Documentación Proyecto 3

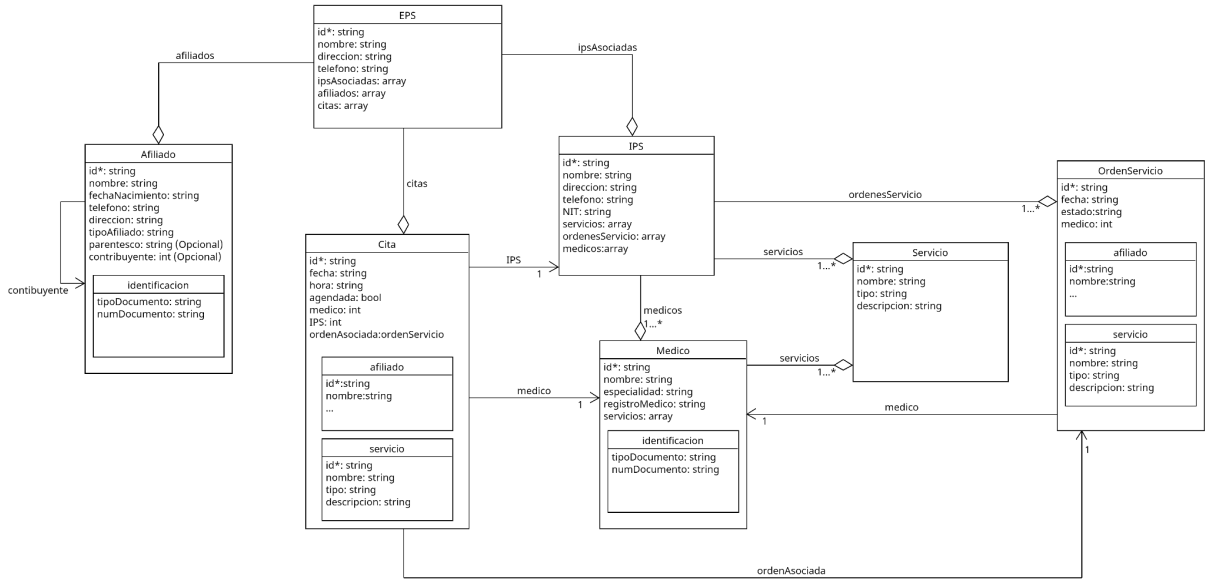
Grupo 6
Emmanuel Blanco 202312743
Nicolás Parra 202322257
Santiago Cordero 202312046

1. **Tras la revisión del caso de estudio se obtuvieron los siguientes elementos cruciales para la representación del negocio:**

    - EPS
    - IPS
    - Afiliados
    - Medicos
    - Servicios
    - Citas
    - Agenda de servicios
    - Ordenes de servicio

Todos esto elementos son cruciales para las principales operaciones que se hacen sobre el negocio, por lo que la gran mayoría se podrían representar luego como una entidad en el modelo

2. **Modelo UML del modelo**



Este diagrama también se incluirá en el repositorio del proyecto como archivo pdf

***** En el diagrama los ids realmente no harían parte del modelo, ya que estos son asignados por default en mongo a las entidades

3. **Diseño de la base de datos para todos los requerimientos funcionales**

Para los elementos pedidos en esta parte nos basaremos en el modelo presentado en el punto anterior, donde presentamos el modelo que usaremos con sus entidades, y los atributos correspondientes a las distintas entidades.

A continuación se presentará el listado de entidades con la descripción de sus atributos:

I. EPS

Atributos
- nombre: nombre de la EPS
- direccion: dirección asociada a las instalaciones de la EPS
- telefono: telefono de contacto de la EPS
- ipsAsociadas: lista que contiene los ids de las IPS contratadas por la EPS
- afiliados: lista que contiene los ids de los afiliados a la EPS
- citas: lista de los ids de las citas asociadas a la EPS, es decir, de las citas de todas las IPS asociadas a la EPS

II. IPS

Atributos
- nombre
- direccion
- telefono
- NIT
- servicios: lista con los ids de los servicios que presta la IPS
- ordenesServicio: lista con los ids de las órdenes de servicios que presta la IPS
- medicos: lista con los ids de los medicos que prestan servicio dentro de la IPS

III. Afiliado

Atributos
- nombre
- fechaNacimiento
- telefono
- direccion
- tipoAfiliado: string que determina si un afiliado es beneficiario o contribuyente
- parentesco (Opcional): parentesco de un beneficiario con su contribuyente, este atributo solo lo tienen los afiliados beneficiarios
- contribuyente (Opcional): id del contribuyente asociado, solo lo tienen los afiliados que son beneficiarios
- identificacion: objeto que incluye información acerca del tipo de documentación del afiliado y su número de identificación
    - ❖ tipoDocumento: tipo de documento del afiliado
    - ❖ numDocumento: numero de documento del afiliado

IV. Medico

Atributos
- nombre
- especialidad
- registroMedico

- servicios: lista con los ids de los servicios que presta el médico
- identificacion: objeto que incluye información acerca del tipo de documentación del afiliado y su número de identificación
    - ❖ tipoDocumento: tipo de documento del médico
    - ❖ numDocumento: numero de documento del médico

V. Cita

Esta entidad se usará para representar los servicios que ya fueron agendados y también aquellos que están disponibles

Atributos

- fecha
- hora
- agendada: booleano que indica si la cita ya está reservada (true) o no (false)
- medico: id del médico asociado a la cita
- IPS: id de la IPS donde se dará la cita
- afiliado (Opcional): es un objeto con la misma estructura de Afiliado donde se detalla la información del afiliado asociado a la cita. Las citas que aún están disponibles no deberían tener este atributo
- servicio: objeto de tipo Servicio donde se detalla la información del servicio que se prestará en la cita

VI. Servicio

Atributos

- nombre
- tipo
- descripcion

VII. OrdenServicio

Atributos

- fecha
- estado
- medico: id del médico que expide la orden
- afiliado: es un objeto con la misma estructura de Afiliado donde se detalla la información del afiliado asociado a la orden.
- servicio: objeto de tipo Servicio donde se detalla la información del servicio que se prestará en con la orden

<u>Relaciones entre entidades con sus cardinalidades y su representación:</u>

- ☐ EPS - IPS
    - Tipo: Uno a Muchos (Sentido EPS → IPS), Uno a Uno (Sentido IPS → EPS)
    - Una EPS contrata a múltiples IPS, pero una IPS solo puede ser contratada por una EPS
    - Representación por referencia, ya que el lado de la cardinalidad de IPS puede crecer

☐ EPS - Cita
- Tipo: Uno a Muchos (Sentido EPS → Cita)
- Una EPS tiene varias citas asociadas, tiene asociadas todas las citas de los servicios de sus IPS
- Representación por referencia, ya que el lado de la cardinalidad de Cita puede crecer

☐ EPS - Afiliado
- Tipo: Uno a Muchos (Sentido EPS → Afiliado), Uno a Uno (Sentido Afiliado → EPS)
- Una EPS tiene muchos afiliados, y un afiliado solo pertenece a una EPS
- Representación por referencia, ya que el lado de la cardinalidad de Afiliado puede crecer

☐ Cita - IPS
- Tipo: Uno a Uno (Sentido Cita → IPS)
- Una cita está vinculada a una sola IPS
- Representación por referencia porque embeber en una cita la IPS aumentaría su ocupamiento de espacio en memoria

☐ Cita - Afiliado
- Tipo: Uno a Uno (Sentido Cita → Afiliado)
- Una cita solo está asociada a un Afiliado
- Representación embebida ya que facilita el acceso a la información del afiliado que podría ser consultado en conjunto con la de la cita

☐ Cita - Servicio
- Tipo: Uno a Uno (Sentido Cita → Servicio)
- Una cita solo está asociada a un servicio
- Representación embebida ya que facilita el acceso a la información del servicio que podría ser consultado en conjunto con la de la cita

☐ Cita - Medico
- Tipo: Uno a Uno (Sentido Cita → Medico)
- Una cita solo está asociada a un médico
- Representación por referencia ya que incluyendo la información del médico se podría aumentar el espacio en memoria de Cita

☐ Medico - Servicio
- Tipo: Uno a Muchos (Sentido Medico → Servicio)
- Un médico presta varios servicios
- Representación por referencia ya que el número de servicios asociados al médico puede crecer

☐ IPS - Medico
- Tipo: Uno a Muchos(Sentido IPS → Medico)

- Una IPS contrata a varios médicos
- Representación por referencia ya que los médicos asociados a la IPS puede aumentar, y añadir los médico a IPS haría que su tamaño aumente considerablemente

☐ IPS - Servicio
- Tipo: Uno a Muchos (Sentido IPS → Servicio)
- Una IPS presta varios servicios
- Representación por referencia ya que los servicios asociados a la IPS puede aumentar, y añadir los servicios a IPS haría que su tamaño aumente considerablemente

☐ IPS - OrdenServicio
- Tipo: Uno a Muchos (Sentido IPS → Ordenservicio)
- Una IPS tiene varias órdenes asociadas a los servicios que presta
- Representación por referencia ya que las órdenes de servicio asociadas a la IPS puede aumentar, y añadirlas a IPS haría que su tamaño aumente considerablemente

☐ OrdenServicio - Medico
- Tipo: Uno a Uno (Sentido OrdenServicio → Medico)
- Una orden está asociada a un solo médico
- Representación por referencia ya que incluyendo la información del médico se podría aumentar el espacio en memoria de OrdenServicio

☐ OrdenServicio - Afiliado
- Tipo: Uno a Uno (Sentido OrdenServicio → Afiliado)
- Una orden está asociada a un solo afiliado
- Representación embebida ya que facilita el acceso a la información del afiliado que podría ser consultado en conjunto con la de la orden

☐ OrdenServicio - Servicio
- Tipo: Uno a Uno (Sentido OrdenServicio → Servicio)
- Una orden está asociada a un solo servicio
- Representación embebida ya que facilita el acceso a la información del afiliado que podría ser consultado en conjunto con la de la orden

Representaciones en el archivo json

La representación la podrán revisar en el archivo creacionColecciones.json, archivo que tiene los scripts usados para la creación de las colecciones en MongoDB y donde los elementos embebidos son atributos de tipo object con la misma estructura que los documentos de la colección a la que representan y los elementos por referencia son de tipo string o List<String> que contienen el id del objeto al que referencian. En este mismo archivo se muestran los esquemas de validación de las colecciones.

Poblado de la base de datos

La base de datos fue poblada con los scripts en el archivo datosPoblarDB.json.

Requerimiento funcionales

- RF4: Este requerimiento se implementa en MedicoController y toma como parámetros el id del médico que se les quiere hacer las operaciones y una ips. Se llama la función getMedicos y se recorre la lista que devuelva. Esto con el fin de verificar si el del médico en cuestión ya se encuentra dentro de la ips o no. Si está registrado en la ips se retorna un texto diciendo que ya está registrado y se devuelve un status de tipo conflict. Si no está registrado, el id del médico se agrega en la lista de médicos de la ips correspondiente y se devuelve un status created. En caso que ocurra cualquier otra cosa se devuelve un internal server error.

- RF5: Este requerimiento se implementa en AfiliadoController y toma como parámetros el id del afiliado que se les quiere hacer las operaciones y una eps. Se llama la función getAfiliados y se recorre la lista que devuelva. Esto con el fin de verificar si el del afiliado en cuestión ya se encuentra dentro de la eps o no. Si está registrado en la eps se retorna un texto diciendo que ya está registrado y se devuelve un status de tipo conflict. Si no está registrado, el id del afiliado se agrega en la lista de afiliados de la eps correspondiente y se devuelve un status created. En caso que ocurra cualquier otra cosa se devuelve un internal server error.

- RF6:Este requerimiento está implementado en el OrdenServicioController, que maneja las operaciones CRUD sobre las órdenes de servicio. Los métodos principales son:
  Crear orden de servicio: Recibe un objeto JSON con toda la información necesaria (estado, fecha, afiliado, servicio, médico) y la guarda en la base de datos.
  Obtener todas las órdenes de servicio: Devuelve la lista completa de órdenes registradas.
  Obtener orden de servicio por ID: Busca una orden específica a partir de su identificador único.
  Actualizar orden de servicio: Permite modificar los atributos de una orden existente.
  Eliminar orden de servicio: Borra una orden en base a su ID.
  Estos métodos forman el núcleo para registrar y mantener actualizadas las órdenes de servicio de salud.Para probar este requerimiento, se creó una colección en Postman llamada OrdenServicio con solicitudes para crear, consultar, actualizar y eliminar órdenes. Se recomienda probar primero la creación de una orden, luego consultar la lista o la orden específica, actualizarla y finalmente eliminarla, verificando que las operaciones se ejecutan correctamente y los datos cambian en la base de datos.
- RF7: Este requerimientos usa dos métodos en CitaController, el primero obtenerDisponibilidadSig4Semanas, el cual es el mismo método usado para el RFC1, retorna las citas relacionadas a un servicio que aún no están agendadas (que en nuestro modelo equivale a ver la disponibilidad de un servicio) en un rango desde una fecha inicial ingresada por el usuario y la fecha que está 4 semanas después de la fecha inicial; y el segundo agendarCita que agenda la cita usando un body que sigue la estructura de cita para cambiar los atributos agendada, afiliado y ordenAsociada de la cita que el usuario quiera agendar.

Si bien los anteriores dos métodos no tienen conexión entre sí en este API, si se implementará un front, el primer método se usaría en una vista de citas disponibles y al seleccionar una de las citas en Citas disponibles se usaría el método agendarCita para cambiar el elemento seleccionado.

Para probar este requerimiento se hicieron dos requests en Postman RFC1 y Agendar cita. Se recomienda para probar este requerimiento primero correr RFC1, luego buscar en la base de Mongo alguno de los elementos que se desplegaron como respuesta del RFC1 y usarlo en Agendar Cita (incluyendo en la variable id de la url el id de la cita que se quiere agendar) para probar que la cita se agenda exitosamente.

Requerimientos de consulta

- RFC1: Lo relacionado a este requerimiento lo pueden revisar en el repositoryCustom de Cita (específicamente en obtenerDisponibilidadSig4Semanas) y en el controller de Cita luego del comentario “Metodo para resolver el RFC1” que marca el método que se usa para este consulta. Adicionalmente se incluye el script original que se puede usar en el shell de mongo para hacer la consulta en el archivo consultas.json.

  La prueba relacionada a este requerimiento está en la coleccion de Cita con el nombre RFC1, para probarlo solo es necesario ingresar el nombre de algún servicio y una fecha inicial en la parte de parámetros de Postman y estar corriendo la aplicación, en Postman la respuesta debería ser una lista de documentos que muestren la información que fue descrita en el enunciado

- RFC2: La lógica de este requerimiento está implementada en el CitaRepositoryCustom, específicamente en el método topServiciosMasSolicitados(). Este método realiza una agregación en la colección cita para contar la frecuencia de servicios solicitados y devuelve los 20 con mayor número de citas.El endpoint para acceder a esta consulta está en el CitaController bajo la ruta GET /citas/top-servicios.Para probar esta consulta, se creó una solicitud en Postman dentro de la colección Cita, llamada RFC2 - Top Servicios, que al ejecutarse devuelve un listado JSON con los servicios y el número de veces que fueron solicitados en las citas registradas.Se recomienda ejecutar esta solicitud con la aplicación corriendo y con datos de citas creados para visualizar correctamente el resultado.