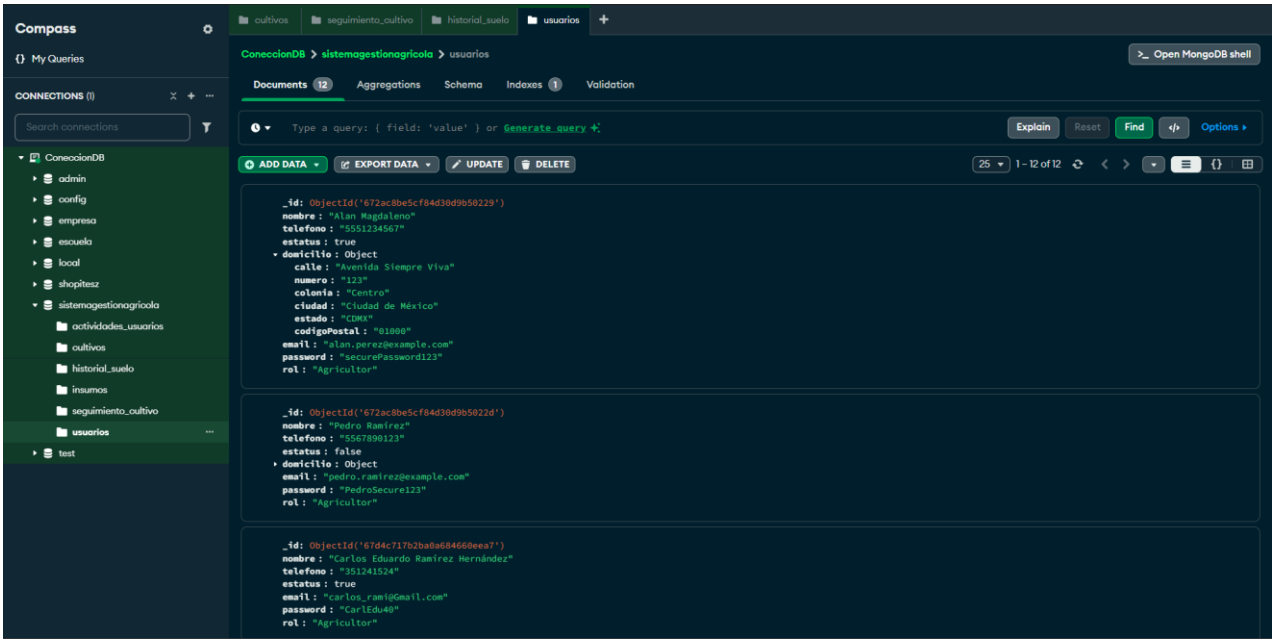
Inserción de 10 documentos a la colección insumos:

```
    {
        "nombreInsumo": "Plástico para Invernadero",
        "tipoInsumo": "Material",
        "cantDisponible": 200,
        "unidadMedida": "m2"
    }
]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d4cd09b30168a0c415b062'),
    '1': ObjectId('67d4cd09b30168a0c415b063'),
    '2': ObjectId('67d4cd09b30168a0c415b064'),
    '3': ObjectId('67d4cd09b30168a0c415b065'),
    '4': ObjectId('67d4cd09b30168a0c415b066'),
    '5': ObjectId('67d4cd09b30168a0c415b067'),
    '6': ObjectId('67d4cd09b30168a0c415b068'),
    '7': ObjectId('67d4cd09b30168a0c415b069'),
    '8': ObjectId('67d4cd09b30168a0c415b06a'),
    '9': ObjectId('67d4cd09b30168a0c415b06b')
  }
}
```



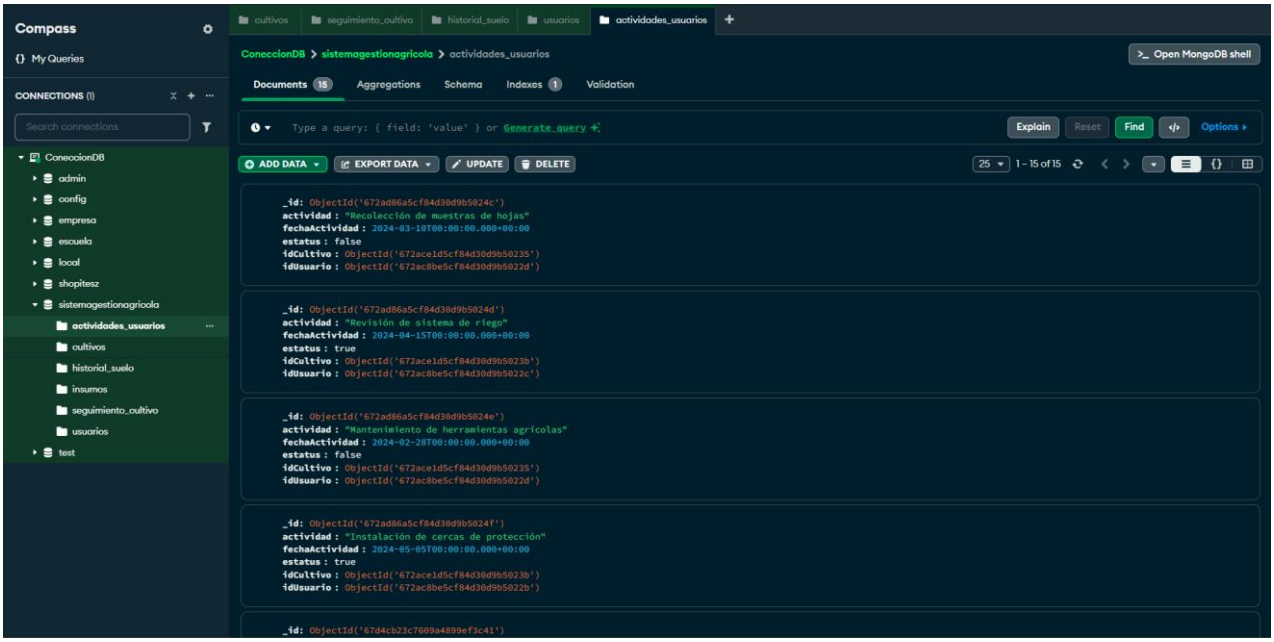
Inserción de 10 documentos a la colección usuarios:


```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d4c717b2ba0a684660eea7'),
    '1': ObjectId('67d4c717b2ba0a684660eea8'),
    '2': ObjectId('67d4c717b2ba0a684660eea9'),
    '3': ObjectId('67d4c717b2ba0a684660eeaa'),
    '4': ObjectId('67d4c717b2ba0a684660eeab'),
    '5': ObjectId('67d4c717b2ba0a684660eeac'),
    '6': ObjectId('67d4c717b2ba0a684660eead'),
    '7': ObjectId('67d4c717b2ba0a684660eeae'),
    '8': ObjectId('67d4c717b2ba0a684660eeaf'),
    '9': ObjectId('67d4c717b2ba0a684660eeb0')
  }
}
```

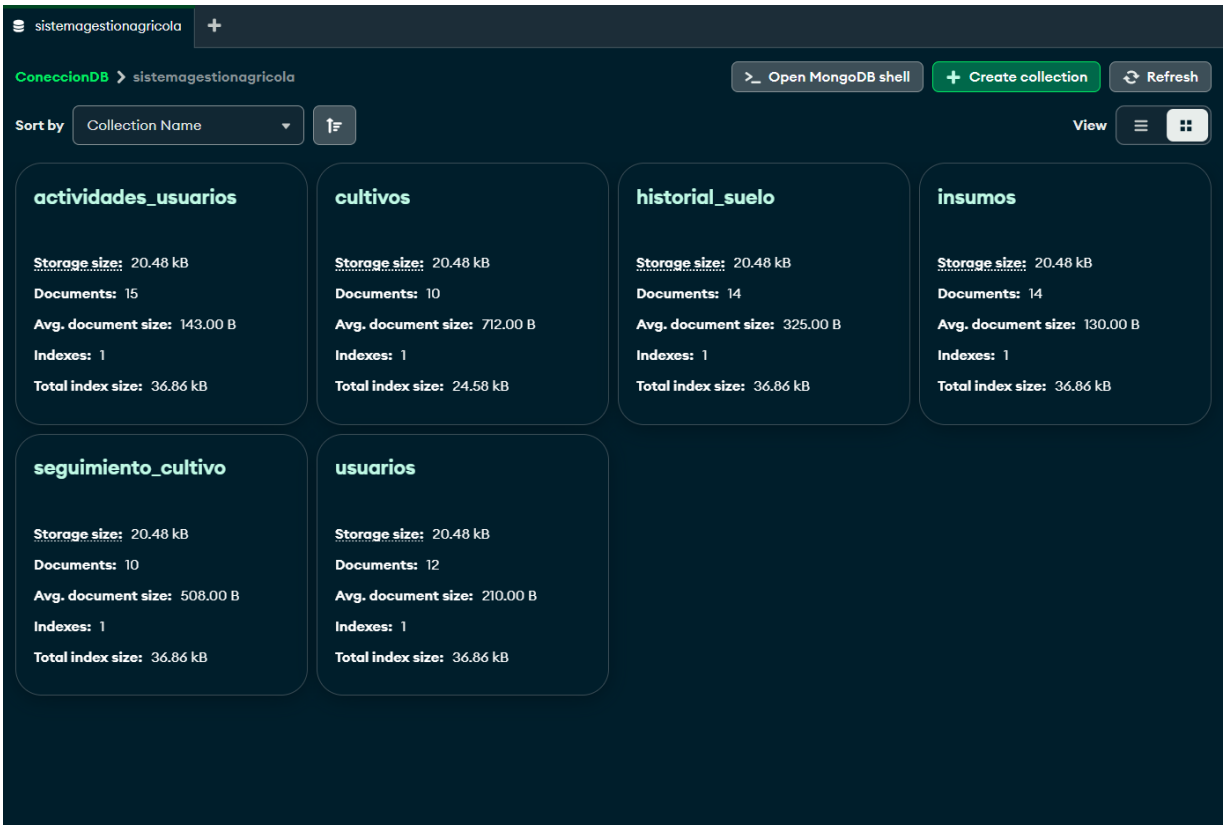



Inserción de 10 documentos a la colección actividades_usuarios:

```
    {
        "actividad": "Supervisión del crecimiento",
        "fechaActividad": ISODate("2024-03-19T15:00:00Z"),
        "estatus": true,
        "idCultivo": ObjectId("672ace1d5cf84d30d9b50234"),
        "idUsuario": ObjectId("67d4c717b2ba0a684660eeab")
    }
]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d4cc79c7609a4899ef3c42'),
    '1': ObjectId('67d4cc79c7609a4899ef3c43'),
    '2': ObjectId('67d4cc79c7609a4899ef3c44'),
    '3': ObjectId('67d4cc79c7609a4899ef3c45'),
    '4': ObjectId('67d4cc79c7609a4899ef3c46'),
    '5': ObjectId('67d4cc79c7609a4899ef3c47'),
    '6': ObjectId('67d4cc79c7609a4899ef3c48'),
    '7': ObjectId('67d4cc79c7609a4899ef3c49'),
    '8': ObjectId('67d4cc79c7609a4899ef3c4a'),
    '9': ObjectId('67d4cc79c7609a4899ef3c4b')
  }
}
```



Vista general de las colecciones y su cantidad de documentos en la base de datos



4. Definición de los requerimientos de servicios:

Lista de Servicios:

Insumos

Tipo de Servicio: Es un servicio de entidad, ya que su principal función es gestionar el ciclo de vida de los insumos dentro del sistema, realizando operaciones CRUD (crear, leer, actualizar y eliminar) sobre ellos. No contiene lógica de negocio compleja ni reglas específicas de procesamiento. Su responsabilidad es exponer y mantener la información de los insumos de manera consistente y estructurada para otros servicios que la requieran.

Responsable: Jesús Ramírez Godínez

Operaciones expuestas:

- **registrarInsumo() –** Permite crear un nuevo insumo en el sistema, almacenando sus datos relevantes como nombre, categoría, cantidad disponible y cualquier otra información necesaria.
- **editarInsumo() –** Modifica los datos de un insumo existente, permitiendo actualizar atributos como nombre, descripción, cantidad o cualquier otro campo necesario.
- **borrarInsumo() –** Elimina un insumo del sistema, ya sea de forma lógica (cambiando su estado) o física (eliminándolo completamente de la base de datos).
- **consultarInsumo() –** Recupera la información de un insumo específico a partir de su identificador, devolviendo detalles como nombre, cantidad disponible y otros atributos.
- **consultarListaInsumos() –** Retorna un listado de todos los insumos registrados en el sistema, permitiendo filtrado o paginación si es necesario.

Descripción de las operaciones:

Atributo	Descripción
Operación	**registrarInsumo()**
Actor(es)	Administrador y gestores de insumos
URL	/insumos/agregar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar un nuevo insumo en el sistema. Se deben validar los siguientes aspectos: • Que todos los campos obligatorios estén presentes. • Que la cantidad sea un valor numérico mayor o igual a cero. • Que no exista un insumo con el mismo nombre en el sistema (si aplica restricción de unicidad). • Que tipoInsumo deba pertenecer a una lista predefinida de tipos válidos (ej. "Insecticidas", "Fertilizantes", etc.). Si todas las validaciones se cumplen, el insumo se almacena en la base de datos y se devuelve una respuesta con éxito.

Entrada	{
	"nombreInsumo": "String",
	"tipoInsumo": "String",
	"cantidadDisponible": "float",
	"unidadMedida": "String"
	}

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**editarInsumo()**
Actor(es)	Administrador y gestores de insumos
URL	/insumos/editar/{id}
Método HTTP	PUT
Lógica de Negocio	La operación permite actualizar los datos de un insumo existente en el sistema. Se deben realizar las siguientes validaciones: • Verificar que el insumo con el id proporcionado exista en la base de datos. • Validar que los campos modificados cumplan con las reglas de negocio (ej. nombre no vacío, cantidad mayor o igual a cero). • Si se cambia el nombre del insumo, verificar que no exista otro insumo con el mismo nombre (si aplica restricción de unicidad).

	Si las validaciones son correctas, se actualizan los datos en la base de datos y se devuelve una respuesta de éxito.

Entrada	{
	"nombreInsumo": "String",
	"tipoInsumo": "String",
	"cantidadDisponible": "float",
	"unidadMedida": "String"
	}

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**borrarInsumo()**
Actor(es)	Administrador
URL	/insumos/eliminar/{id} (El {id} representa el identificador único del insumo a eliminar).
Método HTTP	DELETE
Lógica de Negocio	La operación permite eliminar un insumo del sistema. Se deben realizar las siguientes validaciones: • Verificar que el insumo con el id proporcionado exista en la base de datos. • Si el insumo existe, proceder con su eliminación. • Si la eliminación es exitosa, se devuelve un mensaje de confirmación. • Si el insumo no existe, se devuelve un mensaje de error.
Entrada	No se requieren datos en el cuerpo de la petición. El id del insumo se pasa como parte de la URL (/insumos/eliminar/{id}).

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**consultarInsumo()**
Actor(es)	Administrador, gestores de insumos y agricultores
URL	/insumos/{id} (El {id} representa el identificador único del insumo a consultar).
Método HTTP	GET

Lógica de Negocio	La operación permite obtener los detalles de un insumo específico a partir de su identificador (id). Se deben realizar las siguientes validaciones: • Verificar que el insumo con el id proporcionado exista en la base de datos. Si el insumo existe, se devuelve su información. Si no, se devuelve un mensaje de error.
Entrada	No se requieren datos en el cuerpo de la petición. El id del insumo se pasa como parte de la URL (/insumos/{id}).

Salida	{
	"mensaje": "String",
	"idInsumo": "int",
	"nombreInsumo": "String",
	"tipoInsumo": "String",
	"cantidadDisponible": "float",
	"unidadMedida": "String"
	}

Atributo	**Descripción**
Operación	**consultarListaInsumo()**
Actor(es)	Administrador, gestores de insumos y agricultores
URL	/insumos
Método HTTP	GET
Lógica de Negocio	La operación permite obtener un listado de todos los insumos registrados en el sistema. Puede incluir funcionalidades de filtrado (si es necesario). • Si se incluye un parámetro de filtrado en la solicitud (por ejemplo, por tipo de insumo o nombre), el sistema filtra los resultados según el parámetro proporcionado. • Si no se encuentran insumos que cumplan los criterios, se devuelve una lista vacía.
Entrada	No se requieren datos en el cuerpo de la solicitud. Los parámetros opcionales de filtrado o paginación se incluyen como parámetros de consulta en la URL si aplica.
Salida	Lista de los insumos registrados en la base de datos.

Aplicación de Insumos

Tipo de Servicio: Servicio de Entidad. Este servicio se encarga de gestionar la aplicación de insumos en los cultivos, permitiendo registrar, modificar, eliminar y consultar las aplicaciones realizadas. Dado que su función principal es administrar información estructurada sobre la aplicación de insumos, su clasificación corresponde a un servicio de entidad. No realiza cálculos complejos ni proporciona funcionalidades auxiliares, sino que se centra en la persistencia y administración de datos relacionados con el uso de insumos agrícolas.

Responsable: Ulises Alvarado Godínez

Operaciones expuestas:

- **registrarAplicacionInsumo()**: Permite registrar la aplicación de un insumo, asegurando que se proporcionen datos como la fecha de aplicación, cantidad utilizada, método de aplicación y referencias al insumo y usuario responsable.
- **editarAplicacionInsumo()**: Permite modificar los datos de una aplicación de insumo existente, validando los cambios y asegurando la coherencia con las reglas de negocio.
- **eliminarAplicacionInsumo()**: Permite eliminar una aplicación de insumo, previa verificación de la existencia del registro.
- **consultarAplicacionInsumo()**: Recupera la información detallada de una aplicación de insumo específica, incluyendo todos sus atributos y referencias.
- **consultarListaAplicacionInsumo()**: Obtiene una lista de todas las aplicaciones de insumos registradas en el sistema para un cultivo en particular, permitiendo aplicar filtros y paginación si es necesario.

Descripción de las operaciones:

Atributo	Descripción
Operación	**registrarAplicacionInsumo()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/aplicacionInsumos/agregar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar una aplicación de insumos a un cultivo del sistema. Para lo cual se debe: • Verificar que se incluyan los campos: fechaAplicacion, cantAplicada, metodoAplicacion, estadoAplicacion, idInsumo y idUsuario. • Confirmar que fechaAplicacion sea una fecha válida y que no sea posterior a la fecha actual. • Validar que cantAplicada sea un número positivo mayor a cero. • Asegurar que metodoAplicacion y estadoAplicacion sean strings válidos dentro de los valores permitidos. • Comprobar que idInsumo haga referencia a un insumo existente y que idUsuario corresponda a un usuario registrado en el sistema. Si todas las validaciones se cumplen, registrar la aplicación del insumo en la base de datos y responder con un mensaje de éxito de lo contrario con uno de error.

Entrada	{
	"fechaAplicacion": Date
	"cantAplicada": number
	"metodoAplicacion": String
	"estadoAplicacion": String
	"idInsumo": ObjectId
	"idUsuario": ObjectId
	}

Salida	{
	"mensaje": "String",
	}

Atributo	Descripción
Operación	**editarApliacionInsumo()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/aplicacionInsumos/editar/{idAplicacionInsumo}
Método HTTP	PUT
Lógica de Negocio	La operación permite actualizar los datos de una aplicación de insumo a un cultivo existente en el sistema. Se deben realizar las siguientes validaciones: • Confirmar que los registros identificados por {id} existan en la base de datos. • Revisar cada campo enviado y validar su tipo y coherencia: fechaAplicacion debe ser una fecha válida y no futura. cantAplicada debe ser un número positivo. metodoAplicacion y estadoAplicacion deben cumplir con los valores permitidos. idInsumo e idUsuario deben hacer referencia a registros existentes. • Actualizar el registro en la base de datos y enviar el mensaje correspondiente.

Entrada	{
	"fechaAplicacion": Date
	"cantAplicada": number
	"metodoAplicacion": String
	"estadoAplicacion": String
	"idInsumo": ObjectId
	"idUsuario": ObjectId
	}

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**eliminarAplicacionInsumo()**
Actor(es)	Administrador
URL	cultivos/{idCultivo}/aplicacionInsumos/eliminar/{idAplicacionInsumo}
Método HTTP	DELETE
Lógica de Negocio	La operación permite eliminar la aplicación de insumos de un cultivo del sistema. Para lo cual se debe: • Confirmar que los registros identificados por {id} existan en la base de datos. • Proceder con la eliminación y registrar la operación • Enviar el mensaje de éxito o fallo en la operación realizada.
Entrada	No se requieren datos en el cuerpo de la petición.

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**consultarAplicacionInsumo()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/aplicacionInsumos/consultar/{idAplicacionInsumo}
Método HTTP	GET
Lógica de Negocio	La operación permite obtener los detalles de una aplicación de insumo específica a partir de su identificador (id) a un cultivo especifico. Se deben realizar las siguientes validaciones: • Verificar que la aplicación del insumo con el id proporcionado exista en la base de datos, así como también el cultivo al que fue aplicado.

	• Si existen, se devuelve su información. Si no, se devuelve un mensaje de error.
Entrada	No se requieren datos en el cuerpo de la petición. El id de la aplicación insumo se pasa como parte de la URL, así como también el del cultivo.

Salida	{
	"fechaAplicacion": Date
	"cantAplicada": number
	"metodoAplicacion": String
	"estadoAplicacion": String
	"idInsumo": ObjectId
	"idUsuario": ObjectId
	"mensaje": String
	}

Atributo	**Descripción**
Operación	**consultarListaAplicacionInsumo()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/aplicacionInsumos
Método HTTP	GET
Lógica de Negocio	La operación permite obtener un listado de todas las aplicaciones de insumos registrados en el sistema al cultivo seleccionado. • Verificación de Existencia: Confirmar que los {id} enviado en la URL corresponda a un registro existente. • Consulta y Recuperación de Datos: Recuperar y retornar todos los atributos de la aplicación de insumo.
Entrada	No se requieren datos en el cuerpo de la solicitud. Los parámetros se incluyen en la URL.
Salida	[{ "fechaAplicacion": Date "cantAplicada": number

	"metodoAplicacion": String "estadoAplicacion": String "idInsumo": ObjectId "idUsuario": ObjectId }]

ActividadesUsuario

Tipo de Servicio: Es un servicio de entidad, ya que su principal función es gestionar el ciclo de vida de las actividades realizadas por los usuarios dentro del sistema, permitiendo operaciones CRUD (crear, leer, actualizar y eliminar). Su responsabilidad es exponer y mantener la información de las actividades de manera estructurada y accesible para otros servicios que la requieran.

Responsable: Jesús Ramírez Godínez

Operaciones expuestas:

registrarActividad() – Permite crear una nueva actividad en el sistema, registrando información relevante como el usuario que la realizó, la fecha y la descripción de la acción.

editarActividad() – Modifica los datos de una actividad existente, permitiendo actualizar atributos como la descripción, la fecha o cualquier otro campo necesario.

borrarActividad() – Elimina una actividad del sistema, ya sea de forma lógica (cambiando su estado) o física (eliminándola completamente de la base de datos).

consultarActividad() – Recupera la información de una actividad específica a partir de su identificador, devolviendo detalles como el usuario que la realizó, la fecha y la acción registrada.

consultarListaActividades() – Retorna un listado de todas las actividades registradas en el sistema, permitiendo filtrado o paginación si es necesario.

Descripción de las operaciones:

Atributo	Descripción
Operación	**registrarActividad()**
Actor(es)	Administrador y supervisores
URL	/actividades/registrar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar una nueva actividad en el sistema. Se deben realizar las siguientes validaciones: • Verificar que los campos requeridos (actividad, fechaActividad, estatus) estén presentes y no sean nulos. • Validar que actividad no esté vacía.

	• Asegurar que fechaActividad tenga un formato válido de fecha. • Verificar que estatus tenga un valor permitido (ejemplo: "Pendiente", "Completada", "Cancelada"). Si todas las validaciones son correctas, se almacena la nueva actividad en la base de datos y se devuelve un mensaje de éxito junto con la información registrada. • Verificar que el cultivo existe en la base de datos. • Verificar que el usuario existe en la base de datos.

Entrada	{
	"actividad": "string",
	"fechaActividad": "Date",
	"estatus": "String",
	"idCultivo": "ref (Cultivos) ",
	"idUsuario": "ref (Usuarios) "
	}

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**editarActividad()**
Actor(es)	Administrador y supervisores
URL	/actividades/editar/{idActividad}
Método HTTP	PUT
Lógica de Negocio	La operación permite modificar una actividad existente en el sistema. Se deben realizar las siguientes validaciones: • Verificar que la actividad con el idActividad proporcionado exista en la base de datos. • Validar que los campos a modificar (actividad, fechaActividad, estatus) sean válidos y no estén vacíos. • Asegurar que fechaActividad tenga un formato correcto de fecha. • Verificar que estatus tenga un valor permitido (Ejemplo: "Pendiente", "Completada", "Cancelada"). • Verificar que el cultivo existe en la base de datos. • Verificar que el usuario existe en la base de datos. Si todas las validaciones son correctas, se actualiza la actividad en la base de datos y se devuelve un mensaje de éxito junto con la información modificada.
Entrada	{ "actividad": "string", "fechaActividad": "Date", "estatus": "String",

	"idCultivo": "ref (Cultivos) ", "idUsuario": "ref (Usuarios) " }

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**borrarActividad()**
Actor(es)	Administrador
URL	/actividades/eliminar/{idActividad}
Método HTTP	DELETE
Lógica de Negocio	La operación permite eliminar una actividad del sistema. Se deben realizar las siguientes validaciones: • Verificar que la actividad con el idActividad proporcionado exista en la base de datos. • Si la actividad existe, proceder con su eliminación. • Si la eliminación es exitosa, se devuelve un mensaje de confirmación. • Si la actividad no existe, se devuelve un mensaje de error.
Entrada	No se requieren datos en el cuerpo de la petición. El idActividad se pasa como parte de la URL (/actividades/eliminar/{idActividad}).

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**consultarActividad()**
Actor(es)	Administrador y supervisores
URL	/actividades/{idActividad}
Método HTTP	GET
Lógica de Negocio	La operación permite obtener la información detallada de una actividad específica en el sistema. Se deben realizar las siguientes validaciones: • Verificar que la actividad con el idActividad proporcionado exista en la base de datos. • Si la actividad existe, devolver su información detallada.

	• Si la actividad no existe, devolver un mensaje de error indicando que no se encontró.
Entrada	No se requieren datos en el cuerpo de la petición. El idActividad se pasa como parte de la URL (/actividades/{idActividad}).

Salida	{
	"mensaje": "String",
	"idActividad": "int",
	"actividad": "String",
	"fechaActividad": "Date",
	"estatus": "String",
	"idCultivo": "ref (Cultivos) ",
	"idUsuario": "ref (Usuarios) "
	}

Atributo	**Descripción**
Operación	**consultarListaActividades()**
Actor(es)	Administrador y supervisores
URL	/actividades
Método HTTP	GET
Lógica de Negocio	La operación permite obtener un listado de todas las actividades registradas en el sistema. Se pueden aplicar filtros opcionales como fecha o estatus. • Consultar todas las actividades almacenadas en la base de datos. • Permitir filtros opcionales (ejemplo: estatus=Pendiente, fechaActividad=2025-03-14). • Si existen actividades, devolver la lista con sus detalles. • Si no hay actividades registradas, devolver una lista vacía.
Entrada	No se requieren datos obligatorios en el cuerpo de la petición.
Salida	Lista de los actividades de usuario registrados en la base de datos.

Usuarios

Tipo de Servicio: El servicio de Usuarios es un servicio de entidad y utilidad, ya que gestiona el ciclo de vida de los usuarios dentro del sistema mediante operaciones CRUD (crear, leer, actualizar y eliminar), asegurando que su información esté estructurada y accesible. Además, cumple una función de utilidad al proporcionar funcionalidades adicionales como autenticación, asignación de roles y recuperación de contraseñas, facilitando la seguridad y gestión del acceso al sistema.

Responsable: Jesús Ramírez Godínez

Operaciones expuestas:

registrarUsuario() – Permite crear un nuevo usuario en el sistema, almacenando información como nombre, correo electrónico, contraseña y rol asignado.

editarUsuario() – Modifica la información de un usuario existente, permitiendo actualizar datos personales o credenciales.

borrarUsuario() – Elimina un usuario del sistema, ya sea de forma lógica (desactivándolo) o física (eliminándolo completamente de la base de datos).

consultarUsuario() – Recupera la información de un usuario específico a partir de su identificador.

consultarListaUsuarios() – Retorna un listado de todos los usuarios registrados en el sistema, permitiendo filtros o paginación.

iniciarSesion() – Permite a un usuario autenticarse en el sistema mediante su correo electrónico y contraseña, devolviendo un token de sesión en caso de éxito.

asignarRol() – Asigna o modifica el rol de un usuario, controlando sus permisos y accesos dentro del sistema.

recuperarPassword() – Permite a un usuario solicitar la recuperación de su contraseña, ingresando su correo electrónico para verificación.

Descripción de las operaciones:

Atributo	Descripción
Operación	**registrarUsuario()**
Actor(es)	Administrador
URL	/usuarios
Método HTTP	POST
Lógica de Negocio	La operación permite registrar un nuevo usuario en el sistema. Se deben realizar las siguientes validaciones: • Verificar que el correo electrónico (email) no esté ya registrado en el sistema. • Validar que la contraseña cumpla con criterios de seguridad (mínimo 8 caracteres, mayúsculas, minúsculas y números). • Asignar un rol válido dentro del sistema (Ejemplo: "Administrador", "Agricultor", "Supervisor"). • Si las validaciones son exitosas, almacenar el usuario en la base de datos y devolver un mensaje de éxito con la información registrada.

Entrada	{
	"nombre": "String",
	"telefono": "String",
	"estatus": "boolean",
	"email": "String",
	"password": "String",
	"rol": "String"
	}

Salida	{
	"estatus": "String",
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**editarUsuario()**
Actor(es)	Todos los usuarios (Administrador, Agricultor y Supervisor)
URL	/usuarios/{id}
Método HTTP	PUT
Lógica de Negocio	La operación permite modificar los datos de un usuario existente en el sistema. Se deben realizar las siguientes validaciones: • Verificar que el usuario con el id especificado exista en la base de datos. • Validar que el correo electrónico (email) no esté registrado en otro usuario (si se va a modificar). • Si se actualiza la contraseña, validar que cumpla con los requisitos de seguridad.

	• Si las validaciones son exitosas, actualizar la información del usuario en la base de datos y devolver un mensaje de éxito.

Entrada	{
	"nombre": "String",
	"telefono": "String",
	"estatus": "boolean",
	"email": "String",
	"password": "String",
	"rol": "String"
	}

Salida	{
	"estatus": "String",
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**borrarUsuario()**
Actor(es)	Administrador
URL	/usuarios/{id}
Método HTTP	DELETE
Lógica de Negocio	La operación permite eliminar un usuario del sistema. Se deben realizar las siguientes validaciones: • Verificar que el usuario con el id especificado exista en la base de datos. • Si la validación es exitosa, eliminar el usuario y devolver un mensaje de éxito.
Entrada	No se requiere un cuerpo en la petición. Solo se envía el {id} en la URL (/usuarios/eliminar/{id}).

Salida	{
	"estatus": "String",
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**consultarUsuario()**
Actor(es)	Todos los usuarios (Administrador, Agricultor y Supervisor)
URL	/usuarios/{id}
Método HTTP	GET
Lógica de Negocio	La operación permite obtener la información detallada de un usuario específico en el sistema. Se deben realizar las siguientes validaciones: • Verificar que el usuario con el id especificado exista en la base de datos.

Entrada	No se requiere un cuerpo en la petición. Solo se envía el {id} en la URL (/usuarios/{id}).

Salida	{
	"estatus": "String",
	"mensaje": "String",
	"usuario":{
	"id": "String",
	"nombre": "String",
	"telefono": "String",
	"estatus": "boolean",
	"email": "String",
	"rol": "String"
	}
	}

Atributo	**Descripción**
Operación	**consultarListaUsuarios()**
Actor(es)	Administrador
URL	/usuarios
Método HTTP	GET
Lógica de Negocio	La operación permite obtener un listado de todos los usuarios registrados en el sistema. Se deben realizar las siguientes validaciones: • Si no hay usuarios registrados, devolver una lista vacía en la respuesta.
Entrada	No se requieren datos obligatorios en el cuerpo de la petición.

Salida	{
	"estatus": "String",
	"mensaje": "String",
	"usuarios": [{
	"id": "String",
	"nombre": "String",
	"estatus": "boolean",
	"email": "String"
	}]
	}

Atributo	**Descripción**
Operación	**iniciarSesion()**
Actor(es)	Todos los usuarios (Administrador, Agricultor y Supervisor)
URL	/usuarios/login

Método HTTP	POST
Lógica de Negocio	La operación permite autenticar a un usuario en el sistema. Se deben realizar las siguientes validaciones: • Verificar que el correo electrónico (email) exista en la base de datos. • Comparar la contraseña ingresada con la almacenada en el sistema. • Verificar que el usuario esté activo (estatus: true). • Si las credenciales son correctas, el usuario podrá iniciar sesión en la aplicación.

Entrada	{
	"email": "String",
	"password": "String"
	}

Salida	{
	"estatus": "String",
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**asignarRol()**
Actor(es)	Administrador
URL	/usuarios/{id}/rol
Método HTTP	PUT
Lógica de Negocio	La operación permite asignar un nuevo rol a un usuario específico. Se deben realizar las siguientes validaciones: • Verificar que el usuario con el id especificado exista en la base de datos. • Verificar que el rol proporcionado sea uno válido (por ejemplo, "Agricultor", "Administrador", "Supervisor").

Entrada	{
	"rol": "String"
	}

Salida	{
	"estatus": "String",
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**recuperarPassword()**
Actor(es)	Todos los usuarios (Administrador, Agricultor y Supervisor)
URL	/usuarios/recuperar-password
Método HTTP	POST
Lógica de Negocio	La operación permite a un usuario recuperar su contraseña en caso de olvido. Se deben realizar las siguientes validaciones: • Verificar que el correo electrónico (email) proporcionado exista en el sistema. • Devolver la contraseña del usuario si el correo proporcionado existe en el sistema.

Entrada	{
	"email": "String"
	}

Salida	{
	"estatus": "String",
	"mensaje": "String"
	}

Alertas

Tipo de Servicio: Es un servicio de utilidad debido a que su función principal no es gestionar entidades o recursos directamente, sino proporcionar funcionalidades específicas que pueden ser utilizadas por otros servicios del sistema. Este servicio facilita la gestión de alertas, pero no se centra en mantener un ciclo de vida completo de una entidad, sino en ofrecer servicios para generar, consultar o actualizar alertas cuando se cumplan ciertas condiciones. No está directamente involucrado con el almacenamiento o modificación directa de información fundamental del sistema, sino que se utiliza de forma transversal para comunicar estados, eventos o situaciones que requieren atención.

Responsable: Juan Humberto Báñales Guzmán

Operaciones expuestas:

- **registrarAlerta()** – Crea una nueva alerta en el sistema, almacenando información relevante como el tipo de alerta, la fecha de activación, el estado, entre otros datos.
- **editarAlerta()** – Modifica los detalles de una alerta existente, permitiendo actualizar atributos como el tipo de alerta, la descripción o el estado.
- **borrarAlerta()** – Elimina una alerta del sistema, ya sea de forma lógica (cambiando su estado) o física (eliminándola completamente de la base de datos).
- **consultarAlerta()** – Recupera la información de una alerta específica, utilizando su identificador único para devolver detalles como el tipo, fecha de activación, estado, entre otros.
- **consultarListaAlerta()** – Devuelve una lista de todas las alertas registradas, permitiendo filtrado y paginación según sea necesario.

Descripción de las operaciones:

Atributo	Descripción
Operación	**registrarAlerta()**
Actor(es)	Administrador y agricultores
URL	/alertas/registrar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar una nueva alerta en el sistema. Se deben validar los siguientes aspectos: • Que todos los campos obligatorios estén presentes (tipo de alerta, descripción, fecha de activación y estado). • Que el tipo de alerta pertenezca a una lista predefinida de tipos. • Si todas las validaciones se cumplen, la alerta se almacena en la base de datos y se devuelve una respuesta con éxito

Entrada	{
	"tipoAlerta": "String",
	"descripcion": "String",
	"fechaGenerada": "date",
	"estadoAlerta": "String"
	}

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**editarAlerta()**
Actor(es)	Administrador y agricultores
URL	/alertas/editar/{id}
Método HTTP	PUT
Lógica de Negocio	Esta operación permite actualizar los detalles de una alerta ya existente. Las validaciones incluyen: • Que el identificador de la alerta exista en la base de datos. • Que los campos modificados sean válidos (fecha de activación, tipo de alerta, etc.). • La alerta actualizada se persiste en la base de datos y se devuelve una respuesta con el estado de la operación.
Entrada	{ "tipoAlerta": "String", "descripcion": "String", "fechagenerada": "date",

	"estadoAlerta": "String" }

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**borrarAlerta()**
Actor(es)	Administrador
URL	/alertas/eliminar/{id}
Método HTTP	DELETE
Lógica de Negocio	Permite eliminar una alerta del sistema. Esta operación realiza una validación previa para asegurar que la alerta existe.
Entrada	No se requieren datos en el cuerpo de la petición. El id del insumo se pasa como parte de la URL (/alertas/eliminar/{id}).

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**consultarAlerta ()**
Actor(es)	Administrador y agricultores
URL	/alertas/{id}
Método HTTP	GET
Lógica de Negocio	Recupera los detalles de una alerta específica. El sistema valida que el identificador de la alerta sea válido y devuelve los datos correspondientes (tipo, fecha de activación, estado, descripcion).
Entrada	No se requieren datos en el cuerpo de la petición. El id del insumo se pasa como parte de la URL (/alertas/{id}).

Salida	{
	"mensaje": "String",
	"idAlerta": "int",
	"tipoAlerta": "String",
	"descripcion": "String",
	"fechagenerada": "date",
	"estadoAlerta": "String"
	}

Atributo	Descripción
Operación	**consultarListaAlertas ()**
Actor(es)	Administrador y agricultores
URL	/alertas
Método HTTP	GET
Lógica de Negocio	Recupera una lista de todas las alertas registradas. Esta operación permite filtrado y paginación según los parámetros proporcionados (por ejemplo, filtrar por tipo o estado de alerta).
Entrada	No se requieren datos en el cuerpo de la solicitud. Los parámetros opcionales de filtrado o paginación se incluyen como parámetros de consulta en la URL si aplica.
Salida	Lista de las alertas registradas en la base de datos.

Riegos

Tipo de Servicio: Es un servicio de entidad que gestiona el ciclo de vida de los riegos dentro del sistema, permitiendo realizar operaciones CRUD (crear, leer, actualizar y eliminar) sobre ellos. Este servicio es responsable de exponer y mantener la información de los riegos de manera organizada y consistente, de forma que otros servicios puedan acceder a ella de manera eficiente.

Responsable:

Juan Humberto Bañales Guzmán

Operaciones expuestas:

- **registrarRiego**() – Crea un nuevo registro de riego en el sistema, almacenando detalles como el tipo de riego, fecha, cantidad de agua, entre otros datos relevantes.
- **editarRiego**() – Permite modificar los detalles de un riego existente, actualizando atributos como la cantidad de agua, fecha, tipo de riego, etc.
- **borrarRiego**() – Elimina un riego del sistema, ya sea de manera lógica (cambiando su estado) o física (eliminándolo completamente de la base de datos).
- **consultarRiego**() – Recupera la información de un riego específico, utilizando su identificador único para obtener detalles como la cantidad de agua, fecha de ejecución, y tipo de riego.
- **consultarListaRiegos**() – Devuelve un listado de todos los riegos registrados, permitiendo filtrado y paginación si es necesario.

Descripción de operaciones:

Atributo	Descripción
Operación	**registrarRiego()**
Actor(es)	Administrador y agricultores
URL	/riegos/registrar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar un nuevo riego en el sistema. Se deben validar los siguientes aspectos:

	• Que todos los campos obligatorios estén presentes (fecha de riego, cantidad de agua, método y duración del riego). • Que la fecha de riego no sea anterior a la fecha actual. • Si todas las validaciones son correctas, el riego se almacena en la base de datos y se devuelve una respuesta de éxito. • Verificar que el usuario existe en la base de datos.

Entrada	{
	"fechaRiego": "date",
	"cantidadAgua": "float",
	"metodoRiego": "string",
	"duracionRiego": "float",
	"idUsuario": "ref (Usuarios) "
	}

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**editarRiego()**
Actor(es)	Administrador y agricultores
URL	/riegos/editar/{id}
Método HTTP	PUT
Lógica de Negocio	Esta operación permite modificar los detalles de un riego registrado. Las validaciones incluyen: • Que el identificador de riego exista en la base de datos. • Que los campos modificados (cantidad de agua, fecha de riego, tipo de riego) sean válidos. • Si se cumple con las validaciones, el riego se actualiza y se devuelve una respuesta con éxito. • Verificar que el usuario existe en la base de datos.

Entrada	{
	"fechaRiego": "date",
	"cantidadAgua": "float",
	"metodoRiego": "string",
	"duracionRiego": "float",
	"idUsuario": "ref (Usuarios) "
	}

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**borrarRiego()**
Actor(es)	Administrador
URL	/riegos/eliminar/{id}
Método HTTP	DELETE
Lógica de Negocio	Permite eliminar un riego del sistema. Se realiza una validación previa para verificar que el riego exista.
Entrada	No se requieren datos en el cuerpo de la petición. El id del insumo se pasa como parte de la URL (/riegos/eliminar/{id}).

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**consultarRiego()**
Actor(es)	Administrador y agricultores
URL	/riegos/{id}
Método HTTP	GET
Lógica de Negocio	Recupera los detalles de un riego específico usando su identificador. El sistema valida que el identificador sea válido y devuelve los datos correspondientes como tipo de riego, cantidad de agua, y fecha de ejecución.
Entrada	No se requieren datos en el cuerpo de la petición. El id del insumo se pasa como parte de la URL (/riegos/{id}).

Salida	{
	"mensaje": "String",
	"idRiego": "int",
	"fechaRiego": "date",
	"cantidadAgua": "float",
	"metodoRiego": "string",
	"duracionRiego": "float",
	"idUsuario": "ref (Usuarios) "
	}

Atributo	Descripción
Operación	**consultarListaRiegos()**
Actor(es)	Administrador y agricultores
URL	/riegos
Método HTTP	GET
Lógica de Negocio	Permite consultar un listado de todos los riegos registrados en el sistema. Se pueden aplicar filtros y paginación para devolver una lista ordenada de riegos, por ejemplo, filtrando por tipo de riego o fecha.
Entrada	No se requieren datos en el cuerpo de la solicitud. Los parámetros opcionales de filtrado o paginación se incluyen como parámetros de consulta en la URL si aplica.
Salida	Lista de los riegos registrados en la base de datos.

Historial_Suelo

Tipo de Servicio: Este es un servicio de entidad que gestiona el ciclo de vida de los registros del historial de suelos dentro del sistema. Permite realizar operaciones CRUD (crear, leer, actualizar y eliminar) sobre los registros de suelos, y se encarga de mantener la información relacionada con las condiciones del suelo de forma estructurada y accesible.

Responsable: Juan Humberto Bañales Guzmán

Operaciones expuestas:

- **registrarHistorial**() – Crea un nuevo registro de historial de suelo en el sistema, almacenando información como el tipo de suelo, las condiciones registradas (humedad, pH, etc.), la fecha de la medición, entre otros datos.
- **editarHistorial**() – Permite modificar los datos de un historial de suelo existente, actualizando atributos como tipo de suelo, condiciones registradas, fecha de la medición, etc.
- **borrarHistorial**() – Elimina un registro de historial de suelo del sistema, ya sea de forma lógica (cambiando su estado) o física (eliminándolo completamente de la base de datos).
- **consultarHistorial**() – Recupera la información de un historial de suelo específico, utilizando su identificador único para obtener detalles como tipo de suelo, fecha de medición, y las condiciones asociadas (humedad, pH, etc.).

Descripción de operaciones:

Atributo	Descripción
Operación	**registrarHistorial()**
Actor(es)	Administrador y técnicos de suelos
URL	/historial-suelo/registrar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar un nuevo historial de suelo en el sistema. Se deben validar los siguientes aspectos:

	• Que todos los campos obligatorios estén presentes (fecha de medición, pH, nutrientes y algunas observaciones). • Que la fecha de medición sea una fecha válida. • Si todas las validaciones se cumplen, el historial se almacena en la base de datos y se devuelve una respuesta de éxito. • Verificar que el cultivo existe en la base de datos. • Verificar que el usuario existe en la base de datos.

Entrada	{
	"fechaMedicion": "date",
	"pH": "float",
	"nutrientes": "string",
	"observaciones": "String",
	"idCultivo": "ref (Cultivos) ",
	"idUsuario": "ref (Usuarios) "
	}

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**editarHistorial()**
Actor(es)	Administrador y técnicos de suelos
URL	/historial-suelo/editar/{id}
Método HTTP	PUT
Lógica de Negocio	Permite modificar los datos de un historial de suelo existente. Las validaciones incluyen: • Que el identificador del historial exista en la base de datos. • Que la fecha de medición sea correcta. • Si se cumplen todas las validaciones, los datos se actualizan y se devuelve una respuesta con éxito. • Verificar que el cultivo existe en la base de datos. • Verificar que el usuario existe en la base de datos.
Entrada	{ "fechaMedicion": "date", "pH": "float", "nutrientes": "string", "observaciones": "String", "idCultivo": "ref (Cultivos) ", "idUsuario": "ref (Usuarios) "

	}

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**borrarHistorial()**
Actor(es)	Administrador
URL	/historial-suelos/eliminar/{id}
Método HTTP	DELETE
Lógica de Negocio	Permite eliminar un historial de suelo del sistema. Esta operación realiza una validación previa para asegurarse de que el historial exista en la base de datos.
Entrada	No se requieren datos en el cuerpo de la petición. El id del insumo se pasa como parte de la URL (/historial-suelos/eliminar/{id}).

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**consultarHistorial()**
Actor(es)	Administrador y técnicos de suelos
URL	/historial-suelo/{id}
Método HTTP	GET
Lógica de Negocio	Recupera los detalles de un historial de suelo específico, utilizando su identificador único. El sistema valida que el identificador sea válido y devuelve los datos correspondientes como tipo de suelo, humedad, pH, y fecha de medición.
Entrada	No se requieren datos en el cuerpo de la petición. El id del insumo se pasa como parte de la URL (/historial-suelo/{id}).

Salida	{
	"mensaje": "String",
	"idHistorial": "int",
	"fechaMedicion": "date",
	"pH": "float",
	"nutrientes": "string",
	"observaciones": "String",
	"idCultivo": "ref (Cultivos) ",
	"idUsuario": "ref (Usuarios) "
	}

Cultivos

Tipo de Servicio:

Servicio de Entidad. Este servicio gestiona información estructurada relacionada con cultivos, ubicaciones y seguimientos, permitiendo la creación, modificación, eliminación y consulta de registros en el sistema. Su principal función es almacenar y administrar datos de manera consistente, facilitando la trazabilidad y el control del ciclo productivo agrícola. No realiza tareas aisladas ni se centra en funciones auxiliares, sino que se enfoca en la persistencia y gestión de entidades clave dentro del dominio agrícola.

Responsable: Ulises Alvarado Godínez

Operaciones expuestas:

- **registrarCultivo()**: Permite crear un nuevo cultivo, validando la existencia y coherencia de campos como fechas, área, tipo de suelo y usuario responsable.
- **editarCultivo()**: Permite actualizar los datos de un cultivo existente, asegurando que los cambios cumplan con las reglas de negocio y validaciones correspondientes.
- **eliminarCultivo()**: Permite eliminar un cultivo, verificando que el registro exista antes de proceder a su eliminación.
- **consultarCultivo():** Permite recuperar la información detallada de un cultivo específico a partir de su identificador único.
- **consultarListaCultivos():** Permite obtener un listado de todos los cultivos registrados.
- **registrarSeguimiento()**: Permite registrar un nuevo seguimiento de cultivo, validando la fecha de revisión, el estado del cultivo, observaciones, recomendaciones y verificando la existencia de las referencias a cultivo y usuario.
- **editarSeguimiento()**: Permite actualizar los datos de un seguimiento existente, asegurando que las modificaciones cumplan con las validaciones de formato, coherencia de fechas, estado permitido y relaciones válidas.
- **eliminarSeguimiento()**: Permite eliminar un seguimiento, luego de confirmar la existencia del registro y evaluar posibles dependencias que impidan su eliminación.

- **consultarSeguimiento()**: Permite recuperar la información detallada de un seguimiento específico, mostrando todos sus atributos y las referencias asociadas al cultivo y usuario.
- **consultarListaSeguimiento():** Permite recuperar la lista de los seguimientos registrados a un cultivo en particular.
- **registrarUbicacion():** Permite registrar una nueva ubicación para un cultivo, validando que se proporcionen correctamente los datos como el nombre, coordenadas, superficie, tipo de suelo y disponibilidad de agua.
- **editarUbicacion()**: Permite actualizar la información de una ubicación existente, asegurando que los cambios cumplan con las validaciones de tipo y coherencia de los datos.
- **eliminarUbicacion()**: Permite eliminar una ubicación del sistema, previa verificación de la existencia del registro.
- **consultarUbicacion()**: Permite recuperar la información completa de una ubicación específica, devolviendo todos sus atributos y detalles asociados.

Descripción de las Operaciones:

Atributo	Descripción
Operación	**registrarCultivo()**
Actor(es)	Administrador y agricultor
URL	/cultivos/agregar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar un cultivo al sistema. Para lo cual se debe: • Validar que todos los campos obligatorios se encuentren presentes: nomCultivo, fechaSiembra, fechaCosechaEst, areaCultivo, tipoSuelo, estadoActual y idUsuario. • Verificar que fechaSiembra, fechaCosechaEst y (si se proporciona) fechaCosechaReal sean fechas válidas y

	que tengan coherencia (por ejemplo, la fecha estimada de cosecha no puede ser anterior a la fecha de siembra). • Comprobar que areaCultivo sea un valor numérico mayor a cero. • Hay que asegurar que el campo estadoActual contenga un valor acorde a los estados predefinidos. • Validar que idUsuario corresponda a un usuario existente en el sistema. Si todas las validaciones se cumplen, se procede a registrar el cultivo en la base de datos y se retorna una respuesta exitosa. De lo contrario se devuelve un mensaje de error.

Entrada	{
	"nomCultivo": String
	"fechaSiembra": Date
	"fechaCosechaEst": Date
	"fechaCosechaReal": Date
	"areaCultivo": number
	"tipoSuelo": String
	"estadoActual": String
	"idUsuario": ObjectId
	}

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**editarCultivo()**
Actor(es)	Administrador y agricultor
URL	/cultivos/editar/{idCultivo}
Método HTTP	PUT
Lógica de Negocio	La operación permite editar un cultivo del sistema. Para lo cual se debe:

	• Confirmar que el identificador del cultivo proporcionado en la URL exista en la base de datos. • Revisar cada campo enviado en el cuerpo de la petición y asegurar que cumpla con los tipos de datos esperados. • Validar la coherencia de las fechas: fechaSiembra debe ser anterior o igual a fechaCosechaEst y, en caso de existir, fechaCosechaReal debe tener sentido en el contexto del ciclo del cultivo. • Confirmar que areaCultivo se mantenga como un número positivo. • Asegurar que estadoActual se ajusten a los valores permitidos predefinidos. • Validar la existencia y coherencia de idUsuario. Si todas las validaciones son correctas, se procede a actualizar el registro del cultivo en la base de datos.

Entrada	{
	"nomCultivo": String
	"fechaSiembra": Date
	"fechaCosechaEst": Date
	"fechaCosechaReal": Date
	"areaCultivo": number
	"tipoSuelo": String
	"estadoActual": String
	"idUsuario": ObjectId
	}

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**eliminarCultivo()**
Actor(es)	Administrador

URL	/cultivos/eliminar/{idCultivo}
Método HTTP	DELETE
Lógica de Negocio	La operación permite eliminar un cultivo del sistema. Se deben realizar las siguientes validaciones: • Verificar que el cultivo con el id proporcionado exista en la base de datos. • Si el cultivo existe, proceder con su eliminación. • Si la eliminación es exitosa, se devuelve un mensaje de confirmación. • Si el cultivo no existe, se devuelve un mensaje de error.
Entrada	No se requiere cuerpo; el identificador se suministra en la URL.

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**consultarCultivo()**
Actor(es)	Administrador y agricultor
URL	/cultivos/consultar/{idCultivo}
Método HTTP	GET
Lógica de Negocio	La operación permite consultar la información de un cultivo del sistema. Para lo cual se debe: • Verificar que el identificador (id) incluido en la URL sea válido y que corresponda a un registro existente en la base de datos. • Realizar la consulta en la base de datos para recuperar todos los atributos del cultivo.
Entrada	No se requiere cuerpo; el identificador se suministra en la URL.

Salida	{
	"_id": ObjectId
	"nomCultivo": String
	"fechaSiembra": Date
	"fechaCosechaEst": Date
	"fechaCosechaReal": Date
	"areaCultivo": number
	"tipoSuelo": String
	"estadoActual": String
	"idUsuario": ObjectId
	"mensaje": String
	}

Atributo	**Descripción**
Operación	**consultarListaCultivos()**
Actor(es)	Administrador y agricultor
URL	/cultivos
Método HTTP	GET
Lógica de Negocio	La operación permite consultar la información de un cultivo del sistema. Para lo cual se debe: • Verificar que el identificador (id) incluido en la URL sea válido y que corresponda a un registro existente en la base de datos. • Realizar la consulta en la base de datos para recuperar todos los atributos del cultivo.
Entrada	No se requieren datos en el cuerpo de la solicitud. Los parámetros opcionales de filtrado o paginación se incluyen como parámetros de consulta en la URL si aplica.
Salida	[{ "_id": ObjectId "nomCultivo": String "fechaSiembra": Date "fechaCosechaEst": Date "fechaCosechaReal": Date

	"areaCultivo": number "tipoSuelo": String "estadoActual": String "idUsuario": ObjectId }]

Atributo	Descripción
Operación	**registrarSeguimiento()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/seguimientos/agregar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar un seguimiento a un cultivo del sistema. Para lo cual se debe: • Verificar que se envíen los campos: fechaRevision, estadoCultivo. • Confirmar que fechaRevision sea una fecha válida. • Asegurar que estadoCultivo contenga un valor dentro de los estados permitidos (por ejemplo, "Sano", "Enfermo", "Con Necesidad de Riego", etc.). • Revisar que observaciones y recomendaciones sean arrays de tipo String. En caso de no enviarse, se podrán inicializar como arrays vacíos. • Confirmar que el idCultivo haga referencia a un cultivo existente y que el idUsuario corresponda a un usuario autorizado. • Si el registro se hace de manera correcta se devuelve un mensaje de registro correcto, de lo contrario, se devuelve un mensaje de error.
Entrada	{ "fechaRevision": Date "estadoCultivo": String "observaciones": [String] "recomendaciones": [String]

	}

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**editarSeguimiento()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/seguimientos/editar/{idSeguimiento}
Método HTTP	PUT
Lógica de Negocio	La operación permite editar un seguimiento a un cultivo del sistema. Para lo cual se debe: • Comprobar que el seguimiento identificado por {id} exista en la base de datos. • Revisar que los campos recibidos en el cuerpo de la petición sean del tipo adecuado: fechaRevision: Debe ser una fecha válida. estadoCultivo: Debe pertenecer a los valores permitidos. observaciones y recomendaciones: Deben ser arrays de Strings. idCultivo e idUsuario: Verificar la existencia y validez de las referencias. • Actualizar los campos del seguimiento en la base de datos. • Se enviará un mensaje de éxito o error en la operación de actualización.

Entrada	{
	"fechaRevision": Date
	"estadoCultivo": String
	"observaciones": [String]
	"recomendaciones": [String]
	}

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**eliminarSeguimiento()**
Actor(es)	Administrador
URL	/cultivos/{idCultivo}/seguimientos/eliminar/{idSeguimiento}
Método HTTP	DELETE
Lógica de Negocio	La operación permite eliminar un seguimiento de un cultivo del sistema. Para lo cual se debe: • Confirmar que el seguimiento especificado por {id} exista en la base de datos. • Ejecutar la eliminación y, si aplica, efectuar la operación. Se retornará un mensaje de confirmación de la operación o por el contrario de un error en esta.
Entrada	No se requiere cuerpo en la solicitud; el identificador se suministra en la URL.

Salida	{
	"mensaje": "String"
	}

Atributo	Descripción
Operación	**consultarSeguimiento()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/seguimientos/consultar/{idSeguimiento}

Método HTTP	GET
Lógica de Negocio	La operación permite consultar un seguimiento de un cultivo del sistema. Para lo cual se debe: • Comprobar que el {id} enviado en la URL sea válido y que el registro de seguimiento exista en la base de datos. • Consultar y obtener todos los atributos del seguimiento, incluyendo fechaRevision, estadoCultivo, observaciones, recomendaciones, y las referencias idCultivo e idUsuario. • Retornar la información detallada del seguimiento en un objeto JSON.
Entrada	No se requiere cuerpo en la solicitud; el identificador se incluye en la URL.

Salida	{
	"_id": ObjectId
	"fechaRevision": Date
	"estadoCultivo": String
	"observaciones": [String]
	"recomendaciones": [String]
	"idCultivo": ObjectId
	"idUsuario": ObjectId
	"mensaje": String
	}

Atributo	**Descripción**
Operación	**consultarListaSeguimiento()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/seguimientos
Método HTTP	GET
Lógica de Negocio	La operación permite consultar la información de los seguimientos de un cultivo en el sistema. Para lo cual se debe:

	• Verificar que el identificador (id) incluido en la URL sea válido y que corresponda a un registro existente en la base de datos. • Realizar la consulta en la base de datos para recuperar todos los atributos del cultivo.
Entrada	No se requieren datos en el cuerpo de la solicitud. Los parámetros opcionales de filtrado o paginación se incluyen como parámetros de consulta en la URL si aplica.

Salida	[
	{
	"_id": ObjectId
	"fechaRevision": Date
	"estadoCultivo": String
	"observaciones": [String]
	"recomendaciones": [String]
	"idCultivo": ObjectId
	"idUsuario": ObjectId
	}
	]

Atributo	**Descripción**
Operación	**registrarUbicacion()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/ubicaciones/agregar
Método HTTP	POST
Lógica de Negocio	La operación permite registrar una ubicación a un cultivo del sistema. Para lo cual se debe: • Verificar que el cultivo al que se desea agregar la ubicación existe. • Verificar que se proporcione el campo nombreUbicacion y que no esté vacío. • Confirmar que las coordenadas contengan valores numéricos válidos para latitud y longitud.

	• Validar que superficie sea un número positivo mayor a cero. • Una vez que se validan todos los campos, se procede a almacenar la información en la base de datos. Si todas las validaciones se cumplen, se procede a registrar la ubicación al cultivo en la base de datos y se retorna una respuesta exitosa. De lo contrario se devuelve un mensaje de error.

Entrada	{
	"nombreUbicacion": String
	"coordenadas": {
	"latitud": number,
	"longitud": number }
	"superficie": number
	"tipoSuelo": String
	"accesoAgua": boolean
	}

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**editarUbicacion()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/ubicaciones/editar/{idUbicacion}
Método HTTP	PUT
Lógica de Negocio	La operación permite editar la ubicación de un cultivo del sistema. Para lo cual se debe: • Revisar y validar cada uno de los campos enviados en el cuerpo de la solicitud, asegurándose de que cumplan con los tipos de datos requeridos: nombreUbicacion debe ser un string no vacío.

	Las coordenadas deben contener valores numéricos válidos para latitud y longitud. superficie debe ser un número mayor a cero. • Actualizar el registro en la base de datos. Si todas las validaciones son correctas, se procede a actualizar el registro de la ubicación del cultivo en la base de datos.

Entrada	{
	"nombreUbicacion": String
	"coordenadas": { "latitud": number, "longitud": number }
	"superficie": number
	"tipoSuelo": String
	"accesoAgua": boolean
	}

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**eliminarUbicacion()**
Actor(es)	Administrador
URL	/cultivos/{idCultivo}/ubicaciones/eliminar/{idUbicacion}
Método HTTP	DELETE
Lógica de Negocio	La operación permite eliminar la ubicación de un cultivo del sistema. Para lo cual se debe: • Confirmar que la ubicación especificada por {id} exista en la base de datos. • Ejecutar la eliminación y, si aplica, efectuar la operación. Se retornará un mensaje de confirmación de la operación o por el contrario de un error en esta.

Entrada	No se requiere cuerpo en la solicitud; el identificador se suministra en la URL.

Salida	{
	"mensaje": "String"
	}

Atributo	**Descripción**
Operación	**consultarUbicacion()**
Actor(es)	Administrador y agricultor
URL	/cultivos/{idCultivo}/ubicaciones/consultar/{idUbicacion}
Método HTTP	GET
Lógica de Negocio	La operación permite consultar la ubicación de un cultivo del sistema. Para lo cual se debe: • Comprobar que el identificador {id} sea válido y que exista un registro asociado en la base de datos. • Recuperar la información completa de la ubicación, incluyendo nombreUbicacion, coordenadas, superficie, tipoSuelo y accesoAgua. • Devolver la información en un objeto JSON junto con un mensaje de éxito si así se consigue.
Entrada	No se requiere cuerpo en la solicitud; el identificador se incluye en la URL.

Salida	{
	"nombreUbicacion": String
	"coordenadas": { "latitud": number, "longitud": number }
	"superficie": number
	"tipoSuelo": String
	"accesoAgua": boolean
	"mensaje": String
	}