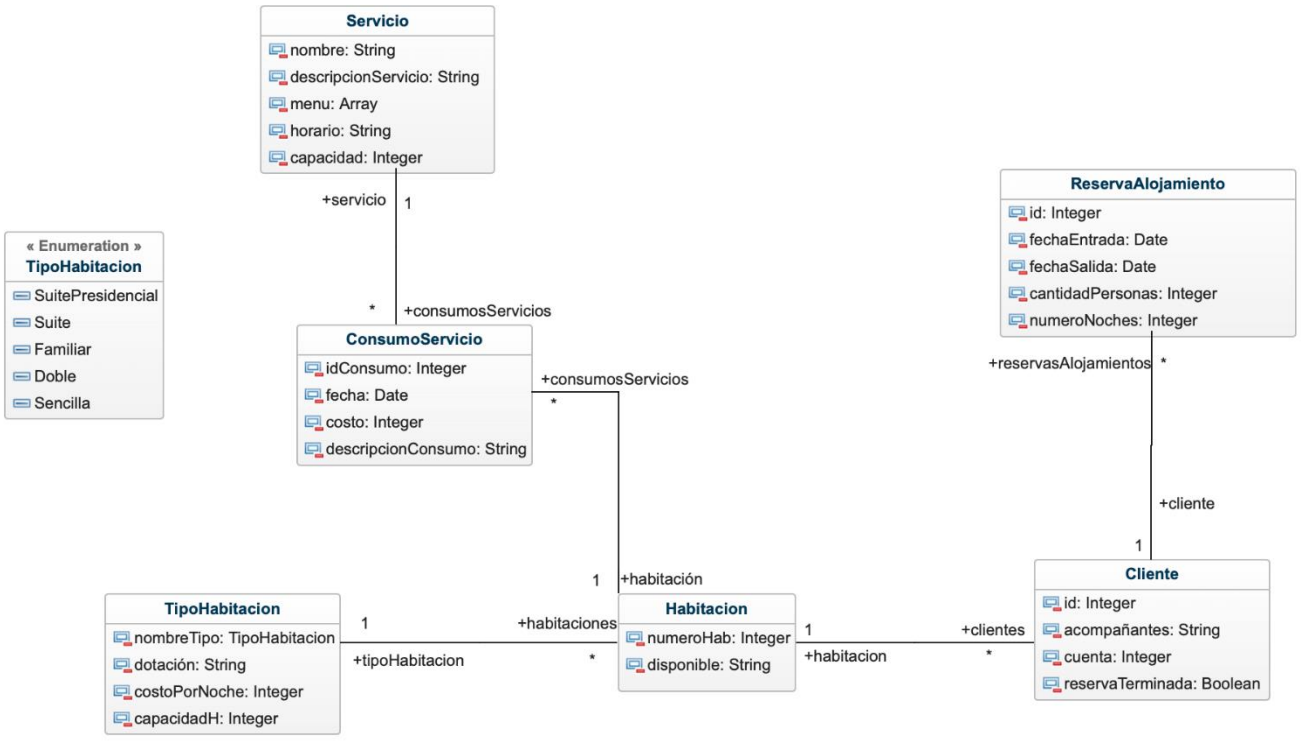
Alejandro Gonzalez

Paulina Arrázola

Natalia Ortega

PROYECTO 3

UML:



2.

a)

Entidades y atributos:

ENTIDAD	ATRIBUTOS
Servicios	nombre (String)
	descripcionServicio (String)
	menu (String)
	Horario (String)
	Capacidad (Integer)
ConsumosServicios	idConsumo (Integer)
	fecha (Date)
	costo (Integer)
	descripcionConsumo (String)
Habitaciones	numeroHab (Integer)
	disponible (Boolean)
TiposHabitaciones	nombreTipo (enum)
	dotacion (String)
	costoPorNoche (Integer)
	capacidadH (Integer)
Clientes	id (Integer)
	acompañantes (Integer)
	cuenta (Integer)
	reservaTerminada (Boolean)
ReservasAlojamientos	id (Integer)
	fechaEntrada (Date)
	fechaSalida (Date)
	cantidadPersonas(Integer)
	numeroNoches (Integer)

b)

Cuantificación de entidades

ENTIDAD	CANTIDAD DE REGISTROS
Servicios	35
ConsumosServicios	250000
Habitaciones	200
TiposHabitaciones	20
Clientes	50000
ReservasAlojamientos	50000

C y D)

Análisis y cuantificación de operaciones de lectura y escritura

ENTIDADES	OPERACIÓN	INFORMACIÓN NECESARIA	TIPO	TASA
Servicios	Añadir/actualizar un servicio	Lista de servicios	Escritura	2x semana
Servicios	Traer un servicio	Lista de servicios e identificador único	Lectura	2x semana
Servicios	Borrar un servicio	Identificador único del servicio	Escritura	1 x año
Tipo de Habitacion	Añadir/actualizar un tipo de habitación	Detalle del tipo de habitación	Escritura	1 x semana
Tipo de Habitacion	Borrar un tipo de habitación	Identificador único del tipo de habitación	Escritura	1 x año
Tipo de Habitacion	Traer un tipo de habitación	Detalle del tipo de habitación	Lectura	1 x semana
ConsumoServicio	Añadir/actualizar un ConsumoServicio	Lista de consumos de servicios	Escritura	500 x semana
ConsumoServicio	Borrar un consumo de servicio	Identificador único del consumo de servicio	Escritura	2 x semana
ConsumoServicio	Traer un consumo de servicio	Detalle del consumo de servicio	Lectura	250 x semana

Habitacion	Añadir/actualizar una habitación	Detalle de la habitación	Escritura	2 x semana
Habitacion	Borrar una habitación	Identificador único de la habitación	Escritura	1 x mes
Habitacion	Traer una habitación	Detalle de la habitación	Lectura	2 x semana
ReservaAlojamiento	Añadir/actualizar una reserva de alojamiento	Detalle de una reserva de alojamiento	Escritura	100 x día
ReservaAlojamiento	Borrar una reserva de alojamiento	Identificador único de una reserva de alojamiento	Escritura	50 x semana
ReservaAlojamiento	Traer una reserva de alojamiento	Detalle de una reserva de alojamiento	Lectura	500 x día
Cliente	Registrar la entrada de un cliente	Detalle de un cliente	Escritura	100 x día
Cliente	Borrar la entrada de un cliente	Identificador unico de un cliente	Escritura	10 x año
Cliente	Traer la entrada de un cliente	Detalle de un cliente	Lectura	50 x día
Cliente	Registrar la salida de un cliente	Consumo total del cliente y paz y salvo	Escritura	100 x día
Cliente	Borrar la salida de un cliente	Identificador único de un cliente	Escritura	10 x año
Cliente	Traer la salida de un cliente	Consumo total del cliente y paz y salvo	Lectura	50 x día
ConsumoServicio, Habitaciones	Consultar dinero recolectado por servicios pedidos a una habitación	Cantidad total de dinero recolectado por la habitación gracias a sus servicios	Lectura	50 x mes
Habitacion, Cliente	Consultar el índice de ocupación de cada habitación en el año	Lista de cada habitación con sus respectivos porcentajes de ocupación	Lectura	200 x semestre

Cliente, Habitación, ConsumoServicio	Mostrar consumo de un cliente en un determinado rango de fechas	Cantidad total de dinero gastado por un cliente, fechas de gasto	Lectura	50 x mes
ReservasAlojamientos, Clientes	Consultar clientes que se hayan hospedado al menos una vez por trimestre	_id cliente, detalle reserva	Lectura	4 x año

2.

a)

Descripción de las entidades

Habitaciones:

La entidad habitaciones representa todas las habitaciones existentes en el hotel. De estas solo se conoce si están ocupadas y su identificación única, pues sus detalles están alojados en la entidad TiposHabitaciones.

TiposHabitaciones:

La entidad tiposHabitaciones contine los posibles tipos de habitaciones que puede tener el hotel. Estos tipos particulares son: Suite, SuitePresidencial, Familiar, Doble, Sencilla; aunque se podrían agregar tipos adicionales.

Servicios:

La entidad representa todos los servicios que presta el hotel. En caso de que el servicio sea un restaurante, una tienda, un supermercado o un bar, la entidad contiene un menú de los productos que se pueden adquirir en estos servicios. Algunos de los servicios están incluidos en el costo de hospedaje y otros son adicionales a la tarifa base.

ConsumosServicios:

La entidad contiene la información de un consumo que el cliente hace de un servicio en el hotel. En particular, se tiene la fecha, el costo y la descripción de lo que el cliente consumió en este servicio.

Clientes:

La entidad representa el registro de un cliente que pasa por el hotel. En específico, la entidad es creada con la entrada de un cliente al hotel para registrar su información. Adicionalmente, el estado de la entidad cambia cuando el cliente sale del hotel. Sin embargo, en la base de datos queda los datos de la estadía del cliente para consultar.

Reservas alojamiento:

La entidad representa la reserva de un cliente en el hotel con toda su información relevante. Este contiene información relevante de estadía como las fechas de entrada y salida, la cantidad de noches que se quedó y la cantidad de personas que se quedaron, pues esta información es útil para las consultas que son requeridas en el hotel.

b)

ANÁLISIS DE CARDINALIDAD

Servicios-ConsumosServicios: relación uno a muchos respectivamente. Esto se modela de esta manera porque un servicio puede ser consumido múltiples veces, sin embargo, solo se puede consumir un servicio a la vez.

TiposHabitaciones-Habitaciones: relación uno a muchos respectivamente. Se modela de esta manera porque muchas habitaciones pueden compartir el mismo tipo de habitación, sin embargo, una habitación no puede ser de dos tipos a la vez.

Habitaciones-ConsumosServicios: relación uno a muchos respectivamente. Se modela de esta manera porque los miembros de una habitación pueden consumir múltiples servicios, sin embargo, un consumo solo lo pudo hacer alguien que se alojaba en una única habitación. Cabe resaltar que esto se remonta a la política del hotel de asociar un consumo a la habitación y no al cliente.

Habitaciones-Clientes: relación uno a muchos respectivamente. Se modela de esta manera porque a lo largo de la historia del hotel múltiples clientes se pueden hosperdar en la misma habitación, sin embargo, se modela de tal manera que un cliente solo puede estar en una habitación por estadía. Esto permite la flexibilidad de que un mismo cliente se quede en diferentes habitaciones siempre y cuando sea en diferentes periodos de tiempo. Cabe resaltar que por regla de negocio no es posible que dos clientes se queden en la misma habitación al mismo tiempo.

Clientes-ReservasAlojamiento: relación uno a muchos. Se modela de esta manera porque un cliente puede realizar una reserva en el hotel en múltiples ocasiones, sin embargo, una reserva solo puede estar asociada a un cliente.

c)

ANÁLISIS DE ESQUEMA DE ASOSIACIÓN

RELACIÓN: SERVICIOS (HIJO)-CONSUMOSERVICIOS (PADRE)

NOMBRE	PREGUNTA	EMBEBIDO	REFERENCIADO
Simplicidad	¿Mantener las piezas de información junta llevaría a un modelo y código más simple?	SI	NO
Van juntos	¿Las piezas de información tienen un 'tiene un', 'contiene' o una relación similar?	SI	NO
Atomicidad de la consulta	¿La aplicación consulta las piezas de información juntas?	SI	NO
Complejidad de actualización	¿Son las piezas de información actualizadas juntas?	SI	NO
Archivo	¿Deberían archivarse los datos al mismo tiempo?	SI	NO
Cardinalidad	¿Hay una alta cardinalidad (actual o creciente) por parte del hijo de la relación?	NO	SI
Duplicidad de datos	¿La duplicación de los datos sería muy difícil de manejar?	NO	SI
Tamaño del documento	¿La combinación del tamaño de la información tomaría mucho espacio en memoria o ancho de banda para la aplicación?	NO	SI

Crecimiento del documento	¿La pieza de información embebida crecerían sin límites?	NO	SI
Carga de trabajo	¿Son las piezas de información escritas en tiempos diferentes en una carga de escritura pesada?	NO	SI
Individualidad	¿Para el hijo de la relación, puede existir sin el padre?	NO	SI

Si tenemos en cuenta los criterios de selección de un esquema de asociación anteriormente presentados claramente se ve una tendencia hacia modelar esta relación como una embebida. Además, si tenemos en cuenta los resultados del análisis de la carga de trabajo, vemos que esto no representaría un problema pues la carga de trabajo de la relación que se embebe, en este caso Servicios, es muy baja y no atentaría contra el mantenimiento de la base de datos. Por lo anteriormente expuesto se toma la decisión de que la relación entre Servicio y ConsumosServicios sea una relación emebebida.

RELACIÓN: HABITACIONES (PADRE)- TIPOSHABITACIONES(HIJO)

NOMBRE	PREGUNTA	EMBEBIDO	REFERENCIADO
Simplicidad	¿Mantener las piezas de información junta llevaría a un modelo y código más simple?	SI	NO
Van juntos	¿Las piezas de información tienen un 'tiene un', 'contiene' o una relación similar?	SI	NO
Atomicidad de la consulta	¿La aplicación consulta las piezas de información juntas?	SI	NO
Complejidad de actualización	¿Son las piezas de información actualizadas juntas?	SI	NO
Archivo	¿Deberían archivarse los datos al mismo tiempo?	SI	NO
Cardinalidad	¿Hay una alta cardinalidad (actual o creciente) por parte del hijo de la relación?	NO	SI

Duplicidad de datos	¿La duplicación de los datos sería muy difícil de manejar?	NO	SI
Tamaño del documento	¿La combinación del tamaño de la información tomaría mucho espacio en memoria o ancho de banda para la aplicación?	NO	SI
Crecimiento del documento	¿Las piezas de información embebidas crecerían sin límites?	NO	SI
Carga de trabajo	¿Son las piezas de información escritas en tiempos diferentes en una carga de escritura pesada?	NO	SI
Individualidad	¿Para el hijo de la relación, puede existir sin el padre?	NO	SI

Si tenemos en cuenta los criterios de selección de un esquema de asociación anteriormente presentados claramente se ve una tendencia hacia modelar esta relación como una embebida. Además, si tenemos en cuenta los resultados del análisis de la carga de trabajo, vemos que esto no representaría un problema pues la carga de trabajo de la relación que se embebe, en este caso TiposHabitaciones, es muy baja y no atentaría contra el mantenimiento de la base de datos. Además, al tener una carga de trabajo similar la actualización del documento se facilitaría en el caso de una relación embebida. Por este motivo, se tomó la decisión de que TiposHabitaciones esté embebido en Habitaciones.

RELACIÓN: HABITACIONES (HIJO)-CONSUMOSERVICIOS (PADRE)

NOMBRE	PREGUNTA	EMBEBIDO	REFERENCIADO
Simplicidad	¿Mantener las piezas de información junta llevaría a un modelo y código más simple?	SI	NO
Van juntos	¿Las piezas de información tienen un 'tiene un', 'contiene' o una relación similar?	SI	NO

Atomicidad de la consulta	¿La aplicación consulta las piezas de información juntas?	SI	NO
Complejidad de actualización	¿Son las piezas de información actualizadas juntas?	SI	NO
Archivo	¿Deberían archivarse los datos al mismo tiempo?	SI	NO
Cardinalidad	¿Hay una alta cardinalidad (actual o creciente) por parte del hijo de la relación?	NO	SI
Duplicidad de datos	¿La duplicación de los datos sería muy difícil de manejar?	NO	SI
Tamaño del documento	¿La combinación del tamaño de la información tomaría mucho espacio en memoria o ancho de banda para la aplicación?	NO	SI
Crecimiento del documento	¿La pieza de información embebida crecerían sin límites?	NO	SI
Carga de trabajo	¿Son las piezas de información escritas en tiempos diferentes en una carga de escritura pesada?	NO	SI
Individualidad	¿Para el hijo de la relación, puede existir sin el padre?	NO	SI

Si tenemos en cuenta los criterios de selección de un esquema de asociación anteriormente presentados podemos ver que la relación que se está analizando debería ser modelada como una referenciación. Adicionalmente, si tenemos en cuenta que la carga de trabajo de ConsumoServicios es muy alta la constante creación y modificación de esta colección con Habitaciones embebido en ella sería muy costoso. Por lo anteriormente expuesto se tomó la decisión de que la relación entre ConsumoServicios y Habitaciones sea una relación de referenciación.

RELACIÓN: HABITACIONES (HIJO)-CLIENTES (PADRE)

NOMBRE	PREGUNTA	EMBEBIDO	REFERENCIADO
Simplicidad	¿Mantener las piezas de información junta llevaría a un modelo y código más simple?	SI	NO

Van juntos	¿Las piezas de información tienen un 'tiene un', 'contiene' o una relación similar?	SI	NO
Atomicidad de la consulta	¿La aplicación consulta las piezas de información juntas?	SI	NO
Complejidad de actualización	¿Son las piezas de información actualizadas juntas?	SI	NO
Archivo	¿Deberían archivarse los datos al mismo tiempo?	SI	NO
Cardinalidad	¿Hay una alta cardinalidad (actual o creciente) por parte del hijo de la relación?	NO	SI
Duplicidad de datos	¿La duplicación de los datos sería muy difícil de manejar?	NO	SI
Tamaño del documento	¿La combinación del tamaño de la información tomaría mucho espacio en memoria o ancho de banda para la aplicación?	NO	SI
Crecimiento del documento	¿La pieza de información embebida crecerían sin límites?	NO	SI
Carga de trabajo	¿Son las piezas de información escritas en tiempos diferentes en una carga de escritura pesada?	NO	SI
Individualidad	¿Para el hijo de la relación, puede existir sin el padre?	NO	SI

Aunque los criterios de selección de un esquema de asociación tienen motivos por los que no sería adecuado hacer un sistema embebido, existe un mayor número de razones para embeber esta relación que para referenciarla. Si adicionalmente tomamos en cuenta que el resultado del análisis de la carga de trabajo indicó que habitaciones no se crean, consultan o modifican habitualmente esto no representaría un problema para el mantenimiento de la base de datos. Por lo expuesto anteriormente se toma la decisión de que la relación entre Habitaciones y Clientes sea una embebida.

RELACIÓN: CLIENTES (PADRE)- RESERVASALOJAMIENTOS (HIJO)

NOMBRE	PREGUNTA	EMBEBIDO	REFERENCIADO
Simplicidad	¿Mantener las piezas de información junta llevaría a un modelo y código más simple?	SI	NO
Van juntos	¿Las piezas de información tienen un 'tiene un', 'contiene' o una relación similar?	SI	NO
Atomicidad de la consulta	¿La aplicación consulta las piezas de información juntas?	SI	NO
Complejidad de actualización	¿Son las piezas de información actualizadas juntas?	SI	NO
Archivo	¿Deberían archivarse los datos al mismo tiempo?	SI	NO
Cardinalidad	¿Hay una alta cardinalidad (actual o creciente) por parte del hijo de la relación?	NO	SI
Duplicidad de datos	¿La duplicación de los datos sería muy difícil de manejar?	NO	SI
Tamaño del documento	¿La combinación del tamaño de la información tomaría mucho espacio en memoria o ancho de banda para la aplicación?	NO	SI
Crecimiento del documento	¿La pieza de información embebida crecerían sin límites?	NO	SI
Carga de trabajo	¿Son las piezas de información escritas en tiempos diferentes en una carga de escritura pesada?	NO	SI
Individualidad	¿Para el hijo de la relación, puede existir sin el padre?	NO	SI

Aunque bajo los criterios de selección de un esquema de asociación existen criterios para modelar esta relación como una embebida, la mayoría de los criterios apuntan a que la relación debe ser una referenciada. Esto es respaldado por los resultados del análisis de la carga de trabajo, los cuales indican que la carga de trabajo es muy alta para manejarla en un solo documento. Por los motivos

expuestos anteriormente se decidió que la relación entre ReservasAlojamiento y Clientes va a ser una referenciada.

d)

DESCRIPCIÓN GRÁFICA JSON ESQUEMA DE ASOACIACIÓN

Ejemplo de datos json relación ConsumosServicios-Servicios (embebido):


```
{
    "idConsumo": 1,
    "fecha": "2020-10-10",
    "costo": 10000,
    "descripcionConsumo": "Consumo de bebida en el bar",
    "servicio":
        {
            "nombre": "Bar",
            "descripcionServicio": "Servicio de bar estilo mexicano",
            "menu": "Fajitas enchiladas, margarilas, tequila, cerveza",
            "horario": "2:00 pm - 1:00 am",
            "capacidad": 100
        }
}
```

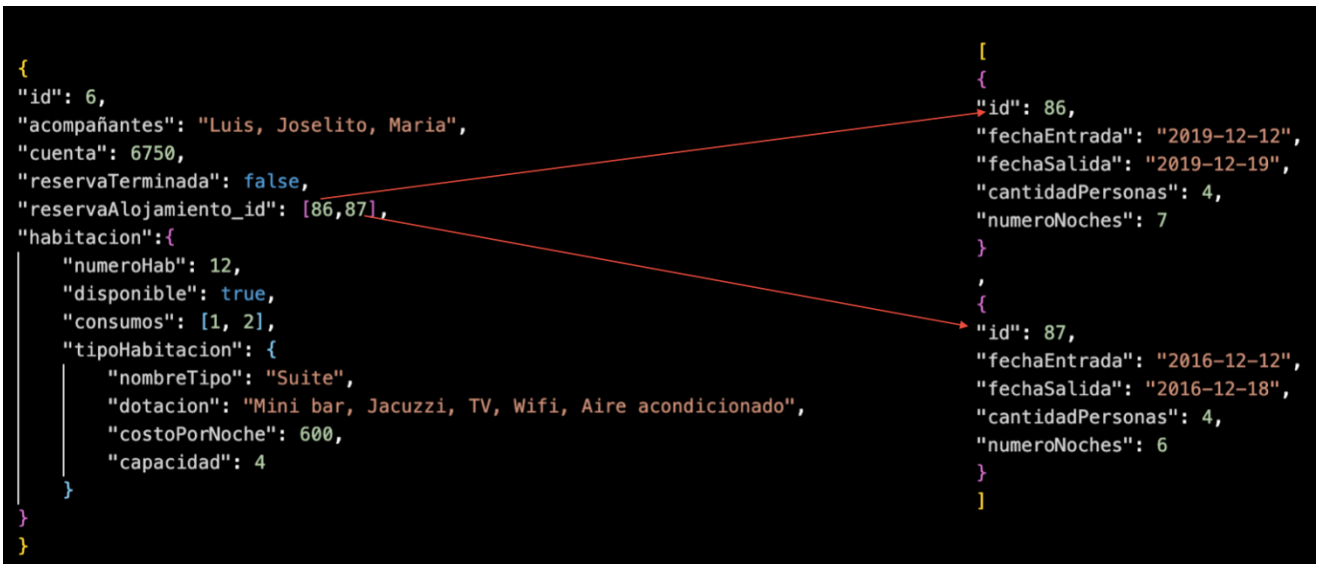

Ejemplo de datos json relación Habitaciones-TiposHabitaciones (embebido):


```
{
    "numeroHab": 12,
    "disponible": true,
    "consumos": [1, 2],
    "tipoHabitacion": {
        "nombreTipo": "Suite",
        "dotacion": "Mini bar, Jacuzzi, TV, Wifi, Aire acondicionado",
        "costoPorNoche": 600,
        "capacidad": 4
    }
}
```

Ejemplo de datos json relación Clientes- Habitaciones (embebido):


```
{
"id": 6,
"acompañantes": "Luis, Joselito, Maria",
"cuenta": 6750,
"reservaTerminada": false,
"reservaAlojamiento_id": [86,87],
"habitacion":{
    "numeroHab": 12,
    "disponible": true,
    "consumos": [1, 2],
    "tipoHabitacion": {
        "nombreTipo": "Suite",
        "dotacion": "Mini bar, Jacuzzi, TV, Wifi, Aire acondicionado",
        "costoPorNoche": 600,
        "capacidad": 4
    }
}
}
```

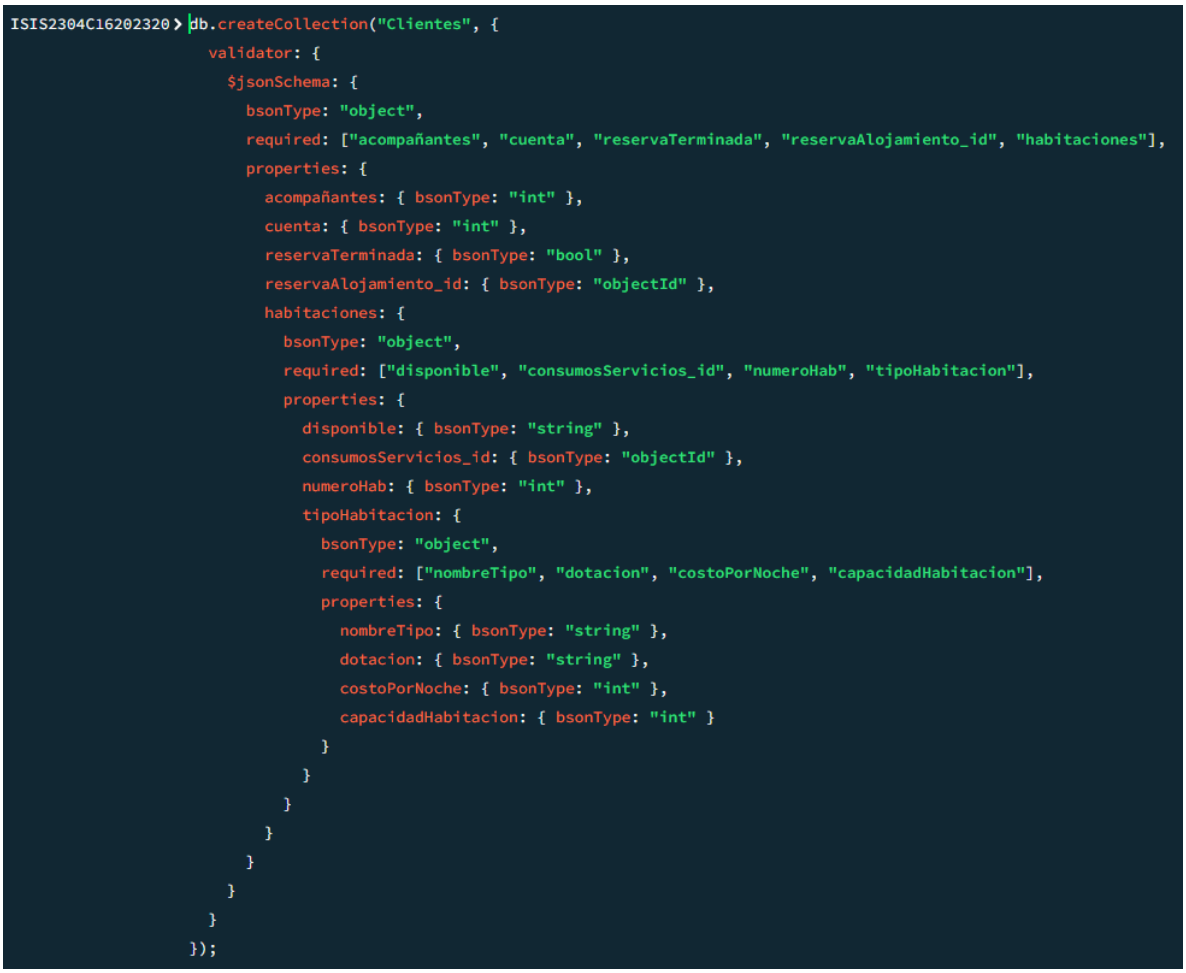

Ejemplo de datos json relación Clientes- ReservasAlojamiento (referenciado):



Ejemplo de datos json relación Habitaciones- ConsumosServicios (referenciado):

Creación de esquemas:

Clientes, habitaciones y tipoHabitacion




```
db.createCollection("Clientes", {
 validator: {
  $jsonSchema: {
   bsonType: "object",
   required: ["acompañantes", "cuenta", "reservaTerminada"],
   properties: {
    acompañantes: { bsonType: "int" },
    cuenta: { bsonType: "int" },
    reservaTerminada: { bsonType: "bool" },
    reservaAlojamiento_id: {
     bsonType: ["array", "objectId", "null"],
```

```
        items: { bsonType: "objectId" }
      },
      habitaciones: {
        bsonType: ["object", "null"],
        required: ["disponible", "numeroHab", "tipoHabitacion"],
        properties: {
          disponible: { bsonType: "string" },
          consumosServicios_id: { bsonType: ["array", "objectId", "null"] },
          numeroHab: { bsonType: "int" },
          tipoHabitacion: {
            bsonType: "object",
            required: ["nombreTipo", "dotacion", "costoPorNoche", "capacidadHabitacion"],
            properties: {
              nombreTipo: { bsonType: "string" },
              dotacion: { bsonType: "string" },
              costoPorNoche: { bsonType: "int" },
              capacidadHabitacion: { bsonType: "int" }
            }
          }
        }
      }
    }
  }
});
```

ConsumosServicio y Servicio:

```
ISIS2304C16202320 > db.createCollection("ConsumosServicio", {
                      validator: {
                        $jsonSchema: {
                          bsonType: "object",
                          required: ["fecha", "costo", "descripcionConsumo", "servicios"],
                          properties: {
                            fecha: { bsonType: "date" },
                            costo: { bsonType: "int" },
                            descripcionConsumo: { bsonType: "string" },
                            servicios: {
                              bsonType: "object",
                              required: ["nombre", "descripcionServicio", "menu", "horario", "capacidadServicio"],
                              properties: {
                                nombre: { bsonType: "string" },
                                descripcionServicio: { bsonType: "string" },
                                menu: { bsonType: "array" },
                                horario: { bsonType: "string" },
                                capacidadServicio: { bsonType: "int" }
                              }
                            }
                          }
                        }
                      }
                    });
```


```
db.createCollection("ConsumosServicio", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["fecha", "costo", "descripcionConsumo", "servicios"],
      properties: {
        fecha: { bsonType: "date" },
        costo: { bsonType: "int" },
        descripcionConsumo: { bsonType: "string" },
        servicios: {
          bsonType: "object",
          required: ["nombre", "descripcionServicio", "menu", "horario", "capacidadServicio"],
          properties: {
```

```
        nombre: { bsonType: "string" },
        descripcionServicio: { bsonType: "string" },
        menu: { bsonType: "array" },
        horario: { bsonType: "string" },
        capacidadServicio: { bsonType: "int" }
      }
    }
  }
 }
});
```

ReservasAlojamientos:

```
ISIS2304C16202320 > db.createCollection("ReservasAlojamientos", {
                      validator: {
                        $jsonSchema: {
                          bsonType: "object",
                          required: ["fechaEntrada", "fechaSalida", "cantidadPersonas", "numeroNoches"],
                          properties: {
                            fechaEntrada: { bsonType: "date" },
                            fechaSalida: { bsonType: "date" },
                            cantidadPersonas: { bsonType: "int" },
                            numeroNoches: { bsonType: "int" }
                          }
                        }
                      });
```

```
db.createCollection("ReservasAlojamientos", {
 validator: {
   $jsonSchema: {
     bsonType: "object",
     required: ["fechaEntrada", "fechaSalida", "cantidadPersonas", "numeroNoches"],
```

```
    properties: {
      fechaEntrada: { bsonType: "date" },
      fechaSalida: { bsonType: "date" },
      cantidadPersonas: { bsonType: "int" },
      numeroNoches: { bsonType: "int" }
    }
   }
  }
});
```

REQUERIMIENTOS FUNCIONALES:

RF1 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR TIPO DE HABITACIÓN CRUD de los tipos de habitaciones de los cuales dispone el hotel con su dotación correspondiente. Considere inicialmente por lo menos 3 tipos de habitaciones.

Registrar:

```
db.TipoHabitacion.insertOne({
 nombreTipo: "",
 dotacion: "",
 costoPorNoche: ,
 capacidadHabitacion:
});
```

Actualizar:

```
db.TipoHabitacion.updateOne(
  { nombreTipo: "nombre_a_actualizar" },
  {
    $set: {
      nombreTipo: "nuevo_nombre",
```

```
        dotacion: "nueva_dotacion",
        costoPorNoche: nuevo_costo,
        capacidadHabitacion: nueva_capacidad
      }
    }
);
```

Borrar:

```
db.TipoHabitacion.deleteOne({ nombreTipo: "" });
```

Consultar:

```
db.TipoHabitacion.findOne({ nombreTipo: "" });
```

EJEMPLOS DE INSERCION:

```
db.TipoHabitacion.insertOne({
  nombreTipo: "Suite Deluxe",
  dotacion: "Jacuzzi, TV de pantalla plana, Minibar",
  costoPorNoche: 300,
  capacidadHabitacion: 2
});

db.TipoHabitacion.insertOne({
  nombreTipo: "Habitación Familiar",
  dotacion: "Cama king-size, Sofá cama, Cocina pequeña",
  costoPorNoche: 250,
  capacidadHabitacion: 4
});

db.TipoHabitacion.insertOne({
```

```
  nombreTipo: "Habitación Económica",
  dotacion: "Cama individual, Baño compartido",
  costoPorNoche: 80,
  capacidadHabitacion: 1
});
```

RF2 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR HABITACIÓN CRUD de las instancias de las habitaciones de las cuales dispone el hotel. Considere inicialmente por lo menos 3 habitaciones de cada tipo de habitación disponible

Registrar:

```
db.Habitaciones.insertOne({
  disponible: "",
  consumosServicios_id: ObjectId(""),
  numeroHab:
});
```

Actualizar:

```
db.Habitaciones.updateOne(
  { numeroHab: "numero_a_actualizar" },
  {
    $set: {
      disponible: "nuevo_estado",
      consumosServicios_id: ObjectId("nuevo_objeto_id"),
      numeroHab: "nuevo_numero"
    }
  }
);
```

Borrar:

db.Habitaciones.deleteOne({ numeroHab: });

Consultar:

db.Habitaciones.findOne({ numeroHab: });

Este es el CRUD para el requerimiento, para asignar 3 habitaciones por cada tipo de habitación registrada

Primero creamos las 9 habitaciones:

```
db.Habitaciones.insertMany([
 {
  disponible: "Sí",
  consumosServicios_id: ObjectId("987654321098765432109876"),
  numeroHab: 101
 },
 {
  disponible: "No",
  consumosServicios_id: ObjectId("876543210987654321098765"),
  numeroHab: 102
 },
 {
  disponible: "Sí",
  consumosServicios_id: ObjectId("765432109876543210987654"),
  numeroHab: 103
 },
 {
  disponible: "Sí",
```

```
    consumosServicios_id: ObjectId("654321098765432109876543"),
    numeroHab: 201
  },
  {
    disponible: "No",
    consumosServicios_id: ObjectId("543210987654321098765432"),
    numeroHab: 202
  },
  {
    disponible: "Sí",
    consumosServicios_id: ObjectId("432109876543210987654321"),
    numeroHab: 203
  },
  {
    disponible: "Sí",
    consumosServicios_id: ObjectId("321098765432109876543210"),
    numeroHab: 301
  },
  {
    disponible: "No",
    consumosServicios_id: ObjectId("210987654321098765432109"),
    numeroHab: 302
  },
  {
    disponible: "Sí",
    consumosServicios_id: ObjectId("109876543210987654321098"),
    numeroHab: 303
  }
]);
```

Luego le asignamos al tipo de habitación que ya fueron creados:

```
db.Habitaciones.updateOne({ numeroHab: 101 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Suite Deluxe" }) } });

db.Habitaciones.updateOne({ numeroHab: 102 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Suite Deluxe" }) } });

db.Habitaciones.updateOne({ numeroHab: 103 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Suite Deluxe" }) } });


db.Habitaciones.updateOne({ numeroHab: 201 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Habitación Familiar" }) } });

db.Habitaciones.updateOne({ numeroHab: 202 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Habitación Familiar" }) } });

db.Habitaciones.updateOne({ numeroHab: 203 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Habitación Familiar" }) } });


db.Habitaciones.updateOne({ numeroHab: 301 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Habitación Económica" }) } });

db.Habitaciones.updateOne({ numeroHab: 302 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Habitación Económica" }) } });

db.Habitaciones.updateOne({ numeroHab: 303 }, { $set: { tipoHabitacion:
db.TipoHabitacion.findOne({ nombreTipo: "Habitación Económica" }) } });
```

RF3 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR UN SERVICIO DEL HOTEL CRUD de la descripción de los servicios de los cuales dispone el hotel. Considere inicialmente por lo menos 1 servicio de cada uno de los tipos de servicio descritos en el enunciado (piscina, spa, bar, … El menú de los bares y restaurantes debe tener por lo menos tres productos).

Registrar:

```
db.Servicios.insertOne({
  nombreServicio: "",
  descripcionServicio: "",
  menu: ["", "", ""],
```

```
  horario: "",
  capacidadServicio:,
  profundidad:,
  costo:
});
```

Actualizar:

Borrar:

```
db.Servicios.deleteOne({ nombreServicio: "" });
```

Consultar:

```
db.Servicios.findOne({ nombreServicio: "" });
```

Ejemplo de datos:

```
// Piscina
db.Servicios.insertOne({
  nombreServicio: "Piscina",
  descripcionServicio: "Área de piscina para relajación",
  menu: ["Toallas", "Bebidas refrescantes", "Snacks"],
  horario: "9:00 AM - 8:00 PM",
  capacidadServicio: 50,
  profundidad: 1.5,
  costo: 200000
});

// Gimnasio
db.Servicios.insertOne({
```

```
  nombreServicio: "Gimnasio",
  descripcionServicio: "Instalaciones de fitness",
  menu: ["Toallas", "Agua embotellada", "Equipos de entrenamiento"],
  horario: "6:00 AM - 10:00 PM",
  capacidadServicio: 30,
  maquinas: ["Caminadoras", "Pesas", "Bicicletas"],
  costo: 100000
});

// Internet
db.Servicios.insertOne({
  nombreServicio: "Internet",
  descripcionServicio: "Conexión Wi-Fi de alta velocidad",
  menu: ["Código de acceso", "Asistencia técnica", "Conexión segura"],
  horario: "24/7",
  capacidadServicio: 100,
  costo: 50000
});

// Bar
db.Servicios.insertOne({
  nombreServicio: "Bar",
  descripcionServicio: "Ambiente agradable con música en vivo",
  menu: ["Cócteles especiales", "Aperitivos", "Cervezas locales"],
  horario: "6:00 PM - 1:00 AM",
  capacidadServicio: 40,
  estiloMusical: "Jazz",
  costo: 150000
});
```

```
// Restaurante
db.Servicios.insertOne({
  nombreServicio: "Restaurante",
  descripcionServicio: "Oferta gastronómica internacional",
  menu: ["Entradas gourmet", "Platos principales variados", "Postres exquisitos"],
  horario: "12:00 PM - 10:00 PM",
  capacidadServicio: 60,
  estiloGastronomico: "Internacional",
  costo: 250000
});

// Supermercado
db.Servicios.insertOne({
  nombreServicio: "Supermercado",
  descripcionServicio: "Oferta de productos de consumo diario",
  menu: ["Frutas frescas", "Productos de limpieza", "Artículos de cuidado personal"],
  horario: "8:00 AM - 8:00 PM",
  capacidadServicio: 50,
  costo: 0
});

// Tiendas
db.Servicios.insertOne({
  nombreServicio: "Tiendas",
  descripcionServicio: "Productos exclusivos de joyería y moda",
  menu: ["Joyas únicas", "Ropa de diseñadores", "Accesorios de moda"],
  horario: "10:00 AM - 6:00 PM",
  capacidadServicio: 10,
```

```
  costo: 0
});

// SPA
db.Servicios.insertOne({
  nombreServicio: "SPA",
  descripcionServicio: "Experiencias relajantes y terapéuticas",
  menu: ["Masaje relajante", "Tratamientos faciales", "Baños aromáticos"],
  horario: "9:00 AM - 8:00 PM",
  capacidadServicio: 20,
  duracion: 60,
  costo: 300000
});

// Lavado/Planchado/Embolada
db.Servicios.insertOne({
  nombreServicio: "Lavado/Planchado/Embolada",
  descripcionServicio: "Servicios de lavandería y cuidado de prendas",
  menu: ["Lavado y planchado de prendas", "Servicio de embolada de calzado", "Entrega rápida"],
  horario: "8:00 AM - 6:00 PM",
  capacidadServicio: 15,
  costo: 80000
});

// Préstamo de utensilios
db.Servicios.insertOne({
  nombreServicio: "Préstamo de utensilios",
  descripcionServicio: "Disponibilidad de utensilios para el cliente",
  menu: ["Toallas", "Almohadas adicionales", "Plancha"],
```

```
  horario: "24/7",
  capacidadServicio: 5,
  costo: 0
});

// Salones de conferencias
db.Servicios.insertOne({
  nombreServicio: "Salones de conferencias",
  descripcionServicio: "Espacios para eventos de gran capacidad",
  menu: ["Proyector y pantalla", "Servicio de café", "Asistencia técnica"],
  horario: "Según reserva",
  capacidadServicio: 500,
  costo: 150000
});

// Salones de reuniones
db.Servicios.insertOne({
  nombreServicio: "Salones de reuniones",
  descripcionServicio: "Espacios para reuniones pequeñas",
  menu: ["Pizarra y marcadores", "Servicio de café", "Conexión Wi-Fi"],
  horario: "Según reserva",
  capacidadServicio: 12,
  costo: 75000
});
```

RF4 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR UNA RESERVA DE ALOJAMIENTO CRUD de la reservación de una habitación. Una reservación, por un período de tiempo, por parte de un cliente, se puede hacer siempre y cuando esté disponible.

Registrar:

```
db.ReservasAlojamientos.insertOne({
  fechaEntrada: new Date(""), // Fecha de entrada en formato Date
  fechaSalida: new Date(""),  // Fecha de salida en formato Date
  cantidadPersonas: ,
  numeroNoches:
});
```

Actualizar:

```
db.ReservasAlojamientos.updateOne(
  { _id: ObjectId("id_del_documento_a_actualizar") },
  {
    $set: {
      fechaEntrada: new Date("nueva_fecha_entrada"),
      fechaSalida: new Date("nueva_fecha_salida"),
      cantidadPersonas: nuevo_valor_personas,
      numeroNoches: nuevo_numero_noches
    }
  }
);
```

Borrar:

```
db.ReservasAlojamientos.deleteOne({ _id:  });
```

Consultar:

```
db.ReservasAlojamientos.findOne({ _id: ObjectId("") });
```

Ejemplos de dato:

```
db.ReservasAlojamientos.insertOne({
  fechaEntrada: new Date("2023-12-01"), // Fecha de entrada en formato Date
  fechaSalida: new Date("2023-12-10"),  // Fecha de salida en formato Date
  cantidadPersonas: 2,
  numeroNoches: 9
});
```

RF5 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR LA LLEGADA DE UN CLIENTE AL HOTEL
CRUD de la llegada de un cliente al hotel. Al registrar la llegada, se debe tener en cuenta que está correspondiente a una reserva ya registrada.

Registrar:

```
db.Clientes.insertOne({
  acompañantes: ,
  cuenta:,
  reservaTerminada:,
  reservaAlojamiento_id: ObjectId(“”),
  habitaciones:
});
```

Actualizar:

```
db.Clientes.updateOne(
  { _id: ObjectId("id_del_documento_a_actualizar") },
  {
    $set: {
      acompañantes: nuevo_valor_acompañantes,
      cuenta: nuevo_valor_cuenta,
      reservaTerminada: nuevo_valor_terminada,
      reservaAlojamiento_id: ObjectId("nuevo_id_reserva"),
      habitaciones: nuevo_valor_habitaciones
    }
  }
);
```

Borrar:

```
db.Clientes.deleteOne({ _id: ObjectId("") });
```

Consultar:

```
db.Clientes.findOne({ _id: ObjectId("") });
```

RF6 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR UN CONSUMO DE UN SERVICIO DEL HOTEL POR PARTE DE UN CLIENTE O SUS ACOMPAÑANTES CRUD del registro un consumo de un servicio por parte de un cliente o sus acompañantes. Cada consumo registrado está asociado a una habitación y tiene descripción y costo (que ya están establecidos al escoger el consumo a registrar) y una fecha (que es escogida por el usuario).

Registrar:

```
db.ConsumosServicio.insertOne({
  _id: ObjectId(""),
  fecha: new Date("2023-12-01"),
  costo:,
  descripcionConsumo: ""
});
```

Actualizar:

```
db.ConsumosServicio.updateOne(
  { _id: ObjectId("id_del_documento_a_actualizar") },
  {
    $set: {
      fecha: new Date("nueva_fecha"),
      costo: nuevo_valor_costo,
      descripcionConsumo: "nueva_descripcion"
    }
  }
);
```

Borrar:

```
db.ConsumosServicio.deleteOne({ _id: ObjectId("") });
```

Consultar:

```
db.ConsumosServicio.findOne({ _id: ObjectId("") });
```

Dato de ejemplo:

```
db.ConsumosServicio.insertOne({
```

```
  _id: ObjectId("987654321098765432109876"),
  fecha: new Date("2023-12-01"),
  costo: 150,
  descripcionConsumo: "Ejemplo de consumo de servicio",
  servicios: {
    nombre: "Servicio Ejemplo",
    descripcionServicio: "Descripción del servicio",
    menu: ["Plato 1", "Plato 2"],
    horario: "Horario del servicio",
    capacidadServicio: 10
  }
});
```

RF7 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR LA SALIDA DE UN CLIENTE CRUD del registro de la salida de un cliente al hotel, con todo lo que eso implica

Registro: Por la forma en la que tenemos el modelo, se registra la salida de un cliente cuando su atributo reservaTerminada pasa a ser true, por ende:

```
db.Clientes.updateOne(
  { _id: ObjectId("") },
  { $set: { reservaTerminada: true } }
);
```

Actualizar: Actualización se cumple con la sentencia anterior

Borrar: Para borrar la salida de un cliente tomaremos la eliminación total de ese cliente

```
db.Clientes.deleteOne({ _id: ObjectId("") });
```

Consultar: Para consultar la salida de un cliente se tendría que buscar por un ID especifico y verificar que el campo de reservaTerminada sea True, podriamos consultar cuales son los clientes que ya acabaron su reserva filtrando

db.Clientes.find({ reservaTerminada: true }).pretty();

REQUERIMIENTOS FUNCIONALES DE CONSULTA BÁSICOS:

RFC1 - MOSTRAR EL DINERO RECOLECTADO POR SERVICIOS EN CADA HABITACIÓN EN EL ÚLTIMO AÑO CORRIDO.

```
db.Habitaciones.aggregate([
 {
  $lookup: {
   from: "ConsumosServicio",
   localField: "consumosServicios_id",
   foreignField: "_id",
   as: "consumosServicioInfo"
  }
 },
 {
  $unwind: {
   path: "$consumosServicioInfo",
   preserveNullAndEmptyArrays: true
  }
 },
 {
  $project: {
```

```
    numeroHab: 1,
    costoConsumoServicio: {
     $cond: {
       if: {
         $and: [
           { $gte: ["$consumosServicioInfo.fecha", ISODate("2023-01-01")] },
           { $lt: ["$consumosServicioInfo.fecha", ISODate("2024-01-01")] }
         ]
       },
       then: "$consumosServicioInfo.costo",
       else: 0
     }
    }
   }
  },
  {
   $group: {
    _id: "$numeroHab",
    totalCostos: { $sum: "$costoConsumoServicio" }
   }
  }
]);
```

Datos para la prueba:

```
db.ConsumosServicio.insertMany([
 {
  _id: ObjectId("987654321098765432109876"),
  fecha: new Date("2023-12-02"),
  costo: 75,
```

```
    descripcionConsumo: "Nuevo Consumo 1",
    servicios: {
      nombre: "Servicio 1",
      descripcionServicio: "Descripción del Servicio 1",
      menu: ["Plato A", "Plato B", "Bebida"],
      horario: "12:00 PM - 10:00 PM",
      capacidadServicio: 15
    }
  },
  {
    _id: ObjectId("656b93f2fbfb9ac4119bed78"),
    fecha: new Date("2023-12-02"),
    costo: 100,
    descripcionConsumo: "Nuevo Consumo 2",
    servicios: {
      nombre: "Servicio 2",
      descripcionServicio: "Descripción del Servicio 2",
      menu: ["Plato C", "Plato D", "Bebida"],
      horario: "2:00 PM - 11:00 PM",
      capacidadServicio: 20
    }
  }
]);

db.Habitaciones.insertOne({
  disponible: "Sí",
  consumosServicios_id: [ObjectId("987654321098765432109876"), ObjectId()],
  numeroHab: 101
});
```

RFC2 - MOSTRAR EL ÍNDICE DE OCUPACIÓN DE CADA UNA DE LAS HABITACIONES DEL HOTEL EN EL ÚLTIMO AÑO CORRIDO Se debe mostrar el % de ocupación de cada habitación en el último año

```
db.Clientes.aggregate([
 {
  $lookup: {
   from: "ReservasAlojamientos",
   localField: "reservaAlojamiento_id",
   foreignField: "_id",
   as: "reservaAlojamientoInfo"
  }
 },
 {
  $project: {
   _id: 1,
   numeroHab: "$habitaciones.numeroHab",
   numeroNoches: { $arrayElemAt: ["$reservaAlojamientoInfo.numeroNoches", 0] },
   porcentajeOcupacion: {
    $cond: {
     if: { $eq: [{ $year: { $arrayElemAt: ["$reservaAlojamientoInfo.fechaEntrada", 0] } }, 2023] },
     then: {
      $multiply: [
       {
        $divide: [
         { $arrayElemAt: ["$reservaAlojamientoInfo.numeroNoches", 0] },
         365 // Suponiendo 365 noches en el año
        ]
       },
```

```
            100
          ]
        },
        else: 0 // Si no es el año 2023, el porcentaje de ocupación es 0
      }
    }
  }
 }
]).pretty();
```

Datos Prueba:

```
db.Clientes.insertOne({
 _id: ObjectId("656ba1d4fbfb9ac4119bed7b"),
 acompañantes: 2,
 cuenta: 500,
 reservaTerminada: false,
 reservaAlojamiento_id: ObjectId("656b72cbfbfb9ac4119bed5c"),
 habitaciones: {
  disponible: "No",
  consumosServicios_id: ObjectId("987654321098765432109876"),
  numeroHab: 505,
  tipoHabitacion: {
   nombreTipo: "ejemploReq",
   dotacion: "TV, minibar, wifi",
   costoPorNoche: 150,
   capacidadHabitacion: 2
  }
 }
});
```

```
db.ReservasAlojamientos.insertOne({
  _id: ObjectId("656b72cbfbfb9ac4119bed5c"),
  fechaEntrada: new Date("2023-12-01"),
  fechaSalida: new Date("2023-12-05"),
  cantidadPersonas: 2,
  numeroNoches: 4
});

// Inserción en Clientes
db.Clientes.insertOne({
  _id: ObjectId("656ba1d4fbfb9ac4119bed7c"),
  acompañantes: 3,
  cuenta: 700,
  reservaTerminada: true,
  reservaAlojamiento_id: ObjectId("656b72cbfbfb9ac4119bed5d"),
  habitaciones: {
    disponible: "No",
    consumosServicios_id: ObjectId("876543210987654321098765"),
    numeroHab: 510,
    tipoHabitacion: {
      nombreTipo: "suiteDeluxe",
      dotacion: "TV, minibar, jacuzzi",
      costoPorNoche: 200,
      capacidadHabitacion: 3
    }
  }
});

// Inserción en ReservasAlojamientos
```

```
db.ReservasAlojamientos.insertOne({
  _id: ObjectId("656b72cbfbfb9ac4119bed5d"),
  fechaEntrada: new Date("2023-11-20"),
  fechaSalida: new Date("2023-11-25"),
  cantidadPersonas: 3,
  numeroNoches: 5
});
```

RFC3 - MOSTRAR EL CONSUMO EN HOTELANDES POR UN CLIENTE, EN UN RANGO DE FECHAS INDICADO. Recuerde que un cliente puede alojarse en el hotel cuantas veces quiera.

```
db.Clientes.aggregate([
  {
    $match: {
      _id: ObjectId("656babb0fbfb9ac4119bed80")
    }
  },
  {
    $lookup: {
      from: "ConsumosServicio",
      localField: "habitaciones.consumosServicios_id",
      foreignField: "_id",
      as: "consumosServiciosInfo"
    }
  },
  {
    $unwind: "$consumosServiciosInfo" // Deshace el array creado por $lookup
  },
```

```
 {
  $match: {
   "consumosServiciosInfo.fecha": {
    $gte: ISODate("2023-01-01"), // Rango de inicio
    $lte: ISODate("2023-12-31")  // Rango de fin (ajústalo según necesites)
   }
  }
 },
 {
  $group: {
   _id: "$_id",
   totalCostoConsumo: { $sum: "$consumosServiciosInfo.costo" }
  }
 }
]);
```

Datos de ejemplo:

```
db.Clientes.insertOne({
 _id: ObjectId("656babb0fbfb9ac4119bed80"),
 acompañantes: 2,
 cuenta: 600,
 reservaTerminada: false,
 reservaAlojamiento_id: null,
 habitaciones: {
  disponible: "Sí",
  consumosServicios_id: [
   ObjectId("98765432109876543210987 6"),
```

```
    ObjectId("656babb0fbfb9ac4119bed7f")
   ],
   numeroHab: 506,
   tipoHabitacion: {
    nombreTipo: "Ejemplo",
    dotacion: "TV, minibar, wifi",
    costoPorNoche: 200,
    capacidadHabitacion: 2
   }
  }
});

db.ConsumosServicio.insertOne({
  _id: ObjectId("987654321098765432109876"),
  fecha: new Date("2023-07-15"),
  costo: 50,
  descripcionConsumo: "Consumo de servicios adicionales",
  servicios: {
   nombre: "Servicio Ejemplo",
   descripcionServicio: "Descripción del servicio",
  },
  menu: ["Opción 1", "Opción 2", "Opción 3"],
  horario: "10:00 AM - 8:00 PM",
  capacidadServicio: 10
});

db.ConsumosServicio.insertOne({
  _id: ObjectId("656babb0fbfb9ac4119bed7f"),
  fecha: new Date("2023-12-03"),
```

```
  costo: 80,
  descripcionConsumo: "Nuevo Consumo de Servicios",
  servicios: {
    nombre: "Nuevo Servicio",
    descripcionServicio: "Descripción del Nuevo Servicio",
  },
  menu: ["Plato 1", "Plato 2", "Plato 3"],
  horario: "12:00 PM - 10:00 PM",
  capacidadServicio: 15
});
```

REQUERIMIENTOS FUNCIONALES DE CONSULTA AVANZADOS

RFC7 - CONSULTAR LOS CLIENTES EXCELENTES Los clientes excelentes son de tres tipos: aquellos que realizan estancias (las estancias están delimitadas por un check in y su respectivo check out) en HotelAndes al menos una vez por trimestre. Esta consulta retorna toda la información de dichos clientes.

```
db.Clientes.aggregate([
  {
    $lookup: {
      from: "ReservasAlojamientos",
      localField: "reservaAlojamiento_id",
      foreignField: "_id",
      as: "reservasAlojamientos"
    }
  },
  {
    $unwind: "$reservasAlojamientos"
  },
```

```
{
  $group: {
    _id: "$_id",
    clienteInfo: {
      $first: {
        _id: "$_id",
        acompañantes: "$acompañantes",
        cuenta: "$cuenta",
        reservaTerminada: "$reservaTerminada",
        reservaAlojamiento_id: "$reservaAlojamiento_id",
        habitaciones: "$habitaciones"
      }
    },
    reservas: {
      $push: {
        fechaEntrada: "$reservasAlojamientos.fechaEntrada"
      }
    }
  }
},
{
  $project: {
    _id: 1,
    clienteInfo: 1,
    clienteExcelente: {
      $cond: {
        if: {
          $and: [
            { $isArray: "$reservas" },
```

```
            { $gte: [{ $size: "$reservas" }, 2] },
            {
              $lt: [
                { $arrayElemAt: ["$reservas.fechaEntrada", 0] },
                { $arrayElemAt: ["$reservas.fechaEntrada", 1] }
              ]
            },
            {
              $lte: [
                { $arrayElemAt: ["$reservas.fechaEntrada", 0] },
                { $add: [{ $arrayElemAt: ["$reservas.fechaEntrada", 1] }, 3 * 30 * 24 * 60 * 60 *
1000] }
              ]
            }
          ]
        },
        then: true,
        else: false
      }
    }
  },
  {
    $match: {
      clienteExcelente: true
    }
  }
]);
```

Datos de prueba:

```
db.Clientes.insertOne({
  "_id": ObjectId("656cb96b9bd1119c1462951d"),
  "acompañantes": 2,
  "cuenta": 100,
  "reservaTerminada": true,
  "reservaAlojamiento_id": [
    ObjectId("6162c2d2c698daab2209b800"),
    ObjectId("6162c2d2c698daab2209b801")
  ],
  "habitaciones": {
    "disponible": "Sí",
    "consumosServicios_id": [
      ObjectId("6162c2d2c698daab2209b802"),
      ObjectId("6162c2d2c698daab2209b803")
    ],
    "numeroHab": 888,
    "tipoHabitacion": {
      "nombreTipo": "ejemploReqAvanzado",
      "dotacion": "WiFi, TV, minibar",
      "costoPorNoche": 150,
      "capacidadHabitacion": 2
    }
  }
});

db.Clientes.insertOne({
  "_id": ObjectId("656cafd89bd1119c14629516"),
  "acompañantes": 2,
```

```
  "cuenta": 100,
  "reservaTerminada": true,
  "reservaAlojamiento_id": [
    ObjectId("6162bfc5c698daab2209b7f1"),
    ObjectId("6162bfc5c698daab2209b7f2")
  ],
  "habitaciones": {
    "disponible": "Sí",
    "consumosServicios_id": [
      ObjectId("6162bfc5c698daab2209b7f3"),
      ObjectId("6162bfc5c698daab2209b7f4")
    ],
    "numeroHab": 999,
    "tipoHabitacion": {
      "nombreTipo": "ejemploReqAvanzado",
      "dotacion": "WiFi, TV, minibar",
      "costoPorNoche": 150,
      "capacidadHabitacion": 2
    }
  }
});

db.ReservasAlojamientos.insertOne({
  "_id": ObjectId("6162c2d2c698daab2209b800"),
  "fechaEntrada": new Date("2023-01-15"),
  "fechaSalida": new Date("2023-01-20"),
  "cantidadPersonas": 2,
  "numeroNoches": 5
});
```

```
db.ReservasAlojamientos.insertOne({
  "_id": ObjectId("6162c2d2c698daab2209b801"),
  "fechaEntrada": new Date("2023-04-10"),
  "fechaSalida": new Date("2023-04-15"),
  "cantidadPersonas": 3,
  "numeroNoches": 5
});
```

```
db.ReservasAlojamientos.insertOne({ "_id": ObjectId("6162bfc5c698daab2209b7f1"), "fechaEntrada": new Date("2023-01-15"), "fechaSalida": new Date("2023-01-20"), "cantidadPersonas": 2, "numeroNoches": 5 }); db.ReservasAlojamientos.insertOne({ "_id": ObjectId("6162bfc5c698daab2209b7f2"), "fechaEntrada": new Date("2023-04-10"), "fechaSalida": new Date("2023-04-15"), "cantidadPersonas": 3, "numeroNoches": 5 });
```