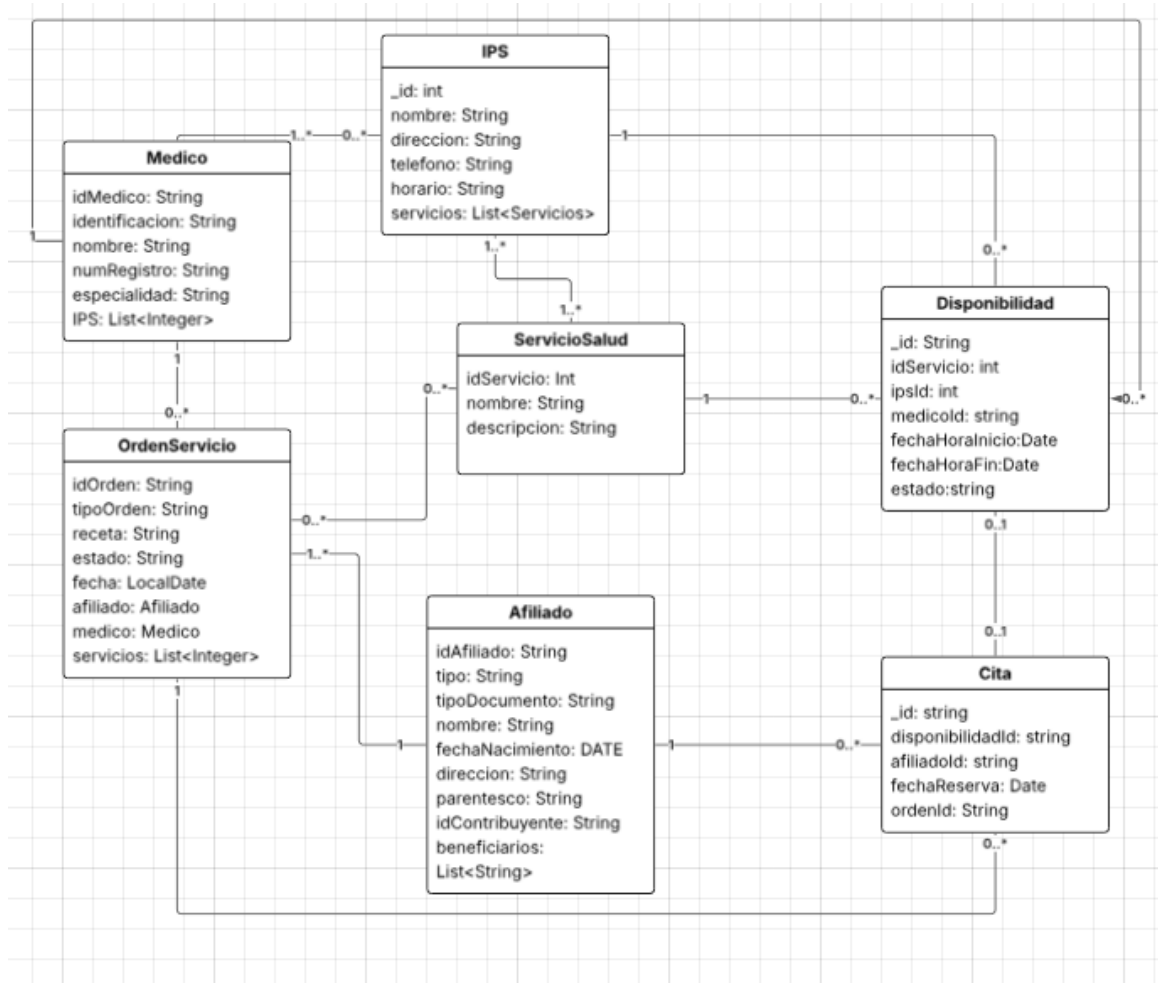


Entrega4 – Documentación

Grupo 4

1. Elementos esenciales
2. Modelo UML actualizado

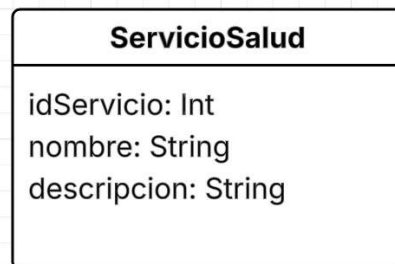


3. Diseño de la BD

a. Análisis de carga de trabajo

a. Entidades y atributos

- ServicioSalud



- OrdenServicio

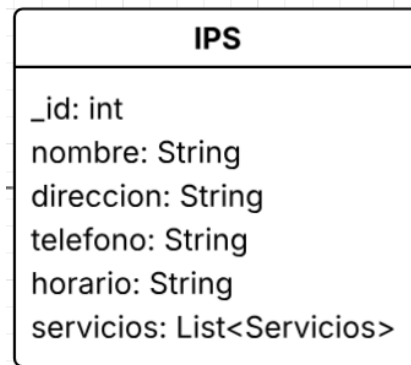
OrdenServicio
idOrden: Int tipoOrden: String receta: String estado: String fecha: DATE afiliado: Afiliado medico: Medico servicios: List<Integer>

- Medico

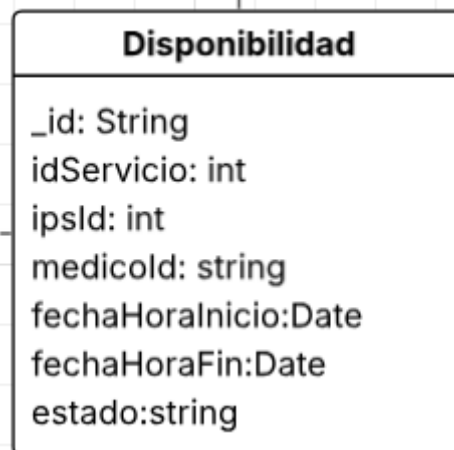
Medico
idMedico: String identificacion: String nombre: String numRegistro: String especialidad: String IPS: List<Integer>

- Afiliado

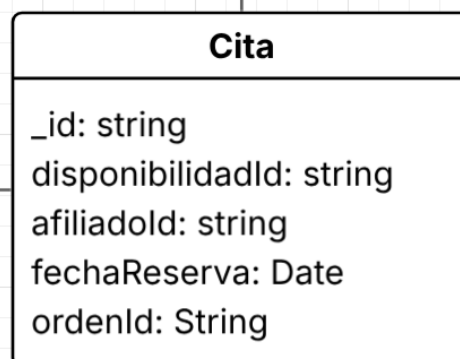
Afiliado
idAfiliado: String tipo: String tipoDocumento: String nombre: String fechaNacimiento: DATE direccion: String parentesco: String idContribuyente: String beneficiarios: List<String>



-Disponibilidad



-Cita



b. Cantidad de registros para cada entidad

Se estima que en promedio cada EPS cuente con 7 servicios de salud, entre 900.000 y 10.000.000 afiliados, 1000 a 9000 médicos, hasta 10.000 ips, 90.000 órdenes de servicio y citas al mes, lo cual quiere decir que un tiempo de un año se podría llegar a 1.000.000 citas y órdenes.

c. Operaciones de lectura y escritura (Anexo A)

- OrdenServicio

Entities	Operations	Information Needed	Type
OrdenServicio, Medico, Afiliado	Registrar nueva orden de servicio	ID afiliado + ID médico + lista servicios + fecha	Write
OrdenServicio, Medico, Afiliado	Consultar orden de servicio por ID	ID orden + estado + afiliado + servicios	Read
OrdenServicio, Afiliado	Consultar órdenes asociadas a un afiliado	ID afiliado + lista de órdenes	Read

OrdenServicio	Validar existencia de orden activa para agendamiento	ID orden + estado	Read
OrdenServicio	Actualizar estado de la orden (completada o cancelada)	ID orden + nuevo estado	Write
OrdenServicio, ServicioSalud	Obtener estadísticas para RFC2 (servicios más solicitados)	Servicios en órdenes + fechas	Read

- ServicioSalud

Entities	Operations	Information Needed	Type
ServicioSalud	Registrar un nuevo servicio de salud	_id + fecha + descripcion	Write
ServicioSalud	Consultar todos los servicios disponibles	Todos los documentos de servicios_collection	Read
ServicioSalud, IPS	Consultar servicios ofrecidos por una IPS específica	ID IPS + lista de servicios asignados	Read
ServicioSalud, IPS, Médico	Consultar agenda de un servicio en las siguientes 4 semanas (RFC1)	descripcion servicio + horarios disponibles + IPS + médico	Read
ServicioSalud, OrdenServicio	Obtener los 20 servicios más solicitados (RFC2)	Conteo de referencias a servicios en OrdenServicio	Read
ServicioSalud, IPS	Asignar un servicio a una IPS	ID servicio (_id) + ID IPS	Write

- Medico

Entities	Operations	Information Needed	Type
Medico	Registrar un nuevo médico	_id + identificación + nombre + numRegistro + especialidad	Write
Medico, IPS	Consultar médicos asignados a una IPS	ID IPS + lista de médicos	Read
Medico, ServicioSalud	Consultar servicios que presta un médico	ID médico + especialidad + servicios relacionados	Read
Médico, OrdenServicio, Afiliado	Registrar una orden de servicio por parte de un médico (RF6)	ID médico + ID afiliado + servicios prescritos	Write
Medico, ServicioSalud, IPS	Consultar disponibilidad del médico para un servicio (RFC1)	ID médico + disponibilidad + servicio + IPS	Read
Medico, OrdenServicio	Obtener estadísticas de uso por servicio (RFC2)	Referencias a médicos en órdenes de servicio	Read

- Afiliado

Entities	Operations	Information Needed	Type
----------	------------	--------------------	------

Afiliado	Registrar un nuevo afiliado (contribuyente o beneficiario) (RF5)	Tipo + número de documento + nombre + fecha nacimiento + parentesco	Write
Afiliado, OrdenServicio	Consultar las órdenes asociadas a un afiliado	ID afiliado + lista de órdenes	Read
Afiliado, OrdenServicio, ServicioSalud	Agendar un servicio de salud (requiere validar orden y disponibilidad) (RF7)	ID afiliado + ID orden válida + disponibilidad en agenda	Write
Afiliado	Consultar datos personales de un afiliado	ID afiliado + nombre + tipo + contacto	Read
Afiliado, OrdenServicio	Obtener histórico de órdenes de un afiliado	ID afiliado + fechas + estados de órdenes	Read
Afiliado, ServicioSalud, OrdenServicio	Consultar los servicios más usados por afiliados (RFC2)	Relación entre afiliados y servicios usados en órdenes	Read

- IPS, Disponibilidad, Cita

Entities	Operations	Information Needed	Type
IPS	Registrar una nueva IPS	Id+ nombre + direccion + telefono + horario + servicios	Write
IPS, ServicioSalud	Consultar los Servicios de salud asociados a una IPS	Id IPS	Read
IPS, Medico	Registrar las IPS donde está asociado un médico.	IdMedico + identificacion + nombre + numRegistro + especialidad + id IPSS	Write
IPS, disponibilidad, ServicioSalud, Medico,	Registrar la disponibilidad que tiene una IPS sobre cierto Servicio de salud	IdServicio + ipsId + medicoId + fecha de inicio y fin + ordenId	Write
Disponibilidad, Orden, Afiliado	Agendar cita	Disponibilidad + Orden + Identificacion Afiliado	Write
IPS, disponibilidad, ServicioSalud, Medico	Consultar la disponibilidad para cierto servicio salud	IdServicio	Read

d. Operaciones de lectura y escritura (Anexo B)

- OrdenServicio

Entities	Operation	Information Needed	Type	Rate
OrdenServicio	Registrar nueva orden de servicio	ID afiliado + ID médico + servicios + fecha	Write	3,000/day ≈ 2/min

OrdenServicio	Consultar orden por ID	ID orden + estado + afiliado + servicios	Read	3,000/day \approx 2/min
OrdenServicio	Consultar órdenes de un afiliado	ID afiliado + lista de órdenes	Read	3,000/day \approx 2/min
OrdenServicio	Validar existencia de orden activa para agendamiento	ID orden + estado	Read	2,000/day \approx 1.5/min
OrdenServicio	Actualizar estado (completada o cancelada)	ID orden + nuevo estado	Write	1,000/day \approx 0.7/min
OrdenServicio	Generar estadísticas de servicios más solicitados	Conteo por tipo de servicio en órdenes	Read	100/hour (estimado para RFC2)

- ServicioSalud

Entities	Operation	Information Needed	Type	Rate
ServicioSalud	Registrar nuevo servicio de salud	<u>_id</u> + fecha + descripcion	Write	1/month \approx 0.00002/min
ServicioSalud	Consultar todos los servicios disponibles	Lista de todos los servicios	Read	10,000/day \approx 7/min
ServicioSalud, IPS	Consultar servicios asignados a una IPS	ID IPS + lista de servicios asignados	Read	5,000/day \approx 3.5/min
ServicioSalud, IPS, Médico	Consultar agenda disponible de un servicio (RFC1)	ID servicio + horarios + IPS + médicos disponibles	Read	5,000/day \approx 3.5/min
ServicioSalud, IPS	Asignar un servicio a una IPS	ID servicio + ID IPS	Write	20/month \approx 0.0005/min
ServicioSalud, OrdenServicio	Obtener estadísticas de uso (RFC2)	Conteo de referencias a servicios en órdenes	Read	100/hour \approx 1.7/min

- Medico

Entities	Operation	Information Needed	Type	Rate
Medico	Registrar nuevo médico (RF4)	<u>_id</u> + identificación + nombre + especialidad + numRegistro	Write	100/month \approx 0.002/min
Méedico	Consultar información de un médico	ID médico + especialidad + datos personales	Read	5,000/day \approx 3.5/min
Medico, IPS	Consultar médicos asignados a una IPS	ID IPS + lista de médicos	Read	500/day (estimado)
Medico, ServicioSalud	Consultar servicios que presta un médico	ID médico + especialidades	Read	500/day (estimado)
Medico, OrdenServicio	Registrar orden emitida por un médico (RF6)	ID médico + servicios prescritos	Write	3,000/day \approx 2/min
Medico, ServicioSalud, IPS	Consultar agenda para un servicio de salud (RFC1)	ID médico + disponibilidad + servicio	Read	5,000/day \approx 3.5/min

Medico, OrdenServicio	Obtener estadísticas de servicios más solicitados (RFC2)	Conteo de médicos en órdenes de servicio	Read	100/hour \approx 1.7/min
--------------------------	--	--	------	----------------------------

- Afiliado

Entities	Operation	Information Needed	Type	Rate
Afiliado	Registrar nuevo afiliado (contribuyente o beneficiario) (RF5)	Tipo y número de documento + nombre + fecha + parentesco	Write	9,000/month \approx 0.2/min
Afiliado	Consultar datos personales de un afiliado	ID afiliado + nombre + tipo + contacto	Read	10,000/day \approx 7/min
Afiliado, OrdenServicio	Consultar órdenes asociadas a un afiliado	ID afiliado + lista de órdenes	Read	3,000/day \approx 2/min
Afiliado, OrdenServicio	Agendar un servicio (requiere ID afiliado y orden) (RF7)	ID afiliado + ID orden	Write	2,000/day \approx 1.5/min
Afiliado, OrdenServicio, ServicioSalud	Obtener servicios más usados (RFC2)	Servicios más usados por afiliados	Read	100/hour \approx 1.7/min

- IPS, Disponibilidad, Cita

Entities	Operations	Information Needed	Type	Rate
IPS	Registrar una nueva IPS	Id+ nombre + direccion + telefono + horario + servicios	Write	10/mes
IPS, ServicioSalud	Consultar información de la IPS como servicios de salud	Id IPS	Read	5000/día
IPS, Medico	Registrar las IPS donde está asociado un médico.	IdMedico + identificacion + nombre + numRegistro + especialidad + id IPSS	Write	100/mes
IPS, disponibilidad, ServicioSalud, Medico,	Registrar la disponibilidad que tiene una IPS sobre cierto Servicio de salud	IdServicio + ipsId + medicoId + fecha de inicio y fin + ordenId	Write	90000/mes
Disponibilidad , Orden, Afiliado	Agendar cita	Disponibilidad + Orden + Identificacion Afiliado	Write	90000/mes
IPS, disponibilidad,	Consultar la disponibilidad para cierto servicio salud	IdServicio	Read	90000/mes

ServicioSalud, Medico				
--------------------------	--	--	--	--

b. Descripción de las colecciones de datos y las relaciones entre ellas

a. Lista de entidades

- ServicioSalud

```
@Document(collection = "servicios_collection")
@ToString
public class ServicioSalud {

    @Id
    private String idServicio;
    private LocalDate fecha;
    private String descripcion;

    public ServicioSalud() {
        ;
    }
}
```

La entidad ServicioSalud almacena los diferentes tipos de servicios médicos que una IPS puede ofrecer: desde consultas generales hasta terapias o procedimientos especializados. Cada documento incluye un identificador, la fecha del servicio y una descripción que define el tipo de servicio. Aunque esta información no cambia con frecuencia —se estima que en promedio se registra o modifica un servicio al mes—, sí se consulta constantemente: alrededor de 10.000 veces al día

- Medico

```
@Document(collection = "medicos_collection")
@ToString
public class Medico {

    @Id
    private String idMedico;
    private String identificacion;
    private String nombre;
    private String numRegistro;
    private String especialidad;
    private List<Integer> ips;

    public Medico() {}
}
```

La entidad Médico representa a cada profesional de salud registrado en el sistema. Incluye atributos como idMedico, identificación, nombre, número de registro, especialidad y una lista de ips, que corresponde a los identificadores de las Instituciones Prestadoras de Salud donde trabaja.

En este diseño, la relación entre médicos e IPS es de muchos a muchos: un médico puede estar vinculado a múltiples IPS y cada IPS puede contratar múltiples médicos. Por eso, se implementa mediante **referencias**, es decir, almacenando únicamente los identificadores de las IPS asociadas dentro de cada documento de médico. Este enfoque es más eficiente que embebido, ya que evita duplicar la información de cada IPS en múltiples documentos de médico, facilita la actualización independiente y permite escalar mejor cuando aumenta el número de vínculos entre entidades.

- Afiliado

```
@Document(collection = "afiliados_collection")
public class Afiliado {

    @Id
    private String idAfiliado;
    private String tipo;
    private String tipoDocumento;
    private String nombre;
    private Date fechaNacimiento;
    private String direccion;

    // Solo si es beneficiario:
    private String parentesco;
    private String idContribuyente;

    // Solo si es contribuyente
    private List<String> beneficiarios;

    public Afiliado() {}
}
```

La entidad Afiliado representa a cualquier persona vinculada a la EPS, ya sea como contribuyente (quien cotiza directamente) o como beneficiario (familiar dependiente de un contribuyente). El documento incluye datos básicos como tipo y número de documento, nombre, fecha de nacimiento y dirección. Si el afiliado es beneficiario, se registra el tipo de parentesco y el idContribuyente del cual depende. Por el contrario, si es un contribuyente, puede incluir una lista de IDs que corresponden a sus beneficiarios.

La relación entre afiliados es de uno a muchos: un solo contribuyente puede tener varios beneficiarios. Esta relación se implementa usando referencias (IDs), lo cual tiene varias ventajas. Por un lado, permite mantener cada afiliado como un documento independiente —necesario para consultas, actualizaciones o creación de órdenes—. Por otro, evita que un cambio en los datos del contribuyente implique reescribir todos sus beneficiarios, o viceversa.

- OrdenServicio

```
@Document(collection = "ordenServicio_collection")
public class OrdenServicio {

    @Id
    private String idOrden;
    private String tipoOrden;
    private String receta;
    private String estado;
    @JsonFormat(pattern = "yyyy-MM-dd")
    private LocalDate fecha;
    private String idAfiliado; // referenciado como String
    private String idMedico;
    private Afiliado afiliado;
    private Medico medico;
    private List<Integer> servicios;
}
```

La entidad OrdenServicio representa las órdenes médicas generadas por un médico para un afiliado, como parte del proceso de atención en salud. Cada documento incluye el tipo de orden (por ejemplo, consulta o terapia), la receta asociada, el estado actual (vigente, completada, cancelada), la fecha, el identificador del afiliado, el médico responsable y la lista de servicios requeridos.

En cuanto a las relaciones, esta entidad funciona como punto de conexión entre varios actores del sistema. Un afiliado puede tener múltiples órdenes de servicio a lo largo del tiempo, pero cada orden pertenece a un solo

afiliado. Lo mismo ocurre con el médico: un médico puede emitir muchas órdenes, pero cada orden tiene un único médico asociado. Por eso, tanto el médico como el afiliado se han embebido en el documento: esto permite acceder rápidamente a sus datos al consultar una orden específica, sin necesidad de hacer joins o búsquedas adicionales.

Distinto es el caso de los servicios. Una orden puede incluir varios servicios de salud (como una consulta, una terapia y un examen), y al mismo tiempo un mismo servicio puede estar presente en muchas órdenes distintas. Como esta relación es claramente de muchos a muchos, los servicios se manejan mediante referencias: se guarda una lista de IDs que apuntan a documentos de la colección ServicioSalud. Así se evita la duplicación y se facilita el mantenimiento cuando cambia algo en la definición de un servicio.

- Disponibilidad

```
@Document(collection = "disponibilidad_collection")
public class Disponibilidad {
    @Id
    private String id;
    private int idServicio;
    private int ipsId;
    private String medicoId;
    private Date fechaHoraInicio;
    private Date fechaHoraFin;
    private String estado;
```

La entidad Disponibilidad representa los diferentes horarios en los cuales una IPS habilita un Servicio de Salud con cierto Medico. Cada documento incluye un identificador de la disponibilidad, el id referenciado para el servicio de Salud, la IPS y el médico, así mismo la fecha y hora tanto de inicio como de finalización, por último, se almacena el estado que puede ser LIBRE u OCUPADO.

En cuanto a las relaciones esta entidad conecta diferentes entidades, de los cuales servicios de salud, IPS y médicos pueden tener muchos horarios disponibles para formar el servicio. Al haber muchas entidades uno a mucho referenciadas se decidió referenciar.

- Cita

```
@Document(collection = "citas_collection")
public class Cita {
    @Id
    private String id;
    private String disponibilidadId;
    private String afiliadoId;
    private Date fechaReserva;
    private String ordenId;
```

La entidad Cita representa la idea de que el paciente reservó un servicio de salud ofertado en cierta fecha y que para realizar está reserva es porque tenía una orden dada por un médico, esto también nos ayuda para poder llevar un historial.

Esta entidad tiene diferentes asociaciones a modo de referencia, iniciando con disponibilidad, la cuál es una relación 1 a 1, y luego tenemos una orden y un afiliado que tienen relaciones muchos a uno con cita.

b. Las relaciones entre entidades y su cardinalidad (uno a uno, uno a muchos o muchos a muchos)

- ServicioSalud
 - Muchos a muchos
 - OrdenServicio
 - IPS
- Medico
 - Muchos a muchos
 - IPS
 - Uno a muchos
 - OrdenServicio
- OrdenServicio
 - Muchos a muchos
 - ServicioSalud
 - Uno a muchos
 - Afiliado
 - Medico
- Afiliado
 - Uno a muchos
 - OrdenServicio
- Disponibilidad
 - Uno a muchos
 - IPS
 - ServicioSalud
 - Medico
- Cita
 - Uno a muchos
 - Afiliados
 - OrdenServicio
 - Uno a uno
 - Disponibilidad

c. Esquema de asociación (Anexo C)

- ServicioSalud

ServicioSalud -> OrdenServicio (muchos a muchos)

Guideline	Respuesta	Justificación
Simplicity	No	Embebido generaría documentos redundantes o muy grandes.
Go Together	No	Un servicio puede estar en muchas órdenes, y no siempre se consulta con estas.

Query Atomicity	No	Orden y servicio no siempre se consultan juntos.
Update Complexity	No	Los servicios rara vez se actualizan; embebido complicaría mantener consistencia.
Archival	No	No se archivan juntos.
Cardinality	Yes	Un servicio puede estar en miles de órdenes.
Data Duplication	Yes	Embebido generaría mucha duplicación innecesaria.
Document Size	Yes	Incrustar servicios en órdenes puede inflar el tamaño.
Document Growth	Yes	Podría crecer sin control si se embeben varios servicios por orden.
Workload	Yes	Los servicios se crean poco, pero las órdenes se escriben constantemente.
Individuality	Yes	Los servicios existen independientemente.

Respuesta: Referenciado

ServicioSalud -> IPS (muchos a muchos)

Guideline	Respuesta	Justificación
Simplicity	No	No aporta simplicidad tener servicios embebidos en IPS o viceversa.
Go Together	No	Pueden consultarse por separado.
Query Atomicity	No	No siempre se necesitan juntos.
Update Complexity	No	Actualizar un servicio no debería implicar modificar múltiples IPS.
Archival	No	No se archivan en conjunto.
Cardinality	Yes	Cada servicio puede estar asignado a muchas IPS, y viceversa.
Data Duplication	Yes	Embebido generaría redundancia y dificultad para mantener consistencia.
Document Size	Yes	Tener todos los servicios embebidos en una IPS puede inflar el documento.
Document Growth	Yes	Las asociaciones entre IPS y servicios pueden crecer con el tiempo.
Workload	Yes	Las IPS se escriben poco, pero los servicios se consultan frecuentemente.
Individuality	Yes	Los servicios y las IPS deben mantenerse como entidades independientes.

Respuesta: Referenciado

- Medico

Medico -> IPS (muchos a muchos)

Guideline	Respuesta	Justificación
Simplicity	No	Embebido complicaría la estructura.
Go Together	No	No siempre se consultan en conjunto.

Query Atomicity	No	No hay una necesidad común de leerlos juntos.
Update Complexity	No	Las relaciones pueden cambiar de forma independiente.
Archival	No	No se archivan en conjunto.
Cardinality	Yes	Un médico puede trabajar en varias IPS y viceversa.
Data Duplication	Yes	Embebido generaría duplicación y riesgo de inconsistencia.
Document Size	Yes	Incluir todas las IPS en cada médico o viceversa puede inflar el documento.
Document Growth	Yes	La lista de asignaciones puede crecer indefinidamente.
Workload	Yes	Los médicos y las IPS se actualizan por separado y con poca frecuencia.
Individuality	Yes	Ambos existen por sí solos.

Respuesta: Referenciado

Medico -> OrdenServicio (uno a muchos)

Guideline	Respuesta	Justificación
Simplicity	Yes	Incluir los datos del médico dentro de la orden simplifica la lectura.
Go Together	Yes	Siempre que se consulta una orden se necesita saber quién la emitió.
Query Atomicity	Yes	Se consultan juntos casi siempre.
Update Complexity	No	Los datos del médico rara vez cambian.
Archival	Yes	La orden y sus datos asociados se archivan como un todo.
Cardinality	No	Cada médico puede tener muchas órdenes, pero no al revés.
Data Duplication	No	La duplicación es manejable dado que los médicos no cambian con frecuencia.
Document Size	No	El tamaño de los datos embebidos es pequeño.
Document Growth	No	No hay crecimiento descontrolado.
Workload	No	Las órdenes se escriben con alta frecuencia; el médico se escribe poco.
Individuality	No	El médico puede existir solo, pero en este caso es más útil tenerlo embebido.

Respuesta: Embebido

- OrdenServicio

OrdenServicio -> ServicioSalud (muchos a muchos)

Guideline	Respuesta	Justificación
Simplicity	No	Embebido complicaría el diseño por duplicación masiva.

Go Together	No	No siempre se consultan los servicios con la orden completa.
Query Atomicity	No	No se consultan o actualizan juntos todo el tiempo.
Update Complexity	No	Cambios en el servicio no deberían afectar todas las órdenes.
Archival	No	Se archivan por separado.
Cardinality	Yes	Un servicio puede estar en muchas órdenes y viceversa.
Data Duplication	Yes	Sería ineficiente duplicar toda la info del servicio en cada orden.
Document Size	Yes	Las órdenes pueden crecer demasiado si se embebe.
Document Growth	Yes	Podría ser impredecible por la cantidad variable de servicios.
Workload	Yes	Servicios se escriben poco, órdenes se escriben mucho.
Individuality	Yes	Los servicios existen de forma independiente.

Respuesta: Referenciado

OrdenServicio -> Afiliado (uno a muchos)

Guideline	Respuesta	Justificación
Simplicity	Yes	Embebido facilita las consultas y evita joins.
Go Together	Yes	Siempre se necesita saber quién es el afiliado al consultar una orden.
Query Atomicity	Yes	Se acceden juntos.
Update Complexity	No	La info del afiliado cambia poco.
Archival	Yes	Se archiva la orden completa.
Cardinality	No	Cada orden tiene un afiliado único.
Data Duplication	No	La duplicación es pequeña y manejable.
Document Size	No	No impacta mucho el tamaño de la orden.
Document Growth	No	El tamaño es constante.
Workload	No	Las órdenes se escriben constantemente, los afiliados rara vez.
Individuality	No	Se puede mantener embebido sin perder consistencia.

Respuesta: Embebido

OrdenServicio -> Medico (uno a muchos)

Guideline	Respuesta	Justificación
Simplicity	Yes	Tener los datos del médico embebidos simplifica la orden.
Go Together	Yes	Siempre se necesita saber quién la emitió.
Query Atomicity	Yes	Se consultan juntos.
Update Complexity	No	Los datos del médico no cambian con frecuencia.
Archival	Yes	Se archiva como parte de la orden.
Cardinality	No	Una orden tiene un solo médico.
Data Duplication	No	No es un problema porque el médico cambia poco.
Document Size	No	Embebido no afecta significativamente el tamaño.
Document Growth	No	El crecimiento está controlado.
Workload	No	La orden se escribe constantemente, el médico casi nunca.
Individuality	No	Se puede manejar embebido sin dificultad.

Respuesta: Embebido











- Afiliado

Afiliado -> OrdenServicio (uno a muchos)

Guideline	Respuesta	Justificación
Simplicity	No	Incluir todas las órdenes en el documento del afiliado lo haría complejo.
Go Together	No	Las órdenes se consultan por separado del afiliado muchas veces.
Query Atomicity	No	No siempre se acceden juntas.
Update Complexity	No	Las órdenes cambian con frecuencia, el afiliado casi nunca.
Archival	No	No se archivan juntas.
Cardinality	Yes	Un afiliado puede tener decenas o cientos de órdenes.
Data Duplication	Yes	Embebido causaría crecimiento y duplicación innecesaria.
Document Size	Yes	Aumentaría mucho si se embeben múltiples órdenes.
Document Growth	Yes	Cada nuevo servicio implicaría alterar el documento del afiliado.
Workload	Yes	Alta frecuencia de escritura sobre órdenes, no sobre afiliado.
Individuality	Yes	Las órdenes deben existir por sí solas.

Respuesta: Embebido

Disponibilidad-> IPS (uno a muchos)

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	 No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	 No
Query Atomicity	Does the application query the pieces of information together?	Yes	 No
Update Complexity	Are the pieces of information updated together?	Yes	 No
Archival	Should the pieces of information be archived at the same time?	Yes	 No
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	 Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	 Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	 Yes
Document Growth	Would the embedded piece grow without bound?	No	 Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	 Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	 Yes

Resultado **referenciado**

Disponibilidad-> Medicos (uno a muchos)

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are the pieces of information updated together?	Yes	No
Archival	Should the pieces of information be archived at the same time?	Yes	No
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes

Resultado **referenciado**

Disponibilidad-> Cita (uno a uno)

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are the pieces of information updated together?	Yes	No
Archival	Should the pieces of information be archived at the same time?	Yes	No
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes

Resultado **embebido**

Cita-> Afiliado (uno a muchos)

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are the pieces of information updated together?	Yes	No
Archival	Should the pieces of information be archived at the same time?	Yes	No
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes

Resultado **referenciado**

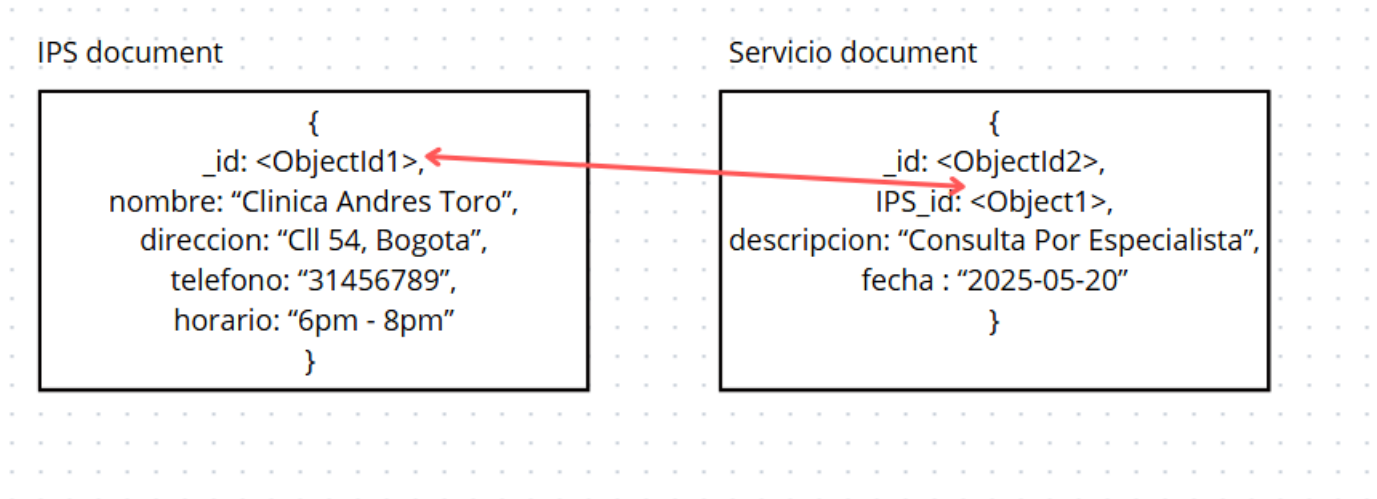
Cita-> Orden Servicio (uno a muchos)

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are the pieces of information updated together?	Yes	No
Archival	Should the pieces of information be archived at the same time?	Yes	No
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes

Resultado **referenciado**

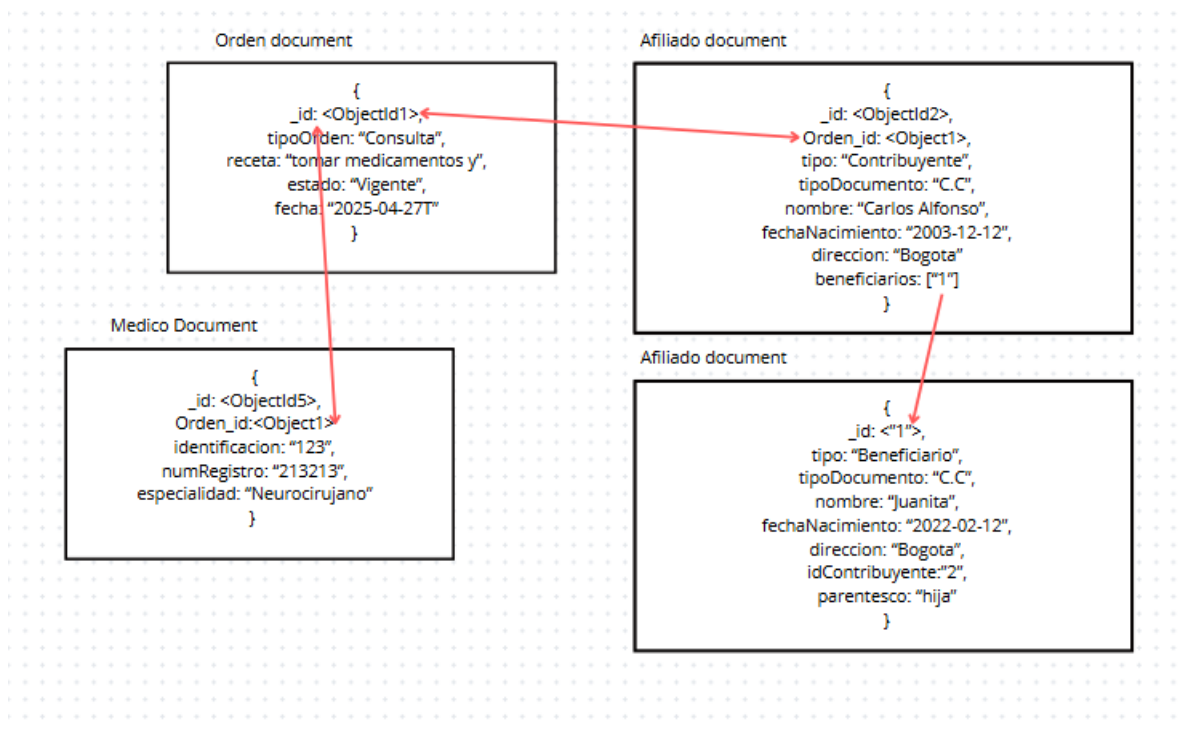
d. Descripción gráfica para cada relación (Anexo D)

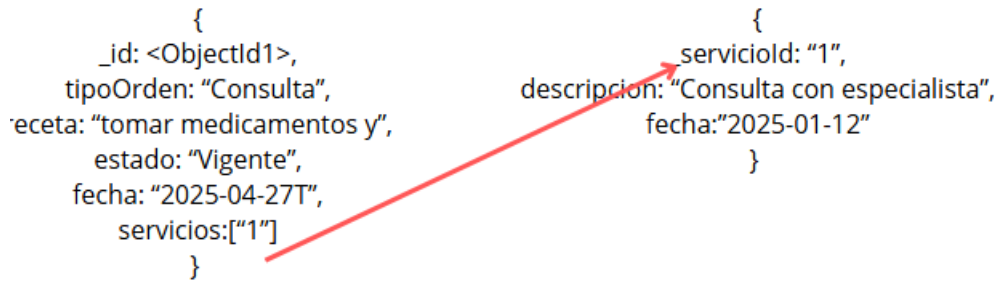
-Ips-Servicio



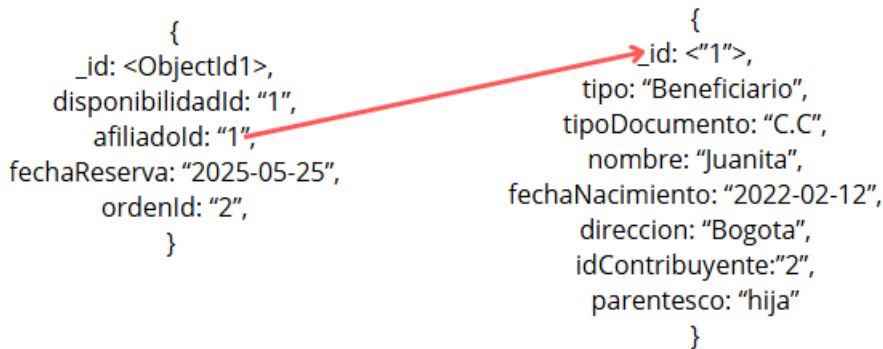
-OrdenServicio embebe -> Afiliado-Medico

Afiliado Referencia -> Afiliado (Esto sucede cuando es del tipo contribuyente, por ende, tiene asociado una lista de beneficiarios.)

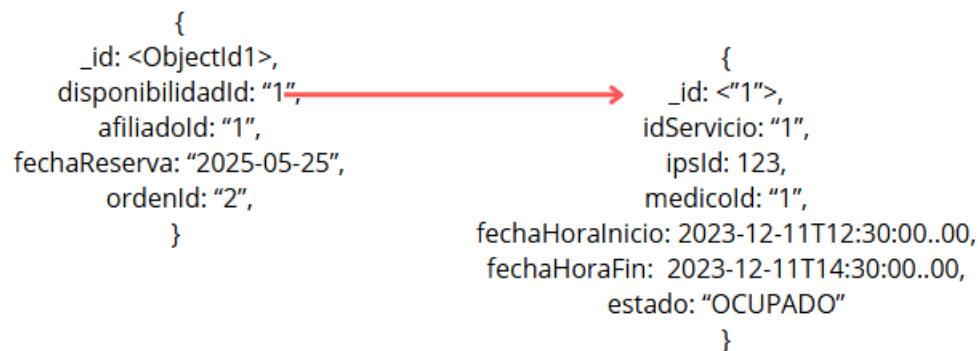




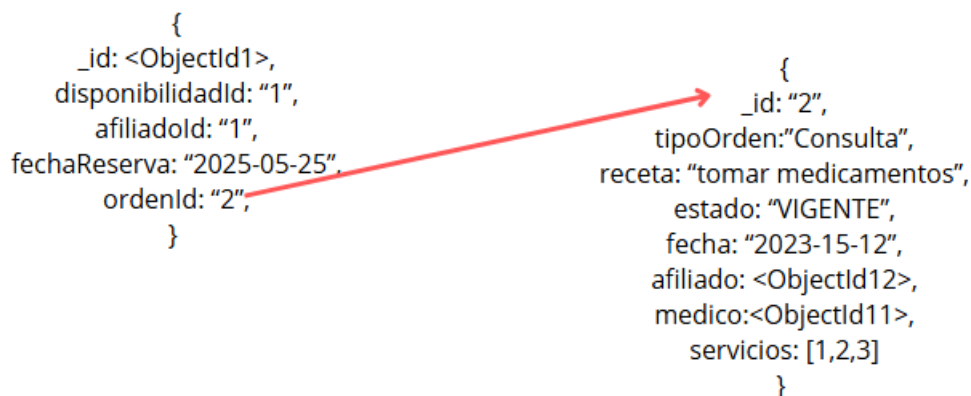
Cita referencia Afiliado



Cita Referencia Disponibilidad



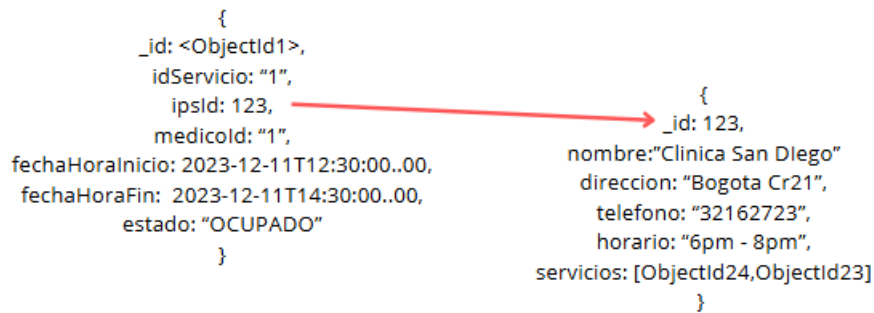
Cita Referencia una orden



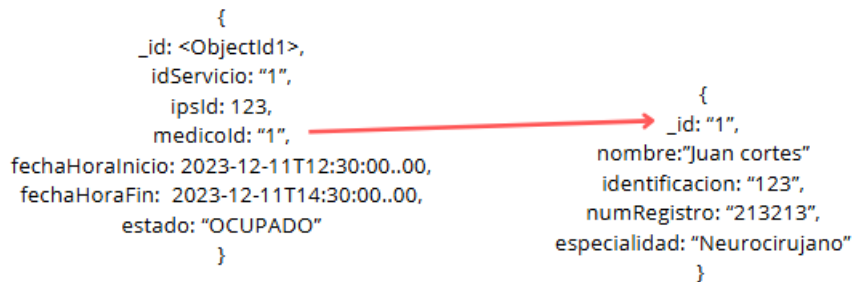
Disponibilidad referencia el servicio



Disponibilidad referencia la IPS



Disponibilidad referencia el medico



c. Los scripts para la creación de las colecciones se encuentran adjunto en la entrega

d. Esquemas de validación:

-IPS:

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": [ "_id", "direccion", "horario", "nombre", "servicios", "telefono" ],
    "properties": {
      "_id": { "bsonType": "int" },
      "direccion": { "bsonType": "string" },
      "horario": { "bsonType": "string" },
      "nombre": { "bsonType": "string" },
```

```

"servicios": {
  "bsonType": "array",
  "items": {
    "bsonType": "object",
    "required": [ "_id", "descripcion", "fecha" ],
    "properties": {
      "_id": { "bsonType": "int" },
      "descripcion": { "bsonType": "string" },
      "fecha": { "bsonType": "string" }
    }
  }
},
"telefono": { "bsonType": "string" }
}
}
}

```

- **Afiliados**

```

{
  "$jsonSchema": {
    "bsonType": "object",
    "required": [
      "_id",
      "beneficiarios",
      "direccion",
      "fechaNacimiento",
      "idContribuyente",
      "nombre",
      "parentesco",
      "tipo",
      "tipoDocumento"
    ],
    "properties": {
      "_id": { "bsonType": "string" },
      "beneficiarios": {
        "anyOf": [ { "bsonType": "array", "items": { "bsonType": "string" } }, { "bsonType": "null" } ]
      },
      "direccion": { "bsonType": "string" },
      "fechaNacimiento": { "bsonType": "date" },
      "idContribuyente": { "bsonType": [ "null", "string" ] },
      "nombre": { "bsonType": "string" },

```

```

    "parentesco": { "bsonType": ["null", "string"] },
    "tipo": { "bsonType": "string" },
    "tipoDocumento": { "bsonType": "string" }
  }
}

```

-Medico

```

{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["_id", "especialidad", "identificacion", "ips", "nombre", "numRegistro"],
    "properties": {
      "_id": { "bsonType": "string" },
      "especialidad": { "bsonType": "string" },
      "identificacion": { "bsonType": "string" },
      "ips": { "bsonType": "array", "items": { "bsonType": "int" } },
      "nombre": { "bsonType": "string" },
      "numRegistro": { "bsonType": "string" }
    }
  }
}

```

- Servicios

```

{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["_id", "descripcion", "fecha"],
    "properties": {
      "_id": { "bsonType": "string" },
      "descripcion": { "bsonType": "string" },
      "fecha": { "bsonType": "string" }
    }
  }
}

```

- OrdenServicio

```

{
  "$jsonSchema": {
    "bsonType": "object",
    "required": [
      "_id",
      "estado",
      "fecha",

```

```

    "receta",
    "servicios",
    "tipoOrden"
  ],
  "properties": {
    "_id": { "bsonType": "string" },
    "afiliado": {
      "bsonType": "object",
      "required": [
        "_id",
        "beneficiarios",
        "direccion",
        "fechaNacimiento",
        "nombre",
        "tipo",
        "tipoDocumento"
      ]
    },
    "properties": {
      "_id": { "bsonType": "string" },
      "beneficiarios": { "bsonType": "array" },
      "direccion": { "bsonType": "string" },
      "fechaNacimiento": { "bsonType": "string" },
      "nombre": { "bsonType": "string" },
      "tipo": { "bsonType": "string" },
      "tipoDocumento": { "bsonType": "string" }
    }
  },
  "estado": { "bsonType": "string" },
  "fecha": { "bsonType": "string" },
  "idAfiliado": { "bsonType": "string" },
  "idMedico": { "bsonType": "string" },
  "medico": {
    "bsonType": "object",
    "required": [
      "_id",
      "especialidad",
      "identificacion",
      "nombre",
      "numRegistro"
    ]
  },
  "properties": {
    "_id": { "bsonType": "string" },
    "especialidad": { "bsonType": "string" },
    "identificacion": { "bsonType": "string" },
    "nombre": { "bsonType": "string" },
    "numRegistro": { "bsonType": "string" }
  }
},
"receta": { "bsonType": "string" },

```

```
    "servicios": { "bsonType": "array", "items": { "bsonType": "int" } },
    "tipoOrden": { "bsonType": "string" }
  }
}
```

- **Cita**

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": [
      "_id",
      "afiliadoId",
      "disponibilidadId",
      "fechaReserva",
      "ordenId"
    ],
    "properties": {
      "_id": { "bsonType": "string" },
      "afiliadoId": { "bsonType": "string" },
      "disponibilidadId": { "bsonType": "string" },
      "fechaReserva": { "bsonType": "date" },
      "ordenId": { "bsonType": "string" }
    }
  }
}
```

-**Disponibilidad**

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": [
      "_id",
      "estado",
      "fechaHoraFin",
      "fechaHoraInicio",

```



```

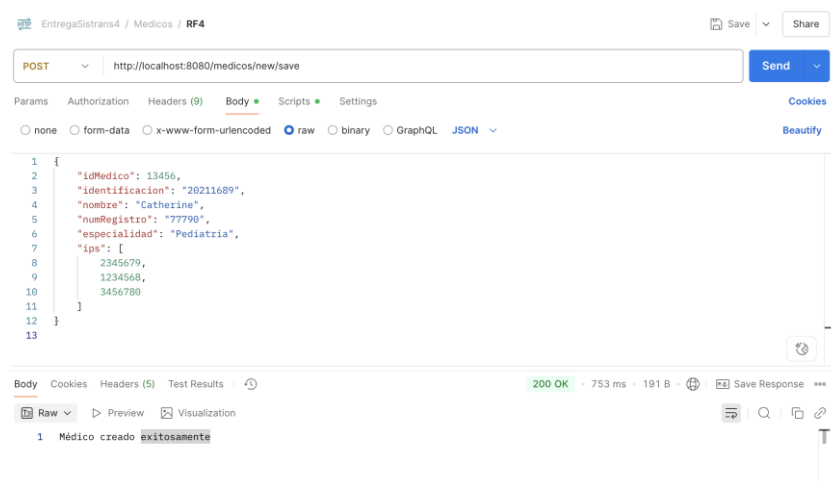
        "idServicio",
        "ipsId",
        "medicoId"
    ],
    "properties": {
        "_id": { "bsonType": "string" },
        "estado": { "bsonType": "string" },
        "fechaHoraFin": { "bsonType": "date" },
        "fechaHoraInicio": { "bsonType": "date" },
        "idServicio": { "bsonType": "int" },
        "ipsId": { "bsonType": "int" },
        "medicoId": { "bsonType": "string" }
    }
}
}
}

```

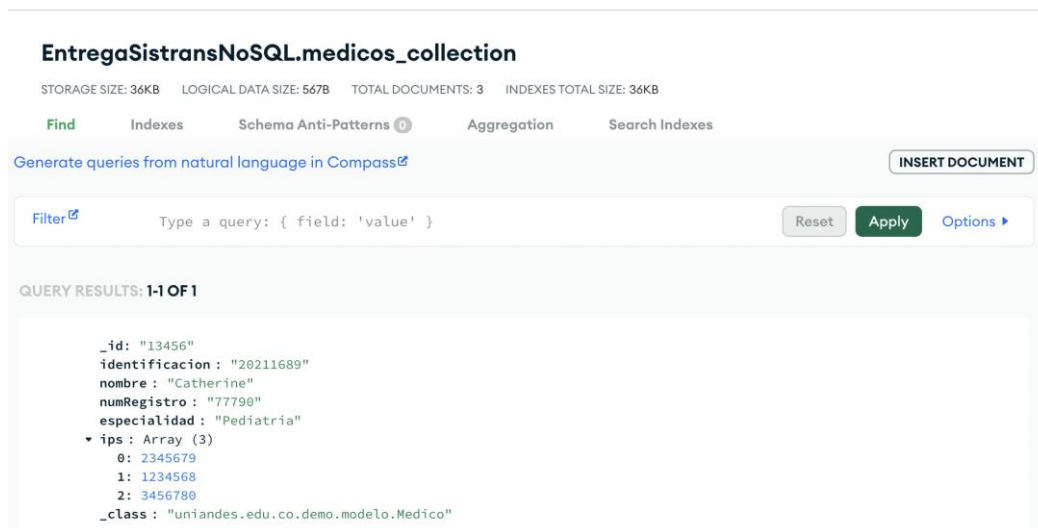
4. Los scripts de la población de la base de datos están adjuntos en el proyecto

5. Implementación requerimientos

RF4 – Registrar médico



Prueba RF4 postman



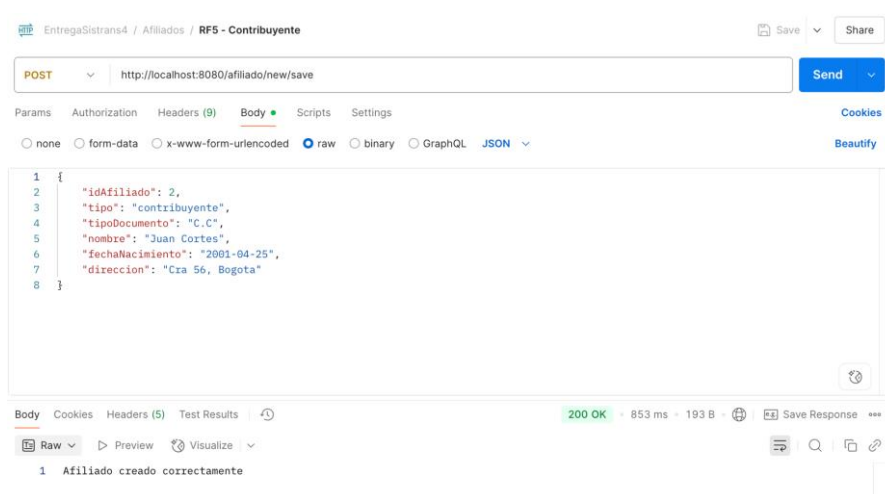
Prueba en Mongo DB

Este requerimiento permite registrar en el sistema un nuevo médico, incluyendo su identificación, nombre, número de registro, especialidad y las IPS en las que trabaja. En la implementación, el campo ips se maneja como una lista de referencias, ya que un médico puede estar asociado a múltiples IPS y esta relación es de muchos a muchos. La operación se probó exitosamente mediante una solicitud POST en Postman, como se evidencia en la imagen, con el endpoint /medicos/new/save. En la solicitud se envió un JSON con todos los datos requeridos, y el sistema respondió con un **código 200 OK** y el mensaje "Médico creado exitosamente", confirmando que el registro fue exitoso.

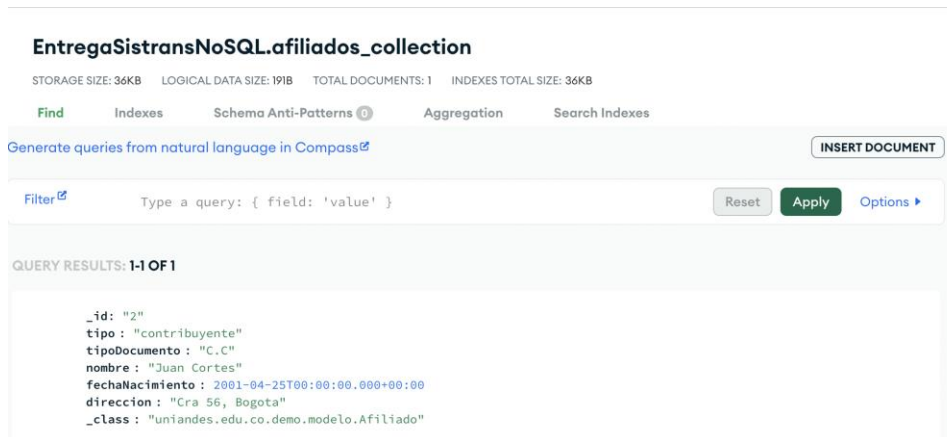
Al revisar la base de datos en MongoDB, se confirma que el documento fue almacenado correctamente en la colección medicos_collection, con los valores esperados. Además, se puede observar que la relación con las IPS fue registrada como un array de identificadores, lo cual sigue la lógica del diseño referenciado discutido en el análisis de esquemas.

Esta implementación cumple con el requerimiento funcional y refleja un uso adecuado de buenas prácticas en modelado NoSQL, asegurando flexibilidad y eficiencia en las consultas y actualizaciones posteriores.

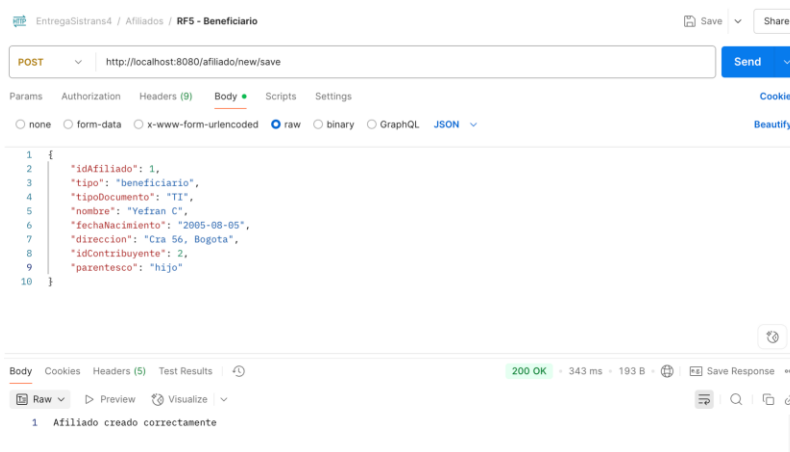
RF5 – Registrar afiliado



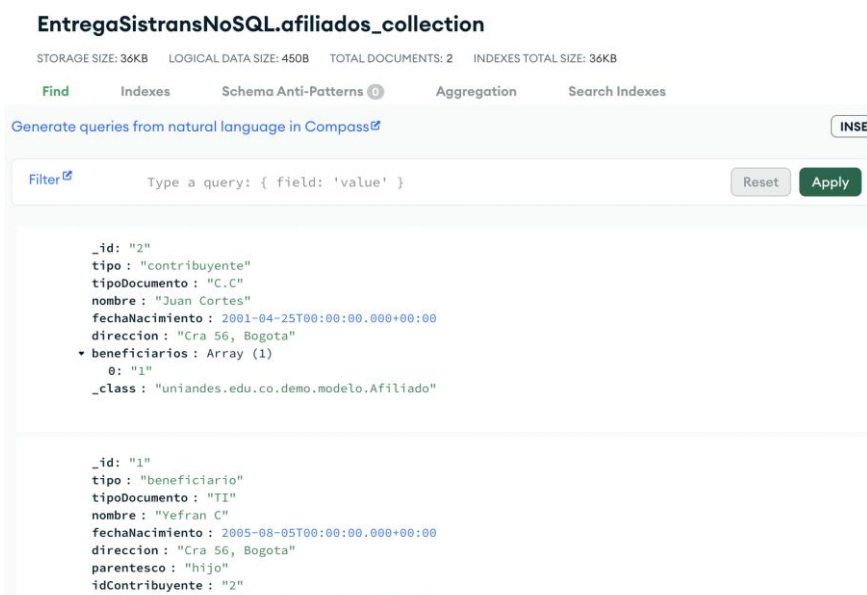
Creación afiliado – tipo Contribuyente



Validación creación Contribuyente



Creación afiliado – tipo Beneficiario



Evidencia MongoDB

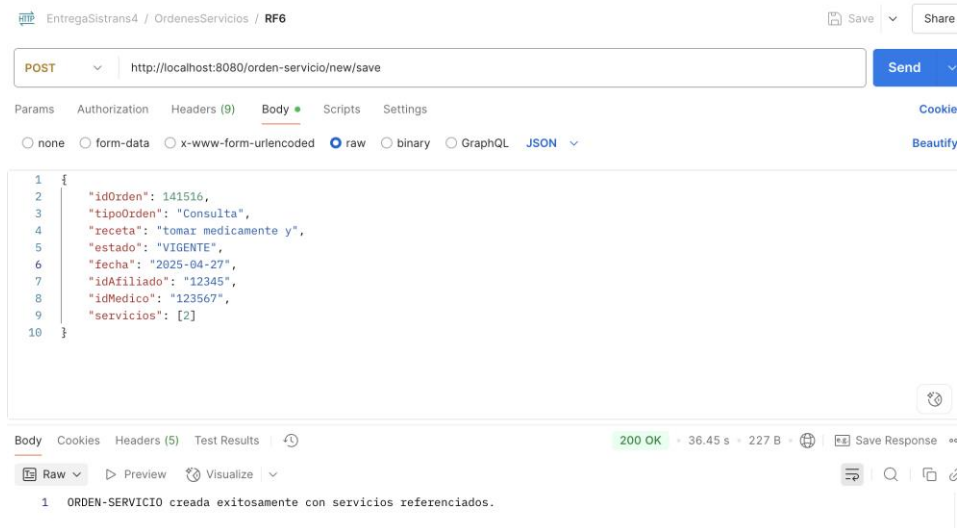
Este requerimiento permite registrar tanto a afiliados de tipo contribuyente como a beneficiarios, vinculando a estos últimos con el contribuyente del cual dependen. La prueba se realizó en dos etapas, como lo requiere el modelo:

1. **Primero se registró un afiliado de tipo "contribuyente"** enviando un JSON con su información básica a través de Postman. El sistema respondió con **200 OK** y el mensaje "Afiliado creado

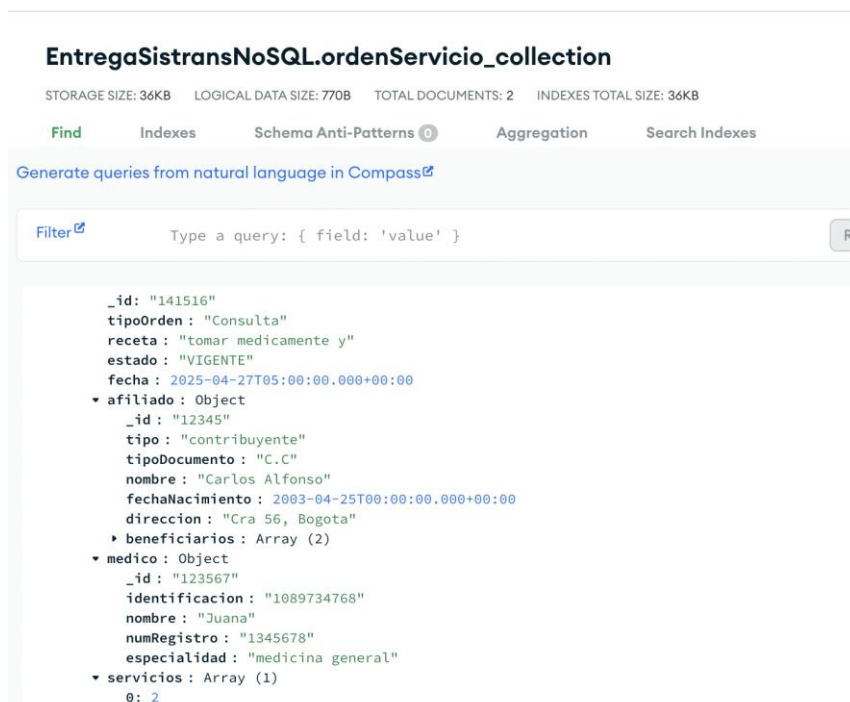
correctamente". Posteriormente, se verificó en MongoDB que el documento fue creado exitosamente en la colección `afiliados_collection`.

2. **Después se registró un afiliado de tipo "beneficiario"**, especificando el campo `idContribuyente` para establecer la relación. Al consultar la base de datos, se pudo comprobar que la referencia quedó registrada en ambos sentidos: el beneficiario tiene guardado el `idContribuyente`, y el contribuyente tiene en su array `beneficiarios` el identificador correspondiente al nuevo dependiente.

RF6 - Registrar una orden de servicio de salud para un afiliado por parte de un médico



Solicitud Postman – RF6



Evidencia en MongoDB

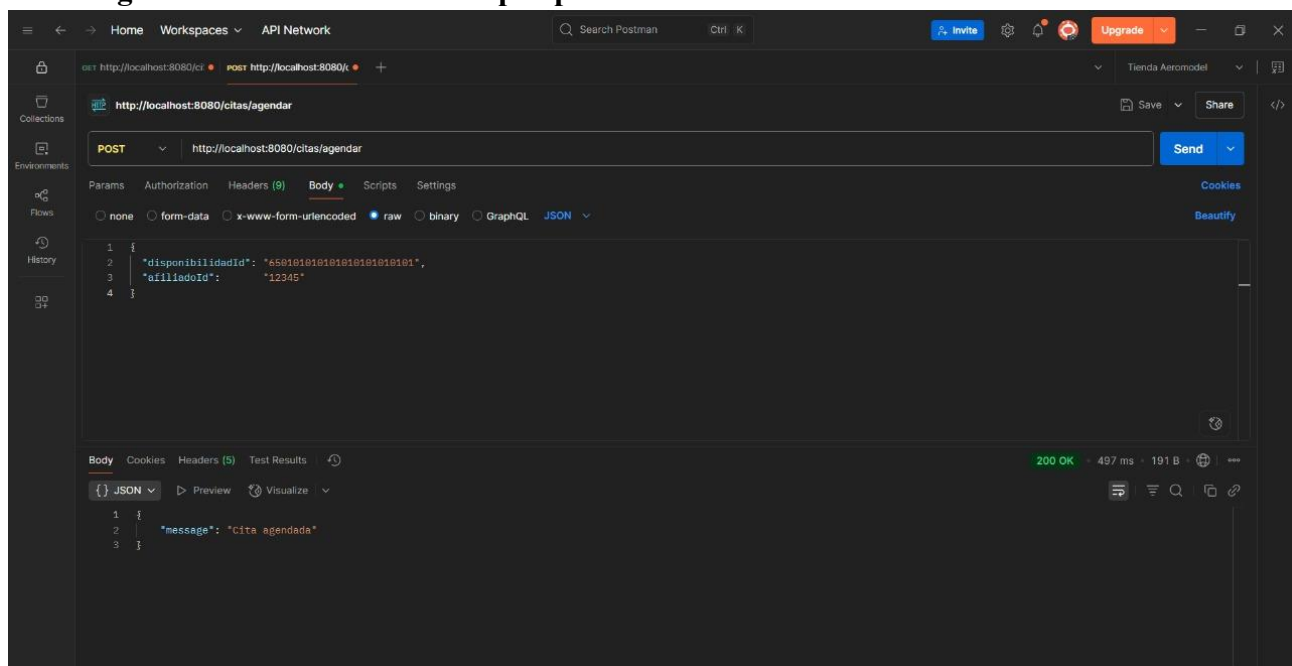
Este requerimiento permite que un médico prescriba una o varias atenciones médicas a un afiliado. La orden queda registrada con información como el tipo de servicio, receta, fecha, estado, médico responsable, afiliado al que está dirigida y los servicios requeridos.

La solicitud se probó mediante un POST en Postman al endpoint /orden-servicio/new/save, donde se envió un JSON con los campos correspondientes: idOrden, tipoOrden, receta, estado, fecha, idAfiliado, idMedico y la lista de servicios como IDs. El servidor respondió con un código 200 OK y el mensaje “ORDEN-SERVICIO creada exitosamente con servicios referenciados”.

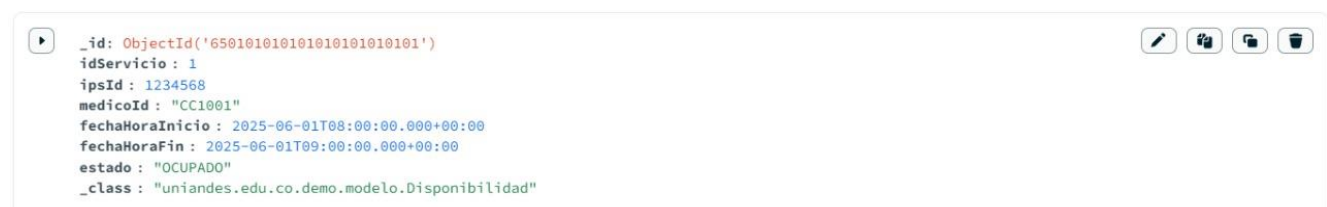
Al consultar la base de datos en MongoDB, se evidencia que la orden fue almacenada correctamente. En el documento correspondiente se observa que:

- El afiliado y el médico fueron embebidos dentro de la orden. Esta decisión se tomó porque cada orden tiene exactamente un afiliado y un médico, y al consultar una orden es común requerir la información completa de ambos. Dado que sus datos no cambian frecuentemente, embebidos mejora el rendimiento sin afectar la consistencia.
- La relación con los servicios de salud se maneja como referenciada, guardando únicamente los IDs. Esto se debe a que cada orden puede incluir múltiples servicios, y al mismo tiempo, un servicio puede estar en muchas órdenes. Duplicar toda la información del servicio generaría un modelo más pesado y difícil de mantener, por lo que el uso de referencias es más apropiado en términos de escalabilidad y reutilización.

RF7 - Agendar un servicio de salud por parte de un afiliado



Solicitud postman RF7

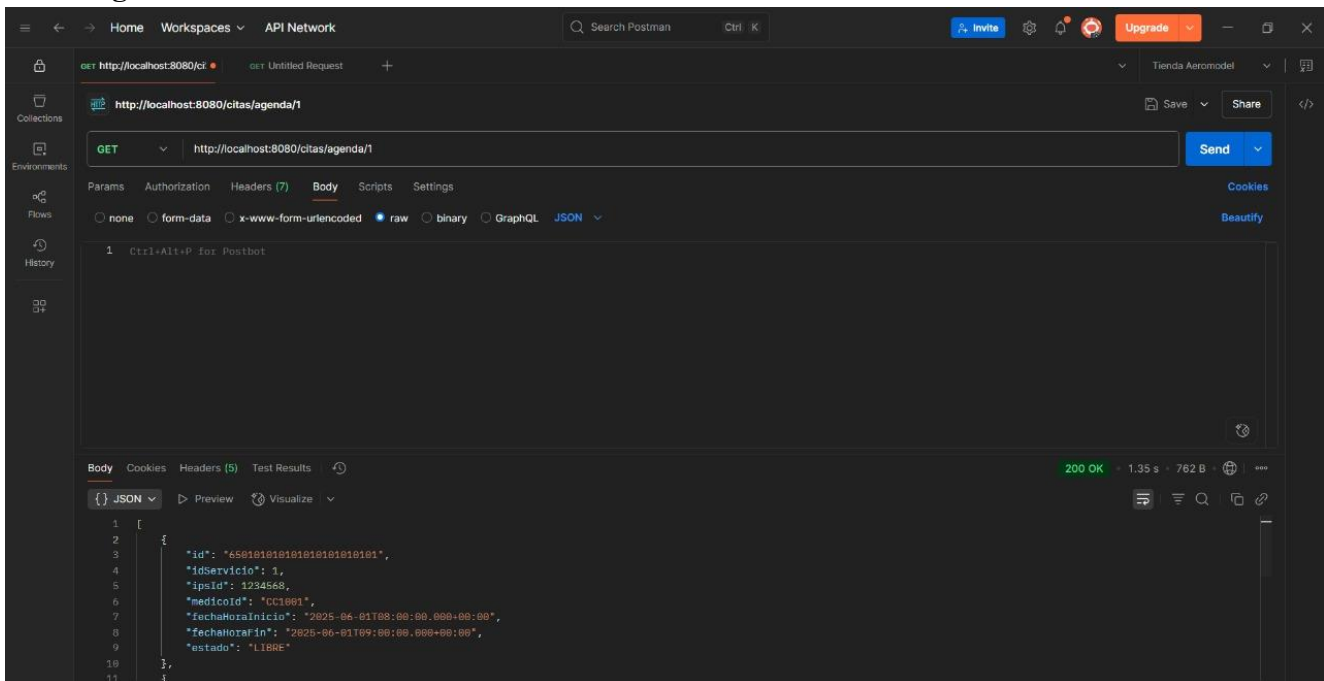


```
_id: ObjectId('6833817438f3af015ffebd38')  
disponibilidadId: "3"  
afiliadoId: "12345"  
fechaReserva: 2025-05-25T20:45:40.819+00:00  
ordenId: "7"  
_class: "uniandes.edu.co.demo.modelo.Cita"
```

Evidencia en MongoDB

Para verificar el RF7 – Agendar cita, en Postman configuras una petición POST a `http://localhost:8080/citas/agendar`, con el encabezado `Content-Type: application/json`. En el cuerpo debes enviar un objeto JSON con al menos los campos "disponibilidadId" (el id de la franja libre que obtuviste con el RFC1) y "afiliadoId" (el documento del afiliado que reserva). Si el servicio no es de tipo consulta general (idServicio distinto de 1 u 2), se debe incluir también "ordenId" con el identificador de una orden de servicio válida. Al pulsar “Send”, si la franja aún está libre y los datos son correctos, recibirás un 200 OK con `{"message": "Cita agendada"}`; en caso de intentar reservar una franja ya ocupada obtendrás un 409 Conflict, y si falta o es inválida la orden, un 400 Bad Request.

RC1 - Consultar la agenda de disponibilidad que tiene un servicio de salud ingresado por el usuario en las siguientes 4 semanas.



Solicitud postman RFC1

Para probar el RFC1 – Consultar disponibilidad, en Postman debes enviar una petición GET a la URL `http://localhost:8080/citas/agenda/{idServicio}`, reemplazando {idServicio} por el identificador del servicio cuyo calendario quieres consultar (por ejemplo, 1 para consulta general). Esta llamada no lleva cuerpo (body) y espera recibir un array JSON con cada franja disponible —cada objeto contendrá campos como id, idServicio, ipsId, medicoId, fechaHoraInicio, fechaHoraFin y estado igual a "LIBRE". Si todo funciona correctamente, Postman mostrará un 200 OK y la lista de franjas en ese intervalo de las próximas cuatro semanas; si ocurre algún error de parametrización o interno, verás el código de estado correspondiente

RC2 - Mostrar los 20 servicios más solicitados.

GET RFC2

EntregaSistrans4 / Agenda / RFC2

GET http://localhost:8080/citas/mas-usados?desde=2021-05-24&hasta=2026-05-27

Params
 Authorization
 Headers (7)
 Body
 Scripts
 Settings

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description
<input checked="" type="checkbox"/>	desde	2021-05-24	
<input checked="" type="checkbox"/>	hasta	2026-05-27	
	Key	Value	Description

Body
 Cookies
 Headers (5)
 Test Results
 200 OK 906 ms 1.04 KB

{} JSON
 Preview
 Visualize

```

1  [
2    {
3      "total": 2,
4      "descripcion": "Consulta con especialista"
5    },
6    {
7      "total": 2,
8      "descripcion": "Control prenatal"
9    },
10   {
11     "total": 2,
12     "descripcion": "Consulta con médico general"
  
```

Solicitud postman RFC2

Para probar el requerimiento funcional de consulta 2 se necesita la ruta <http://localhost:8080/citas/mas-usados> y pasar como parámetros la fecha de inicio y de fin en el formato (“yyyy-mm-dd”). La función de este requerimiento es obtener los 20 servicios más usados en determinada fecha, para realizar esa función filtra entre todos los documentos de Citas por fecha reservada, esto con el fin de saber si el usuario cuando registro su cita está dentro del intervalo de tiempo que buscamos, luego, realiza un \$lookup con las disponibilidades con el fin de validar que la disponibilidad asociada está ocupada ya que contaremos las disponibilidades ocupadas, hacemos otro \$lookup con servicios para obtener las descripciones y luego retornamos en orden descendente los 20 primeros servicios.

Escenarios de prueba

```

db.medicos_collection.insertOne({
  _id: 123,                // >> debe ser string
  identificacion: 202116259, // >> debe ser string
  nombre: "Dr. Prueba"
  // Falta numRegistro y especialidad
});
// → Error 121: Document failed validation
✖ ▶ MongoServerError: Document failed validation
  
```

```

db.afiliados_collection.insertOne({
  _id: 50,                // >> debe ser string
  tipo: "beneficiario",
  tipoDocumento: "CC",
  nombre: "Test",
  // falta fechaNacimiento y direccion
  parentesco: 123,        // >> debe ser string
  // falta idContribuyente
});
// → Error 121: Document failed validation
✖ ▶ MongoServerError: Document failed validation

```

RF5

```

db.ordenServicio_collection.insertOne({
  _id: "0100",
  idOrden: 100,           // >> debe ser string
  tipoOrden: "Consulta",
  receta: "Tomar X",
  estado: "VIGENTE",
  fecha: "2025-06-01",    // >> debe ser ISODate
  idAfiliado: "12345"
  // faltan idMedico y servicios
});
// → Error 121: Document failed validation
✖ ▶ MongoServerError: Document failed validation

```

RF6

```

db.citas_collection.insertOne({
  _id: "C100",
  disponibilidadId: 21,    // >> debe ser string
  afiliadoId: "12345"
  // falta fechaReserva
});
// → Error 121: Document failed validation
✖ ▶ MongoServerError: Document failed validation

```

RF7

