ENTREGA PROYECTO 3 Sistemas Transaccionales

Samuel Ramirez

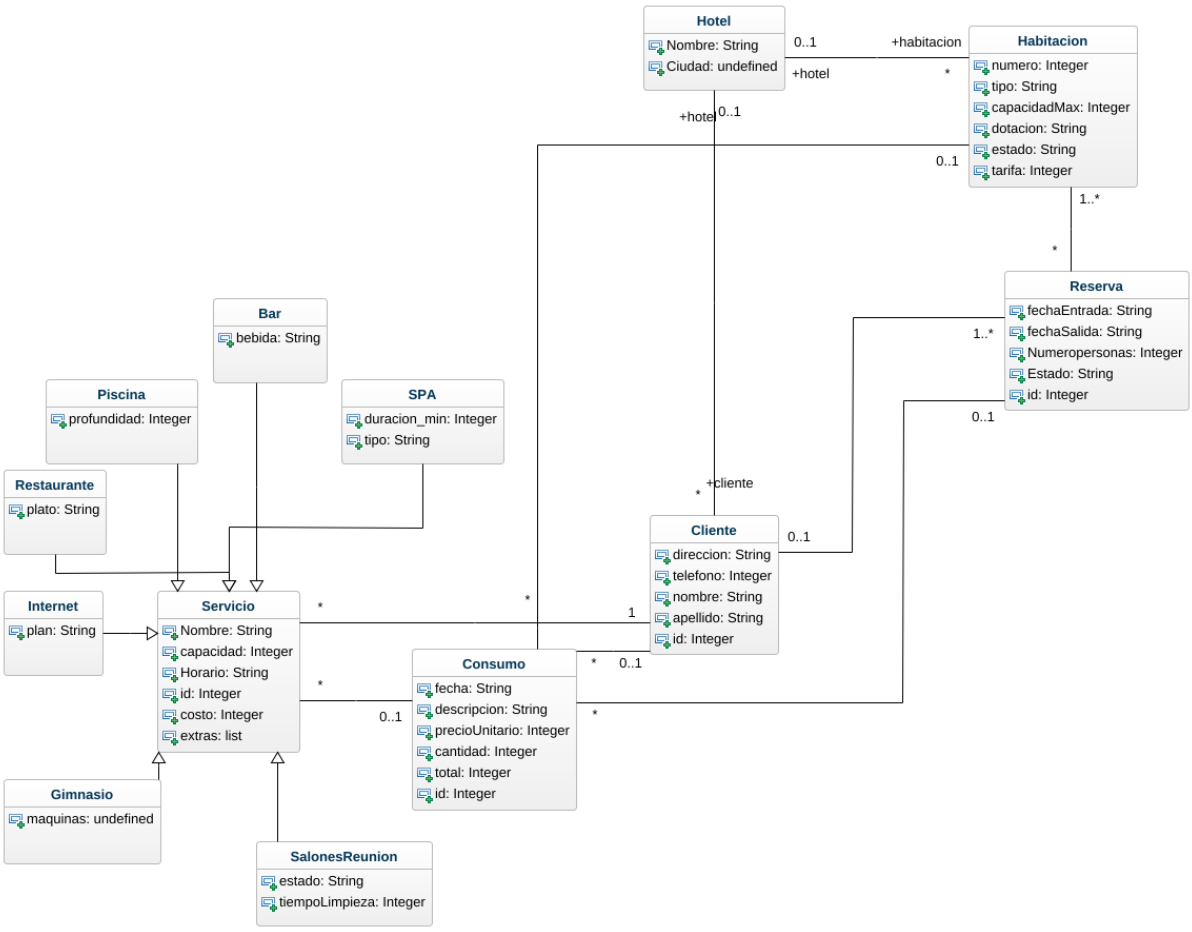
Andres Serrano

Juana Mejia

1. Revise el caso de estudio propuesto. Identifique los elementos fundamentales que hacen parte del negocio que se describe.

a. (7%) Análisis y modelo conceptual

a. Proponga un modelo conceptual en UML o E/R que describa las entidades del modelo de datos para la aplicación que se quiere desarrollar.



3. **(35%)** Diseño de la base de datos

1. (10%) Análisis de la carga de trabajo (workload). Para ello, presenten lo siguiente:

a. Identifiquen entidades y sus atributos:

Entidad	**Atributos**
Habitación	• *Numero:* Número identificador único para la habitación. • *Tipo:* Tipo de habitación (individual, doble, suite, etc.). • *CapacidadMax:* Capacidad máxima de personas permitidas en la habitación. • *Dotacion:* Equipamiento o características específicas de la habitación. • *Estado:* Estado actual de la habitación (ocupada, disponible, en limpieza, etc.). • *Tarifa:* Precio asociado al alquiler de la habitación.
Reserva	• *FechaEntrada***:** Fecha de inicio de la reserva. • *FechaSalida:* Fecha de finalización de la reserva. • *NumeroPersonas:* Número de personas para las que se realiza la reserva. • *Estado:* Estado actual de la reserva (confirmada, pendiente, cancelada, etc.). • *Id:* Identificador único de la reserva.
Cliente	• *Dirección:* Dirección del cliente. • *Telefono:* Número de teléfono del cliente. • *Nombre:* Nombre del cliente. • *Apellido:* Apellido del cliente. • *Id:* Identificador único del cliente.
Consumo	• *Fecha:* Fecha en la que se registró el consumo. • *Descripcion:* Descripción del consumo. • *Precio Unitario:* Precio por unidad del artículo consumido. • *Cantidad:* Cantidad de artículos consumidos. • *Total:* Total gastado en el consumo. • *Id:* Identificador único del consumo.
Servicio	• *Nombre:* Nombre del servicio. • *Capacidad:* Capacidad máxima de personas para el servicio. • *Horario:* Horario de disponibilidad del servicio. • *Id:* Identificador único del servicio. • *Costo:* Costo asociado al uso del servicio. • *Extras:* Información adicional sobre el servicio

b. Cuantifiquen las entidades (cantidad de registros que tendría la BD para cada una de las entidades, pueden encontrar un aproximado en el enunciado).

Entidad	**Cuantificación**
Habitación	200
Reserva	50000
Cliente	500
Consumo	250000
Servicio	35
Tipo habitacion	20

c. Analicen las operaciones de lectura y escritura para cada entidad. Para ello utilicen una tabla como la del ejemplo del anexo A. Recuerden que este análisis sirve para saber que información se accederá de manera conjunta.

Entidades	**Operaciones**	**Información Necesaria**	**Tipo de operación (read/write)**
Habitación	Consultar Tipo de Habitación	Tipo de habitación, Dotación	Read
Habitación	Registrar/Actualizar/Borrar Habitación	Numero, Tipo, Dotación, Estado, Tarifa	Write
Servicio	Consultar un Servicio del Hotel	Nombre, Capacidad, Horario, Costo, Extras	Read
Servicio	Registrar/Actualizar/Borrar Servicio del Hotel	Nombre, Capacidad, Horario, Costo, Extras	Write
Reserva	Consultar una Reserva de Alojamiento	FechaEntrada, FechaSalida, NumeroPersonas, Estado	Read
Reserva	Registrar/Actualizar/Borrar Reserva de Alojamiento	FechaEntrada, FechaSalida, NumeroPersonas, Estado	Write
Reserva	Consultar la Llegada de un Cliente al Hotel	Fecha, IdReserva	Read
Reserva	Registrar/Actualizar/Borrar Llegada de un Cliente	Fecha, IdReserva	Write

Consumo	Consultar un Consumo de un Servicio del Hotel	Fecha, Descripcion, Precio Unitario, Cantidad, Total	Read
Consumo	Registrar/Actualizar/Borrar Consumo de un Servicio del Hotel	Fecha, Descripcion, Precio Unitario, Cantidad, Total	Write
Cliente	Consultar la Salida de un Cliente	Fecha, IdReserva	Read
Cliente	Registrar/Actualizar/Borrar Salida de un Cliente	Fecha, IdReserva	Write

d. Cuantifiquen las operaciones de lectura y escritura para cada entidad. Para ello utilicen una tabla como la del ejemplo del anexo B.

Entidades	Operacion	Informacion Necesaria	Tipo	Rate
Habitación	Crear/Modificar Tipo de Habitación	Tipo de habitación, Dotación	Write	1/semana
Habitación	Crear/Modificar Habitación	Numero, Tipo, Dotación, Estado, Tarifa	Write	2/semana
Habitación	Consultar Tipo de Habitación	Tipo de habitación, Dotación	Read	1/semana
Habitación	Consultar Habitación	Numero, Tipo, Dotación, Estado, Tarifa	Read	2/semana
Servicio	Crear/Modificar Servicio del Hotel	Nombre, Capacidad, Horario, Costo, Extras	Write	2/semana
Servicio	Consultar Servicio del Hotel	Nombre, Capacidad, Horario, Costo, Extras	Read	2/semana
Reserva	Crear/Modificar Reserva de Alojamiento	FechaEntrada, FechaSalida, NumPersonas, Estado	Write	100/día
Reserva	Consultar Reserva de Alojamiento	FechaEntrada, FechaSalida, NumPersonas, Estado	Read	50/día
Consumo	Crear/Modificar Consumo de un	Fecha, Descripcion, Precio	Write	500/día

	Servicio del Hotel	Unitario, Cantidad, Total		
Consumo	Consultar Consumo de un Servicio del Hotel	Fecha, Descripcion, Precio Unitario, Cantidad, Total	Read	250/día
Cliente	Crear/Modificar Llegada o Salida de un Cliente	Fecha, IdReserva	Write	100/día
Cliente	Consultar Llegada o Salida de un Cliente	Fecha, IdReserva	Read	50/día

2. (15%) Describan las entidades de datos y las relaciones entre ellas que corresponden al modelo conceptual

UML propuesto. Para ello, presenten lo siguiente:

1. La lista de entidades con la descripción de cada una de ellas.

Entidad	**Descripcion**
Habitación	Representa las habitaciones existentes en el hotel. Cada habitación tiene un número único, un tipo específico (individual, doble, suite, etc.), una capacidad máxima de personas permitidas, una dotación que describe sus características, un estado actual (ocupada, disponibley una tarifa asociada.
Reserva	Se refiere a las reservas de habitaciones realizadas por los clientes. Contiene información como la fecha de entrada y salida, el número de personas para la reserva, el estado de la reserva y un identificador único (Id).
Cliente	Representa a los clientes del hotel. Incluye información como la dirección, teléfono, nombre, apellido e identificador único (Id) de cada cliente.
Consumo	Se refiere a los consumos realizados por los clientes durante su estancia en el hotel. Contiene detalles como la fecha del consumo, la descripción, el precio unitario, la cantidad consumida, el total gastado y un identificador único (Id).
Servicio	Representa los servicios ofrecidos por el hotel, como piscina, spa, bar, etc. Incluye información como el nombre del servicio, la capacidad, el horario de disponibilidad, un identificador único (Id), el costo y detalles adicionales sobre extras.

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

Entidades	**Relacion**
Habitacion+ Reserva	Muchos a muchos Una habitación puede tener muchas reservaciones y una reservación puede tener muchas habitaciones
Habitacion + Consumo	Uno a muchos Una habitación puede tener muchos consumos pero un consumo solo puede ser de 1 habitación
Reserva+ Cliente	Uno a muchos Un cliente puede tener muchas reservas pero una reserva solo puede tener 1 cliente
Cliente + Servicio	1 a muchos Un cliente puede tener muchos servicios pero un servicio solo es de 1 cliente
Cliente + Consumo	uno a muchos un cliente tiene muchos consumos per un consumo tiene solo 1 cliente
Consumo + Servicio	Uno a muchos Un servicio solo tiene 1 consumo asociado mientras el consumo puede tener varios servicios

c. El análisis de selección de esquema de asociación (referenciado o embebido) para cada relación entre entidades. Para ello use la tabla de análisis vista en clase, la cual se retoma en el anexo C, junto con los resultados del análisis de la carga de trabajo (workload), descrita antes.

Relacion: Habitacion + Reserva

Guideline Name	Question	Embed (yes/no)	Reference (opposite of embed)	Respuesta
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No	No
Go Together	Do the pieces of information have a “has.a” , “contains” or similar relationship?	Yes	No	Yes

Query Atomicity	Does the application query the pieces of information together?	Yes	No	Yes
Update Complexity	Are the pieces of information updated together?	Yes	No	Yes
Archival	Should the pieces of information be archived at the same time?	No	Yes	Yes
Cardinality	Is there a high cardinality current or growing in the child side of the relationship?	No	Yes	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes	No
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes	Yes
Document Growth	Would the embedded piece grow without bound ?	No	Yes	Yes
Workload	Are the pieces of information writte at different times in a write heavy workload?	No	Yes	No
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes	no

Se decide que esta relacion se hará referenciada .

Relacion: Habitacion+ Consumo

Guideline Name	Question	Embed (yes/no)	Reference (opposite of embed)	Respuesta
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No	No
Go Together	Do the pieces of information have a “has.a” , “contains” or similar relationship?	Yes	No	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No	Yes
Update Complexity	Are the pieces of information updated together?	Yes	No	No
Archival	Should the pieces of information be archived at the same time?	No	Yes	No

Cardinality	Is there a high cardinality current or growing in the child side of the relationship?	No	Yes	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes	No
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes	Yes
Document Growth	Would the embedded piece grow without bound ?	No	Yes	Yes
Workload	Are the pieces of information writte at different times in a write heavy workload?	No	Yes	Yes
Individuality	Fort he children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes	Yes

Para esta relación se decidió hacerla de forma referenciada debido a que estas son 2 colecciones que pueden existir individualmente

Relacion: Reserva + Cliente

Guideline Name	Question	Embed (yes/no)	Reference (opposite of embed)	Respuesta
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No	Yes
Go Together	Do the pieces of information have a “has.a” , “contains” or similar relationship?	Yes	No	Yes
Query Atomicity	Does the application query the pieces of information together?	Yes	No	Yes
Update Complexity	Are the pieces of information updated together?	Yes	No	Yes
Archival	Should the pieces of information be archived at the same time?	No	Yes	Yes
Cardinality	Is there a high cardinality current or growing in the child side of the relationship?	No	Yes	No
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes	No
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes	no

Document Growth	Would the embedded piece grow without bound ?	No	Yes	No
Workload	Are the pieces of information writte at different times in a write heavy workload?	No	Yes	Yes
Individuality	Fort he children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes	no

Se decidio que reserva estará embedido dentro de cliente.

Relacion: Cliente+Servicio

Guideline Name	Question	Embed (yes/no)	Reference (opposite of embed)	Respuesta
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No	Yes
Go Together	Do the pieces of information have a “has.a” , “contains” or similar relationship?	Yes	No	Yes
Query Atomicity	Does the application query the pieces of information together?	Yes	No	Yes
Update Complexity	Are the pieces of information updated together?	Yes	No	Yes
Archival	Should the pieces of information be archived at the same time?	No	Yes	Yes
Cardinality	Is there a high cardinality current or growing in the child side of the relationship?	No	Yes	No
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes	No
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes	No
Document Growth	Would the embedded piece grow without bound ?	No	Yes	Yes

Workload	Are the pieces of information writte at different times in a write heavy workload?	No	Yes	Yes
Individuality	Fort he children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes	no

Se decide hacer esta relacion embedida dentro de servicio

Relacion: Cliente + Consumo

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

Se decide hacer esta relacion como embedida en cliente

Relacion; Consumo+ Servicio

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

Se decide hacer esta relacion al embeber servicio dentro de consumo

Al final se decidio hacer todas las relaciones referenciadas debido a que no se encontro como hacer las relaciones embedidas para que springboot la reconociera.

d. Una descripción gráfica usando Json de cada relación entre entidades en donde presente un ejemplo de datos junto con el esquema de asociación usado (referenciado o embebido). En el anexo D se muestra un ejemplo de lo que se requiere.

Relacion: Habitacion + Reserva. **Reserva**

Habitacion

```
{_id: <ObjectId1>
‘id’: 1,
   "tipo": a,
   "capacidadMax": 2 ,
   "dotacion": ["TV", "Wi-Fi", "Air Conditioning"],
   "estado": "Disponible",
   "tarifa": 150.00,
   "consumo": [<ObjectId3>,<ObjectId4>]
}
```

```
{_id: <ObjectId2>
‘id’: 1,
fechaEntrada": ISODate("2023-04-01"),
"fechaSalida": ISODate("2023-04-05"),
"numeroPersonas": 2,
"estado": 'Confirmada',
'habitacion': <ObjectId1>}
```

Relacion: Habitacion+ Consumo.

Consumo

habitacion

```
{_id: <ObjectId1>
‘id’: 1,
   "tipo": a,
   "capacidadMax": 2 ,
   "dotacion": ["TV", "Wi-Fi", "Air Conditioning"],
   "estado": "Disponible",
   "tarifa": 150.00,
   "consumo": [<ObjectId3>,<ObjectId4>]
}
```

```
{_id: <ObjectId3>

“id’: 1, fecha": ISODate("2023-03-01"), "descripcion":
"Room Service", "precioUnitario": 8.00, "cantidad":
2, "total": 16.00, “servicio”: [<ObjectId10>]

`
```

Relacion: Reserva + Cliente

Reserva

cliente

```
{_id: <ObjectId6>

‘id’: 1,”fechaEntrada": ISODate("2023-04-
01"), "fechaSalida": ISODate("2023-04-05"),
"numeroPersonas": 2, "estado":
'Confirmada', 'habitacion':
[getRandomId("habitaciones")]}
```

```
{_id: <ObjectId7>
‘Id’: 1,
   "direccion": "Address " ,
   "telefono": "123-456-789" ,
   "apellido": "LastName " ,
   "reservas": [<ObjectId6],
   "consumos": [<ObjectId3>]

   "servicios": [<Object10>]
```

Relacion Cliente + Consumo

Consumo

```
{_id: <ObjectId3>

“id’: 1, fecha": ISODate("2023-03-01"), "descripcion":
"Room Service", "precioUnitario": 8.00, "cantidad":
2, "total": 16.00, “servicio”: [<ObjectId10>]

}
```

Relacion; Cliente+ Servicio

Servicio

{_id: <ObjectId10>

id’: 1,”nombre": "Wi-Fi", "capacidad": 100, "horario": "24/7", "costo": 10.00

Consumo.

{_id: <ObjectId3>

“id’: 1, fecha": ISODate("2023-03-01"), "descripcion": "Room Service", "precioUnitario": 8.00, "cantidad": 2, "total": 16.00, “servicio”: [<ObjectId10>]

}

Servicio

{_id: <ObjectId10>

id’: 1,”nombre": "Wi-Fi", "capacidad": 100, "horario": "24/7", "costo": 10.00

c. (10%) Cree en MongoDB las colecciones principales de su base, así como la validación de los esquemas. Puede usar Compass o Mongo Shell (Mongosh) para realizar este proceso. Guarde lo hecho en un archivo. Anexe a los entregables los archivos con los scripts utilizados.

```
db.servicios.insertMany([
  { ‘id’: 1,”nombre": "Wi-Fi", "capacidad": 100, "horario": "24/7", "costo": 10.00 },
  { ‘id’: 2,”nombre": "Gym", "capacidad": 50, "horario": "9 AM - 9 PM", "costo": 20.00 },
  { ‘id’: 3,”nombre": "Spa", "capacidad": 20, "horario": "10 AM - 8 PM", "costo": 30.00 }
]);
db.consumos.insertMany([
  { ‘id’: 1, fecha": ISODate("2023-03-01"), "descripcion": "Room Service", "precioUnitario": 8.00,
"cantidad": 2, "total": 16.00 },
  {‘id’: 2, "fecha": ISODate("2023-03-05"), "descripcion": "Minibar", "precioUnitario": 5.00, "cantidad": 3,
"total": 15.00 },
  { ‘id’: 3, ”fecha": ISODate("2023-03-10"), "descripcion": "Restaurant", "precioUnitario": 15.00,
"cantidad": 1, "total": 15.00 }
]);
db.tipos.insertOne({
  "nombre": "Doble"
})
function getRandomTipoId() {
  var randomTipo = db.tipos.aggregate([{ $sample: { size: 1 } }]).next();
  return randomTipo._id;
}
```

```
for (var i = 0; i < 5; i++) {
  var habitacionDocument = {
            ‘id’: i,
    "tipo": getRandomTipoId(),
    "capacidadMax": 2 + i,
    "dotacion": ["TV", "Wi-Fi", "Air Conditioning"],
    "estado": "Disponible",
    "tarifa": 150.00 + i * 10,
    "consumo": [getRandomId("consumos")] };
  db.habitaciones.insertOne(habitacionDocument);
}
// Create sample data for the servicios collection

function getRandomTipoId() {
  var randomTipo = db.tipos.aggregate([{ $sample: { size: 1 } }]).next();
  return randomTipo.id;
}
db.reservas.insertMany([
  { ‘id’: 1,”fechaEntrada": ISODate("2023-04-01"), "fechaSalida": ISODate("2023-04-05"),
"numeroPersonas": 2, "estado": 'Confirmada', 'habitacion': [getRandomId("habitaciones")] },
  { ‘id’: 2”,fechaEntrada": ISODate("2023-04-10"), "fechaSalida": ISODate("2023-04-15"),
"numeroPersonas": 1, "estado": 'Pendiente', 'habitacion': [getRandomId("habitaciones")] },
  { ‘id’: 3,”fechaEntrada": ISODate("2023-04-20"), "fechaSalida": ISODate("2023-04-25"),
"numeroPersonas": 3, "estado": "Cancelada" ,'habitacion': [getRandomId("habitaciones")]}
]);
function getRandomId(collection) {
  var randomDoc = db[collection].aggregate([{ $sample: { size: 1 } }]).next();
  return randomDoc.id;
}
for (var i = 0; i < 3; i++) {
  var clienteDocument = {
‘Id’: i,
    "direccion": "Address " + i,
    "telefono": "123-456-789" + i,
    "apellido": "LastName " + i,
    "reservas": [getRandomId("reservas")],
    "consumos": [getRandomId("consumos")],
    "servicios": [getRandomId("servicios")]
  };
  // Insert the cliente document into the clientes collection
  db.clientes.insertOne(clienteDocument);
}
```

ISIS2304C21202320.clientes

Documents | Aggregations | Schema | Indexes | **Validation**

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'direccion',
      'telefono',
      'apellido',
      'reservas',
      'consumos',
      'servicios'
    ],
    properties: {
      direccion: {
        bsonType: 'string'
      },
      telefono: {
        bsonType: 'string'
      },
      apellido: {
        bsonType: 'string'
      },
      reservas: {
        bsonType: 'array',
        items: {
          bsonType: 'objectId'
        }
      },
      consumos: {
        bsonType: 'array',
        items: {
          bsonType: 'objectId'
        }
      },
      servicios: {
        bsonType: 'array',
        items: {
          bsonType: 'objectId'
        }
      }
    }
  }
}
```

ISIS2304C21202320.consumos

Documents | Aggregations | Schema | Indexes | **Validation**

Validation Action: Error | Validation Level: Strict

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'fecha',
      'descripcion',
      'precioUnitario',
      'cantidad',
      'total'
    ],
    properties: {
      fecha: {
        bsonType: 'date'
      },
      descripcion: {
        bsonType: 'string'
      },
      precioUnitario: {
        bsonType: 'double'
      },
      cantidad: {
        bsonType: 'int'
      },
      total: {
        bsonType: 'double'
      }
    }
  }
}
```

ISIS2304C21202320.habitaciones

Documents Aggregations Schema Indexes Validation

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'tipo',
      'capacidadMax',
      'dotacion',
      'estado',
      'tarifa',
      'consumo'
    ],
    properties: {
      tipo: {
        bsonType: 'objectId'
      },
      capacidadMax: {
        bsonType: 'int'
      },
      dotacion: {
        bsonType: 'array',
        items: {
          bsonType: 'string'
        }
      },
      estado: {
        'enum': [
          'Disponible',
          'Ocupada',
          'Mantenimiento'
        ]
      },
      tarifa: {
        bsonType: 'double'
      },
      consumo: {
        bsonType: 'array',
        items: {
          bsonType: 'objectId'
        }
      }
    }
  }
}
```

ISIS2304C21202320.reservas

Documents Aggregations Schema Indexes Validation

Validation Action: Error Validation Level: Strict

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'fechaEntrada',
      'fechaSalida',
      'numeroPersonas',
      'estado',
      'habitacion'
    ],
    properties: {
      fechaEntrada: {
        bsonType: 'date'
      },
      fechaSalida: {
        bsonType: 'date'
      },
      numeroPersonas: {
        bsonType: 'int'
      },
      estado: {
        bsonType: 'string'
      },
      habitacion: {
        bsonType: 'objectId'
      }
    }
  }
}
```

ISIS2304C21202320.servicios

Documents Aggregations Schema Indexes Validation

Validation Action: Error Validation Level: Strict

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'nombre',
      'capacidad',
      'horario',
      'costo'
    ],
    properties: {
      nombre: {
        bsonType: 'string'
      },
      capacidad: {
        bsonType: 'int'
      },
      horario: {
        bsonType: 'string'
      },
      costo: {
        bsonType: 'double'
      }
    }
  }
}
```

ISIS2304C21202320.tipos

Documents Aggregations Schema Ind

Validation Action: Error

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'nombre'
    ],
    properties: {
      nombre: {
        bsonType: 'string'
      }
    }
  }
}
```

4. **(5%)** Escenarios de prueba: realice las operaciones descritas abajo (ver escenarios de prueba) para poder realizar pruebas sobre los RF de CRUD (RF1-RF7) así como sobre los RF de Consulta (RFC1 a RFC3 + 1 RFC de consulta avanzado seleccionado) del documento marco del caso de estudio . Anexe los scripts SQL cuando sea necesario.

Por validación:

```
> db.tipos.insertOne({
  nombre:1})
```

```
MongoServerError: Document failed validation
```

```
> db.reservas.insertOne({
            "fechaEntrada": {
                "$date": "2021-12-28T05:00:00.000+00:00"
            },
            "fechaSalida": {
                "$date": "2021-12-28T05:00:00.000+00:00"
            },
            "numeroPersonas": 3,
            "estado": "Aprovada",
            "id": 1
        })
```

```
MongoServerError: Document failed validation
```

```
> db.servicios.insertOne( {
              "nombre": "Wi-Fi gratuito",
              "capacidad": 100,
              "costo": 0,
              "horario": "12pm-12am",
              "id": 1
          },
          {
              "nombre": "Piscina climatizada",
              "capacidad": 50,
              "costo": 30000,
              "horario": "7am - 7pm",
              "id": 2
          })
```

```
MongoServerError: Document failed validation
```

```
ISIS2304C21202320>
```

```
> db.habitaciones.insertOne({
            "tipo": 12,
            "capacidadMax": 10,
            "dotacion": [
                "Wi-Fi",
                null,
                "Air Conditioning",
                null,
                "Mini bar",
                null,
                "Jacuzzi",
                null,
                "Working Table",
                null,
                "Safebox",
                null,
                "Balcony",
                null,
                "Sea view",
                null
            ],
            "estado": "Ocupada",
            "tarifa": 2077893,
            "consumo": [
                154338,
                248620,
                234575

            ],
            "id": 1
        })
```

▸ **MongoServerError:** Document failed validation

```
> db.clientes.insertOne({
            "direccion": "calle100carrera78",
            "apellido": "pinzon",
            "telefono": 3650585673,
            "reservas": [
                44972,
                9659,
                42441,
                7482
            ],
            "consumos": [
                154338,
                248620,
                234575

            ],
            "servicios": [
                9,
                35,
                27

            ],
            "id": 10000})
```

⊗ ▸ **MongoServerError:** Document failed validation

```
MongoServerError: Document failed validation
> db.consumos.insertOne({
            "fecha": {
                "$date": "2021-12-28T05:00:00.000+00:00"
            },
            "descripcion": "Guide",
            "precioUnitario": 103602,
            "cantidad": 13,
            "total": 1346826,
            "id": 1
        })
MongoServerError: Document failed validation
```