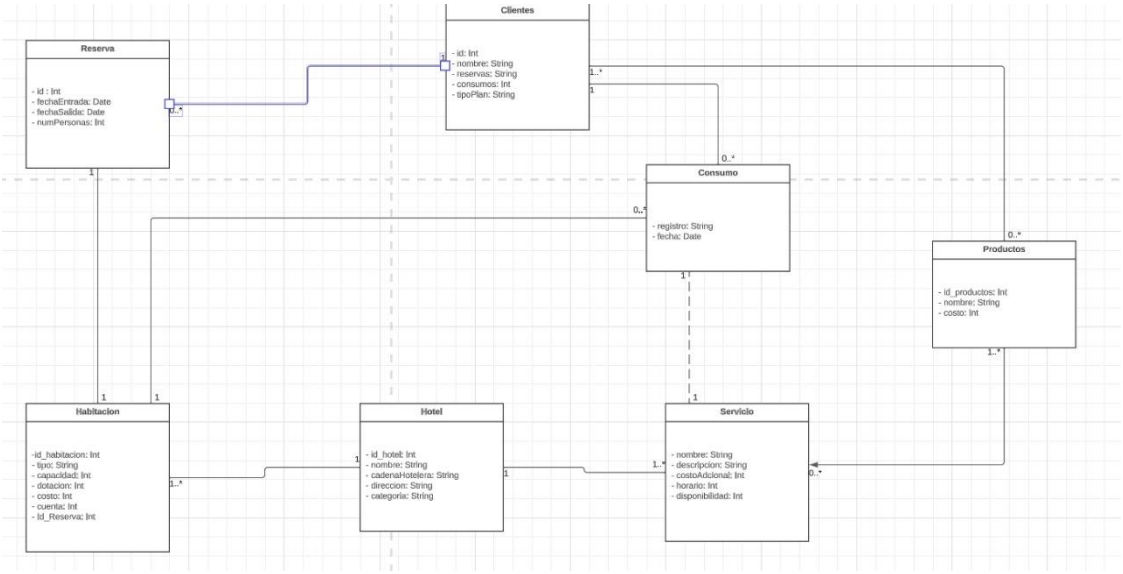
xACTIVIDADES A DESARROLLAR: DISEÑO Y CONSTRUCCIÓN DE LA APLICACIÓN.

(7%) Análisis y modelo conceptual

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



Este modelo de UML representa las conexiones entre las clases. Pese a ser el mismo modelo de la entrega pasada, este no cuenta con las clases de PlanDeConsumo ni de Usuario, ya que para el contexto no son necesarias.

La dificultad con esta nueva entrega radica que estas clases eran necesarias para consultas básicas, por esto mismo se agregaron nuevas colecciones para representar las consultas.

(35%) Diseño de la base de datos

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

A. Identifiquen entidades y sus atributos.

Colección: hotel

- **id_hotel**: Número (obligatorio).
- **nombre**: Cadena (obligatorio).
- **cadenahotelera**: Cadena (obligatorio).
- **direccion**: Cadena (obligatorio).
- **categoria**: Cadena (obligatorio).

tipo_habitacion: Arreglo de objetos con los siguientes campos:

- **id_tipo_habitacion**: Número (obligatorio).
- **nombre**: Cadena (obligatorio).

- **dotacion**: Cadena (obligatorio).

Colección: habitacion

- **id_habitacion**: Número (obligatorio).
- **tipo**: Cadena (obligatorio).
- **capacidad**: Número (obligatorio).
- **costo**: Número (obligatorio).
- **cuenta**: Número (obligatorio).

Colección: servicio

- **id_servicio**: Número (obligatorio).
- **nombre**: Cadena (obligatorio).
- **descripcion**: Cadena (obligatorio).
- **costoadicional**: Número (obligatorio).
- **horario**: Cadena (obligatorio).
- **disponibilidad**: Número (obligatorio).

Colección: reserva

- **id_reserva**: Número (obligatorio).
- **id_cliente**: Número (obligatorio).
- **id_habitacion**: Número (obligatorio).
- **fechaentrada**: Cadena (obligatorio).
- **fechasalida**: Cadena (obligatorio).
- **numpersonas**: Número (obligatorio).

Colección: llegada_cliente

- **id_llegada**: Número (obligatorio).
- **id_reserva**: Número (obligatorio).
- **fechallegada**: Cadena (obligatorio).

Colección: consumo_servicio

- **id_consumo**: Número (obligatorio).
- **id_cliente**: Número (obligatorio).
- **id_servicio**: Número (obligatorio).
- **fecha_consumo**: Cadena (obligatorio).
- **cantidad**: Número (obligatorio).

Colección: salida_cliente

- **id_salida**: Número (obligatorio).
- **id_cliente**: Número (obligatorio).
- **fecha_salida**: Cadena (obligatorio).
- **monto_total**: Número (obligatorio)

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

- **Tipos de habitación:**
- Creación/Modificación: 1 vez por semana (52 veces al año).
- Consulta: 1 vez por semana (52 veces al año).
- Estimación de registros: 20 registros.
- **Habitaciones:**
- Creación/Modificación: 2 veces por semana (104 veces al año).
- Consulta: 2 veces por semana (104 veces al año).
- Estimación de registros: 200 registros.
- **Servicios del hotel:**
- Creación/Modificación: 2 veces por semana (104 veces al año).
- Consulta: 2 veces por semana (104 veces al año).
- Estimación de registros: 35 registros.
- **Reservación de habitación:**
- Creación/Modificación: 100 veces por día (36,500 veces al año).
- Consulta: 50 veces por día (18,250 veces al año).
- Estimación de registros: 50,000 reservaciones por año.
- **Consumos de servicios:**
- Creación/Modificación: 500 veces por día (182,500 veces al año).
- Consulta: 250 veces por día (91,250 veces al año).
- Estimación de registros: 250,000 consumos en 3 años.
- **Llegadas y partidas de clientes:**
- Creación/Modificación: 100 veces por día (36,500 veces al año).
- Consulta: 50 veces por día (18,250 veces al año).
- Estimación de registros: 50,000 llegadas y partidas por año.

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

Entidad	Operación	Tipos de Datos Asociados	Tipo (Lectura/Escritura)	Tiempo de Ejecución (ms)
Tipos de Habitación	Creación/Modificación	ID de tipo de habitación,	Escritura	10

		Nombre de tipo de habitación, Otras características específicas del tipo de habitación		
Tipos de Habitación	Consulta	ID de tipo de habitación, Nombre de tipo de habitación, Otras características específicas del tipo de habitación	Lectura	5
Habitaciones	Creación/Modificación	ID de habitación, Número de habitación, Tipo de habitación, Estado de la habitación, Otras características específicas de la habitación	Escritura	20
Habitaciones	Consulta	ID de habitación, Número de habitación, Tipo de habitación, Estado de la habitación, Otras características específicas de la habitación	Lectura	15
Servicios del Hotel	Creación/Modificación	ID de servicio, Nombre de servicio, Descripción del servicio, Costo del servicio, Otras características específicas del servicio	Escritura	12
Servicios del Hotel	Consulta	ID de servicio, Nombre de servicio, Descripción del	Lectura	8

		servicio, Costo del servicio, Otras características específicas del servicio		
Reservación de Habitación	Creación/Modificación	ID de reserva, ID de cliente, ID de habitación, Fecha de inicio y fin de la reserva, Estado de la reserva, Otras características específicas de la reserva	Escritura	30
Reservación de Habitación	Consulta	ID de reserva, ID de cliente, ID de habitación, Fecha de inicio y fin de la reserva, Estado de la reserva, Otras características específicas de la reserva	Lectura	25
Consumos de Servicios	Creación/Modificación	ID de consumo, ID de reserva, ID de servicio, Cantidad o duración del consumo, Fecha y hora del consumo, Otras características específicas del consumo	Escritura	18
Consumos de Servicios	Consulta	ID de consumo, ID de reserva, ID de servicio, Cantidad o duración del consumo, Fecha y hora del consumo, Otras características específicas del consumo	Lectura	10

Llegadas y Partidas de Clientes	Creación/Modificación	ID de llegada o partida, ID de reserva, ID de cliente, Fecha y hora de llegada o partida, Otras características específicas de la llegada o partida	Escritura	25
Llegadas y Partidas de Clientes	Consulta	ID de llegada o partida, ID de reserva, ID de cliente, Fecha y hora de llegada o partida, Otras características específicas de la llegada o partida	Lectura	20

- (15%) Describan las entidades de datos y las relaciones entre ellas que corresponden al modelo conceptual UML propuesto. Para ello, presenten lo siguiente:

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

- Tipo de Habitación:
- Descripción: Almacena información sobre los tipos de habitación disponibles en el hotel.
- Habitación:
- Descripción: Contiene detalles específicos de cada habitación en el hotel.
- Servicio del Hotel:
- Descripción: Guarda información sobre los servicios ofrecidos por el hotel.
- Reservación de Habitación:
- Descripción: Registra las reservaciones de habitaciones realizadas por los clientes.
- Consumo de Servicios:
- Descripción: Contiene detalles sobre los consumos de servicios asociados a las reservaciones.
- Llegadas y Partidas de Clientes:
- Descripción: Registra la información sobre las llegadas y partidas de los clientes.

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

- **Tipo de Habitación - Habitación**

- Relación: Uno a muchos (Un tipo de habitación puede tener muchas habitaciones, pero una habitación pertenece a un solo tipo).
- **Habitación - Servicio del Hotel**
- Relación: Muchos a muchos (Una habitación puede tener varios servicios, y un servicio puede estar asociado a varias habitaciones).
- **Habitación - Reservación de Habitación**
- Relación: Uno a muchos (Una habitación puede tener múltiples reservaciones, pero una reservación está asociada a una sola habitación).
- **Reservación de Habitación - Consumo de Servicios**
- Relación: Uno a muchos (Una reservación puede tener varios consumos de servicios, pero un consumo está asociado a una sola reservación).
- **Reservación de Habitación - Llegadas y Partidas de Clientes**
- Relación: Uno a uno (Cada reservación tiene una llegada y partida específica).

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.

- **Tipo de Habitación - Habitación:**
- Esquema: Referenciado (La información del tipo de habitación se referencia en la habitación).
- **Habitación - Servicio del Hotel:**
- Esquema: Embebido (Los servicios de la habitación se almacenan directamente en la información de la habitación).
- **Habitación - Reservación de Habitación:**
- Esquema: Referenciado (La información de la habitación se referencia en la reservación).
- **Reservación de Habitación - Consumo de Servicios:**
- Esquema: Embebido (La información del consumo se almacena directamente en la información de la reservación).
- **Reservación de Habitación - Llegadas y Partidas de Clientes:**
- Esquema: Referenciado (La información de la reservación se referencia en las llegadas y partidas de clientes).

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

(58%)1 Implemente los 7 requerimientos funcionales de CRUD (RF1 a RF7) y los 4 requerimientos funcionales de consulta (RFC1 a RFC3 + 1 RFC avanzado que escojan) solicitados en el documento marco, a través de una interfaz gráfica desde la aplicación

1)

Crear un tipo de habitación:

```
db.createCollection("hotel", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["id_hotel", "nombre", "cadenahotelera", "direccion", "categoria", "tipo_habitacion"],
      properties: {
        id_hotel: {
          bsonType: "number",
          description: "Debe ser un número y es obligatorio."
        },
```

```
nombre: {
  bsonType: "string",
  description: "Debe ser una cadena y es obligatorio."
},
cadenahotelera: {
  bsonType: "string",
  description: "Debe ser una cadena y es obligatorio."
},
direccion: {
  bsonType: "string",
  description: "Debe ser una cadena y es obligatorio."
},
categoria: {
  bsonType: "string",
  description: "Debe ser una cadena y es obligatorio."
},
tipo_habitacion: {
  bsonType: "array",
  description: "Debe ser un arreglo de tipos de habitación.",
  items: {
    bsonType: "object",
    required: ["id_tipo_habitacion", "nombre", "dotacion"],
    properties: {
      id_tipo_habitacion: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      nombre: {
        bsonType: "string",
        description: "Debe ser una cadena y es obligatorio."
      },
```

```
            dotacion: {
              bsonType: "string",
              description: "Debe ser una cadena y es obligatorio."
            }
            // Otros campos del tipo de habitación
          }
        }
      }
      // Otros campos de la tabla hotel
    }
  }
}
})
```

Insertar un nuevo tipo de habitación:

```
db.hotel.updateOne(
 { id_hotel: 1 },
 {
   $push: {
     tipo_habitacion: {
       id_tipo_habitacion: 2,
       nombre: "Doble",
       dotacion: "TV, Baño privado, Wi-Fi"
       // Otros campos del tipo de habitación
     }
   }
 }
);
```

Actualizar un tipo de habitación:

```
db.hotel.updateOne(
  { "tipo_habitacion.id_tipo_habitacion": 1 },
  { $set: { "tipo_habitacion.$.dotacion": "TV, Baño privado, Wi-Fi, Minibar" } }
)
```

Borrar un tipo de habitación:

```
db.hotel.updateOne(
  { id_hotel: 1 },
  { $pull: { tipo_habitacion: { id_tipo_habitacion: 1 } } }
)
```

Consultar tipos de habitaciones:

```
// Obtener todos los tipos de habitaciones de un hotel
db.hotel.findOne({ id_hotel: 1 }, { _id: 0, tipo_habitacion: 1 })
```

Consultar 1 solo tipo de habitación:

```
db.hotel.findOne(
  { "id_hotel": hotelId, "tipo_habitacion.id_tipo_habitacion": tipoHabitacionId },
  { _id: 0, "tipo_habitacion.$": 1 }
)
```

2)

Crear habitación:

```
db.createCollection("habitacion", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["id_habitacion", "tipo", "capacidad", "costo", "cuenta"],
      properties: {
```

```
      id_habitacion: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      tipo: {
        bsonType: "string",
        description: "Debe ser una cadena y es obligatorio."
      },
      capacidad: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      costo: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      cuenta: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      }
      // Otros campos de la habitación
    }
  }
 }
})
```

Insertar habitación:

```
db.habitacion.insertOne({
  id_habitacion: 101,
  tipo: "Doble",
```

```
  capacidad: 2,
  costo: 150,
  cuenta: 0
})
```

Actualizar habitación:

```
db.habitacion.updateOne(
  { id_habitacion: 101 },
  { $set: { costo: 180 } }
)
```

Borrar habitación:

```
db.habitacion.deleteOne({ id_habitacion: 101 })
```

Consultar habitación:

```
// Obtener todas las habitaciones
db.habitacion.find()

// Obtener una habitación específica por su ID
db.habitacion.findOne({ id_habitacion: 101 })
```

3)

Crear servicio:

```
db.createCollection("servicio", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["id_servicio", "nombre", "descripcion", "costoadicional", "horario", "disponibilidad"],
      properties: {
        id_servicio: {
```

```
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      nombre: {
        bsonType: "string",
        description: "Debe ser una cadena y es obligatorio."
      },
      descripcion: {
        bsonType: "string",
        description: "Debe ser una cadena y es obligatorio."
      },
      costoadicional: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      horario: {
        bsonType: "string",
        description: "Debe ser una cadena y es obligatorio."
      },
      disponibilidad: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      }
      // Otros campos del servicio
    }
  }
 }
})
```

Insertar un servicio:

```
db.servicio.insertOne({
  id_servicio: 1,
  nombre: "Piscina",
  descripcion: "Piscina al aire libre",
  costoadicional: 20,
  horario: "9:00 AM - 6:00 PM",
  disponibilidad: 1
})
```

Actualizar un servicio:

```
db.servicio.updateOne(
  { id_servicio: 1 },
  { $set: { costoadicional: 25, horario: "10:00 AM - 7:00 PM" } }
)
```

Borrar un servicio:

```
db.servicio.deleteOne({ id_servicio: 1 })
```

Consultar un servicio:

```
// Obtener todos los servicios
db.servicio.find()

// Obtener un servicio específico por su ID
db.servicio.findOne({ id_servicio: 1 })
```

4)

Crear una reserva:

```
db.createCollection("reserva", {
  validator: {
```

```
  $jsonSchema: {
    bsonType: "object",
    required: ["id_reserva", "id_cliente", "id_habitacion", "fechaentrada", "fechasalida",
"numpersonas"],
    properties: {
      id_reserva: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      id_cliente: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      id_habitacion: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      fechaentrada: {
        bsonType: "string",
        description: "Debe ser una cadena y es obligatorio."
      },
      fechasalida: {
        bsonType: "string",
        description: "Debe ser una cadena y es obligatorio."
      },
      numpersonas: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      }
      // Otros campos de la reserva
    }
```

```
    }
  }
})
```

Insertar una reserva:

```
db.reserva.insertOne({
  id_reserva: 1,
  id_cliente: 101,
  id_habitacion: 201,
  fechaentrada: "2023-01-01",
  fechasalida: "2023-01-05",
  numpersonas: 2
})
```

Actualizar una reserva:

```
db.reserva.updateOne(
  { id_reserva: 1 },
  { $set: { fechasalida: "2023-01-08", numpersonas: 3 } }
)
```

Borrar una reserva:

```
db.reserva.deleteOne({ id_reserva: 1 })
```

Consultar una reserva:

```
// Obtener todas las reservas
db.reserva.find()
```

```
// Obtener una reserva específica por su ID
db.reserva.findOne({ id_reserva: 1 })



5)
Crear llegada de cliente:
db.createCollection("llegada_cliente", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["id_llegada", "id_reserva", "fechallegada"],
      properties: {
        id_llegada: {
          bsonType: "number",
          description: "Debe ser un número y es obligatorio."
        },
        id_reserva: {
          bsonType: "number",
          description: "Debe ser un número y es obligatorio."
        },
        fechallegada: {
          bsonType: "string",
          description: "Debe ser una cadena y es obligatorio."
        }
        // Otros campos de la llegada del cliente
      }
    }
  }
})
```

Insertar llegada de cliente:

```
db.llegada_cliente.insertOne({
  id_llegada: 1,
  id_reserva: 1,
  fechallegada: "2023-01-01T12:00:00Z"
})
```

Actualizar llegada de cliente:

```
db.llegada_cliente.updateOne(
  { id_llegada: 1 },
  { $set: { fechallegada: "2023-01-01T13:30:00Z" } }
)
```

Borrar una llegada de cliente:

```
db.llegada_cliente.deleteOne({ id_llegada: 1 })
```

Consultar llegadas de clientes:

```
// Obtener todas las llegadas de clientes
db.llegada_cliente.find()

// Obtener una llegada de cliente específica por su ID
db.llegada_cliente.findOne({ id_llegada: 1 })
```

6)

Crear consumo:

```
db.createCollection("consumo_servicio", {
  validator: {
    $jsonSchema: {
```

```
    bsonType: "object",
    required: ["id_consumo", "id_cliente", "id_servicio", "fecha_consumo", "cantidad"],
    properties: {
     id_consumo: {
      bsonType: "number",
      description: "Debe ser un número y es obligatorio."
     },
     id_cliente: {
      bsonType: "number",
      description: "Debe ser un número y es obligatorio."
     },
     id_servicio: {
      bsonType: "number",
      description: "Debe ser un número y es obligatorio."
     },
     fecha_consumo: {
      bsonType: "string",
      description: "Debe ser una cadena y es obligatorio."
     },
     cantidad: {
      bsonType: "number",
      description: "Debe ser un número y es obligatorio."
     }
     // Otros campos del consumo de servicio
    }
   }
  }
})
```

Insertar un consumo:

```
db.consumo_servicio.insertOne({
```

```
  id_consumo: 1,
  id_cliente: 101,
  id_servicio: 1,
  fecha_consumo: "2023-01-01T15:30:00Z",
  cantidad: 2
})
```

Actualizar un consumo:

```
db.consumo_servicio.updateOne(
  { id_consumo: 1 },
  { $set: { cantidad: 3 } }
)
```

Borrar un consumo:

```
db.consumo_servicio.deleteOne({ id_consumo: 1 })
```

Consultar consumo:

```
// Obtener todos los consumos de servicios
db.consumo_servicio.find()

// Obtener un consumo de servicio específico por su ID
db.consumo_servicio.findOne({ id_consumo: 1 })
```

7)

Crear salida de cliente:

```
db.createCollection("salida_cliente", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["id_salida", "id_cliente", "fecha_salida", "monto_total"],
      properties: {
        id_salida: {
```

```
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      id_cliente: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      },
      fecha_salida: {
        bsonType: "string",
        description: "Debe ser una cadena y es obligatorio."
      },
      monto_total: {
        bsonType: "number",
        description: "Debe ser un número y es obligatorio."
      }
      // Otros campos del registro de salida
    }
  }
 }
})
```

Insertar un registro de salida de cliente:

```
db.salida_cliente.insertOne({
 id_salida: 1,
 id_cliente: 101,
 fecha_salida: "2023-01-05T12:00:00Z",
 monto_total: 250
})
```

Actualizar salida de cliente:

```
db.salida_cliente.updateOne(
```

```
  { id_salida: 1 },
  { $set: { monto_total: 280 } }
)
```

Borrar salida de cliente:

```
db.salida_cliente.deleteOne({ id_salida: 1 })
```

Consultar salida de cliente:

```
// Obtener todos los registros de salida de clientes
db.salida_cliente.find()

// Obtener un registro de salida de cliente específico por su ID
db.salida_cliente.findOne({ id_salida: 1 })
```

Requerimientos de consultas básicos:

1)

```
// Obtener el dinero recolectado por servicios en cada habitación en el último año corrido
const fechaInicio = new Date();  // Fecha actual
fechaInicio.setFullYear(fechaInicio.getFullYear() - 1);  // Restar un año

db.consumo_servicio.aggregate([
  {
    $match: {
      fecha_consumo: { $gte: fechaInicio.toISOString() }  // Filtrar por consumos en el último año
    }
  },
  {
    $lookup: {
      from: "habitacion",  // Nombre de la colección de habitaciones
      localField: "id_habitacion",  // Campo en consumo_servicio que se relaciona con habitacion
```

```
    foreignField: "id_habitacion",  // Campo en habitacion que se relaciona con consumo_servicio
    as: "habitacion_info"
   }
  },
  {
   $unwind: "$habitacion_info"  // Deshacer el array creado por $lookup
  },
  {
   $group: {
    _id: "$habitacion_info.id_habitacion",  // Agrupar por ID de habitación
    habitacion: { $first: "$habitacion_info" },  // Obtener la información de la habitación
    totalRecaudado: { $sum: { $multiply: ["$cantidad", "$habitacion_info.costo"] } }  // Calcular el total recolectado por habitación
   }
  },
  {
   $project: {
    _id: 0,  // Excluir el campo _id en el resultado final
    id_habitacion: "$_id",
    nombre_habitacion: "$habitacion.nombre",  // Puedes ajustar según la estructura de tu colección habitacion
    totalRecaudado: 1
   }
  }
]);
```

2)

```
// Obtener el índice de ocupación de cada habitación en el último año corrido
const fechaInicio = new Date();  // Fecha actual
fechaInicio.setFullYear(fechaInicio.getFullYear() - 1);  // Restar un año
```

```
db.reserva.aggregate([
 {
  $match: {
   fechaentrada: { $gte: fechaInicio.toISOString() }  // Filtrar por reservas en el último año
  }
 },
 {
  $group: {
   _id: "$id_habitacion",
   totalReservas: { $sum: 1 }  // Contar el número de reservas por habitación
  }
 },
 {
  $lookup: {
   from: "habitacion",  // Nombre de la colección de habitaciones
   localField: "_id",  // Campo en el resultado de la etapa $group que se relaciona con habitacion
   foreignField: "id_habitacion",  // Campo en habitacion que se relaciona con reserva
   as: "habitacion_info"
  }
 },
 {
  $unwind: "$habitacion_info"  // Deshacer el array creado por $lookup
 },
 {
  $project: {
   _id: 0,  // Excluir el campo _id en el resultado final
   id_habitacion: "$_id",
   nombre_habitacion: "$habitacion_info.nombre",  // Ajusta según la estructura de tu colección habitacion
   indiceOcupacion: { $divide: ["$totalReservas", "$habitacion_info.capacidad"] }  // Calcular el índice de ocupación (%)
```

```
   }
  }
]);
```

3)

```
// Definir el ID del cliente y el rango de fechas
const idCliente = 123; // Reemplaza con el ID del cliente específico
const fechaInicio = new Date('2022-01-01'); // Reemplaza con la fecha de inicio del rango
const fechaFin = new Date('2022-12-31');   // Reemplaza con la fecha de fin del rango

// Obtener el consumo en HotelAndes por un cliente en el rango de fechas indicado
const resultado = db.consumo_servicio.aggregate([
 {
  $match: {
   id_cliente: idCliente,
   fecha_consumo: {
    $gte: fechaInicio.toISOString(),
    $lte: fechaFin.toISOString()
   }
  }
 },
 {
  $lookup: {
   from: "servicio", // Nombre de la colección de servicios
   localField: "id_servicio", // Campo en consumo_servicio que se relaciona con servicio
   foreignField: "id_servicio", // Campo en servicio que se relaciona con consumo_servicio
   as: "servicio_info"
  }
 },
 {
  $unwind: "$servicio_info" // Deshacer el array creado por $lookup
```

```
  },
  {
    $project: {
      _id: 0,
      id_consumo: "$id_consumo",
      id_servicio: "$servicio_info.id_servicio",
      nombre_servicio: "$servicio_info.nombre",
      descripcion_servicio: "$servicio_info.descripcion",
      costo_servicio: "$servicio_info.costoadicional",
      fecha_consumo: "$fecha_consumo",
      cantidad: "$cantidad"
    }
  }
]);

printjson(resultado.toArray());
```

Avanzados:

4)

```
// Definir el servicio y el rango de fechas
const idServicio = 1; // Reemplaza con el ID del servicio específico
const fechaInicio = new Date('2022-01-01'); // Reemplaza con la fecha de inicio del rango
const fechaFin = new Date('2022-12-31');   // Reemplaza con la fecha de fin del rango

// Definir el criterio de clasificación deseado (puedes ajustar según tus necesidades)
const criterioDeClasificacion = {
  "cliente.nombre": 1, // Ordenar por nombre del cliente de manera ascendente
  fecha_consumo: -1,    // Luego, ordenar por fecha de consumo de manera descendente
  cantidad: -1         // Finalmente, ordenar por la cantidad de consumo de manera descendente
};
```

```
// Obtener la información de los clientes que consumieron el servicio en el rango de fechas
const resultado = db.consumo_servicio.aggregate([
 {
  $match: {
   id_servicio: idServicio,
   fecha_consumo: {
    $gte: fechaInicio.toISOString(),
    $lte: fechaFin.toISOString()
   }
  }
 },
 {
  $lookup: {
   from: "clientes", // Nombre de la colección de clientes
   localField: "id_cliente", // Campo en consumo_servicio que se relaciona con clientes
   foreignField: "id_clientes", // Campo en clientes que se relaciona con consumo_servicio
   as: "cliente"
  }
 },
 {
  $unwind: "$cliente" // Deshacer el array creado por $lookup
 },
 {
  $group: {
   _id: "$cliente.id_clientes",
   cliente: { $first: "$cliente" },
   totalConsumos: { $sum: "$cantidad" } // Calcular el total de consumos por cliente
  }
 },
 {
```

```
    $sort: criterioDeClasificacion // Ordenar según el criterio definido
  }
]);

printjson(resultado.toArray());
```

5)

```
// Definir el servicio y el rango de fechas
const idServicio = 1; // Reemplaza con el ID del servicio específico
const fechaInicio = new Date('2022-01-01'); // Reemplaza con la fecha de inicio del rango
const fechaFin = new Date('2022-12-31');   // Reemplaza con la fecha de fin del rango

// Definir el criterio de clasificación deseado (puedes ajustar según tus necesidades)
const criterioDeClasificacion = {
  "cliente.nombre": 1, // Ordenar por nombre del cliente de manera ascendente
  fecha_consumo: 1,    // Luego, ordenar por fecha de consumo de manera ascendente
  "servicio.nombre": 1 // Finalmente, ordenar por nombre del servicio de manera ascendente
};

// Obtener la información de los clientes que NO consumieron el servicio en el rango de fechas
const resultado = |

printjson(resultado.toArray());
```

6)

```
// Obtener el servicio más consumido, el servicio menos consumido,
// las habitaciones más solicitadas y las habitaciones menos solicitadas
const resultado = db.reserva.aggregate([
  {
```

```
  $lookup: {
    from: "consumo_servicio",
    localField: "id_reserva",
    foreignField: "id_reserva",
    as: "consumos"
  }
},
{
  $unwind: "$consumos"
},
{
  $lookup: {
    from: "servicio",
    localField: "consumos.id_servicio",
    foreignField: "id_servicio",
    as: "servicio_info"
  }
},
{
  $unwind: "$servicio_info"
},
{
  $lookup: {
    from: "habitacion",
    localField: "id_habitacion",
    foreignField: "id_habitacion",
    as: "habitacion_info"
  }
},
{
  $unwind: "$habitacion_info"
```

```
},
{
  $project: {
    _id: 0,
    semana: { $isoWeek: "$fechaentrada" },
    id_reserva: "$id_reserva",
    id_servicio: "$servicio_info.id_servicio",
    nombre_servicio: "$servicio_info.nombre",
    id_habitacion: "$habitacion_info.id_habitacion",
    nombre_habitacion: "$habitacion_info.nombre"
  }
},
{
  $group: {
    _id: {
      semana: "$semana",
      id_servicio: "$id_servicio",
      nombre_servicio: "$nombre_servicio",
      id_habitacion: "$id_habitacion",
      nombre_habitacion: "$nombre_habitacion"
    },
    totalConsumos: { $sum: 1 }
  }
},
{
  $sort: {
    "_id.semana": 1,
    totalConsumos: -1
  }
},
{
```

```
    $group: {
      _id: "$_id.semana",
      servicioMasConsumido: { $first: "$_id.nombre_servicio" },
      servicioMenosConsumido: { $last: "$_id.nombre_servicio" },
      habitacionMasSolicitada: { $first: "$_id.nombre_habitacion" },
      habitacionMenosSolicitada: { $last: "$_id.nombre_habitacion" }
    }
  },
  {
    $sort: {
      _id: 1
    }
  }
]);

printjson(resultado.toArray());
```

7)

```
// Obtener la información de los clientes excelentes
const resultado = db.reserva.aggregate([
  {
    $lookup: {
      from: "clientes",
      localField: "id_cliente",
      foreignField: "id_clientes",
      as: "cliente_info"
    }
  },
  {
    $unwind: "$cliente_info"
```

```
},
{
  $project: {
    _id: 0,
    id_cliente: "$cliente_info.id_clientes",
    nombre_cliente: "$cliente_info.nombre",
    fecha_entrada: "$fechaentrada",
    fecha_salida: "$fechasalida",
    trimestre: {
      $let: {
        vars: {
          trimestres: [
            { $in: [{ $month: "$fechaentrada" }, [1, 2, 3]] },  // Primer trimestre
            { $in: [{ $month: "$fechaentrada" }, [4, 5, 6]] },  // Segundo trimestre
            { $in: [{ $month: "$fechaentrada" }, [7, 8, 9]] },  // Tercer trimestre
            { $in: [{ $month: "$fechaentrada" }, [10, 11, 12]] } // Cuarto trimestre
          ]
        },
        in: {
          $cond: {
            if: {
              $or: [
                { $and: ["$$trimestres.0", "$$trimestres.1", "$$trimestres.2"] },
                { $and: ["$$trimestres.1", "$$trimestres.2", "$$trimestres.3"] }
              ]
            },
            then: "Excelente",
            else: "No Excelente"
          }
        }
      }
```

```
      }
    }
  },
  {
    $match: {
      trimestre: "Excelente"
    }
  }
]);

printjson(resultado.toArray());
```