Entrega #3 – Proyecto

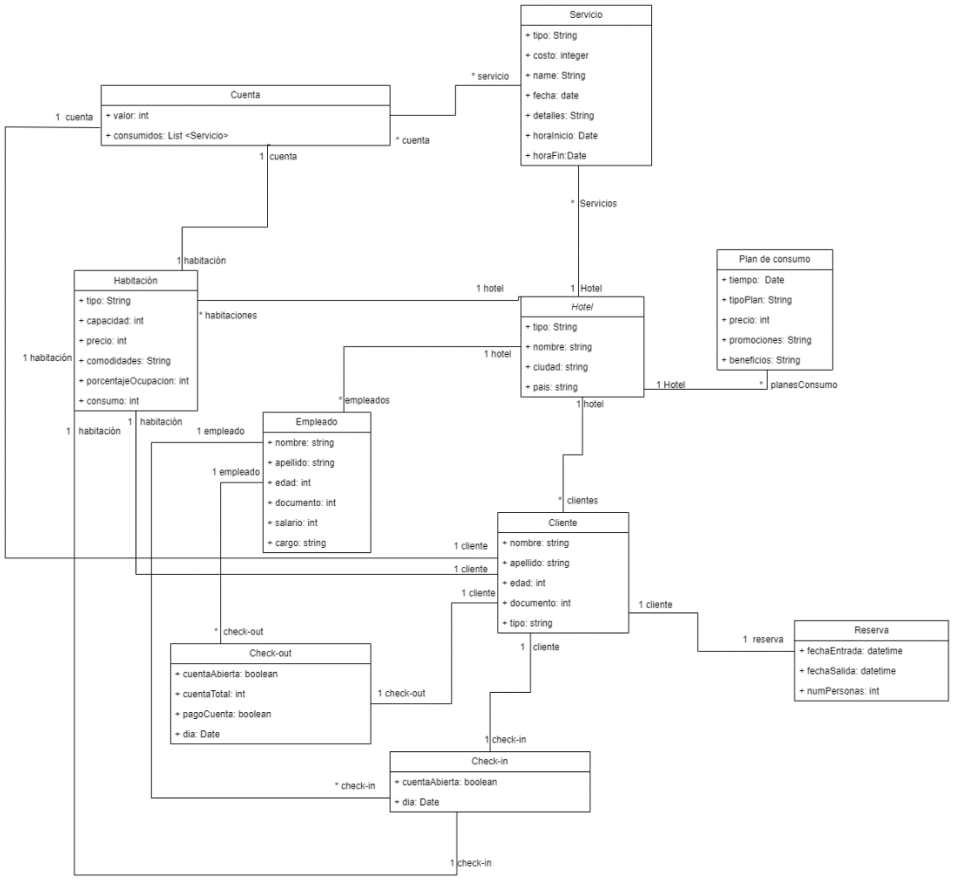
Integrantes:

Gabriela Soler – 202123744

Silvana Sandoval – 202123682

Santiago Celis – 202111131

UML



Diseño de la base de datos

Entidades y atributos (Cantidad de registros):

Entidad	Atributos	Registros
Hotel	tipo: String	3
	nombre: String	3
	ciudad: String	3
	país: String	3
Plan consumo	tiempo: date	9
	tipoPlan: String	9
	precio: int	9

	promociones: String	9
	beneficios: String	9
Servicio	tipo: String	9
	costo: Integer	9
	name: String	9
	fecha: Date	9
	detalles: String	9
	horaInicio: Date	9
	horaFin: Date	9
Cuenta	Valor: int	9
	Consumidos: List<Servicio>	27
Habitación	tipo: String	15
	numero: int	15
	capacidad: int	15
	precio: int	15
	comodidades: String	15
	porcentajeOcupacion: int	15
	consumo: int	15
Empleado	nombre: String	9
	apellido: String	9
	edad: int	9
	documento: int	9
	salario: int	9
	cargo: String	9
Cliente	nombre: String	9
	apellido: String	9
	edad: int	9
	documento: int	9
	tipo: String	9
Check-out	cuentaAbierta: boolean	9
	cuentaTotal: int	9
	pagoCuenta: boolean	9
	dia: date	9
Check-in	cuentaAbierta: boolean	9
	dia: date	9
Reserva	fechaEntrada: datetime	9
	fechaSalida: datetime	9
	numPersonas: int	9

Análisis de operaciones- Cuantificación operaciones

Entities	Operation	Information needed	Type	Rate

Habitación	Consultar un tipo de habitación.	Hotel.nombre + Habitación.id	Read	40 veces /día
Habitación	Añadir un tipo de habitación.	Hotel.nombre + Habitación.Id	Write	1vez/mes
Habitación	Consultar la información de una habitación.	Hotel.nombre + Habitación.Id	Read	80 veces/ día
Habitación	Crear una habitación.	Hotel.nombre	Write	1 vez/ mes
Servicio	Consultar un servicio.	Hotel.nombre + Servicio.name	Read	40 veces /día
Servicio	Añadir un nuevo servicio.	Hotel.nombre	Write	1 vez/ mes
Reserva, cliente	Consultar una reserva de alojamiento	Cliente.documeto	Read	40 veces /día
Reserva, cliente	Añadir una reserva de alojamiento	Cliente.documeto	Write	80 veces/ día
Reserva, cliente	Actualizar una reserva de alojamiento	Cliente.documeto	Write	20 veces/ día
Check-in, Cliente	Consultar la llegada de un cliente al hotel	Check-in	Read	10 veces/día
Check-in, Cliente	Añadir la llegada de un cliente al hotel	Check-in	Write	40 veces/ día
Cuenta, servicio, cliente	Consultar el consumo de un servicio por parte de un cliente	Cliente.documento + cuenta.id + servicio.nombre	Read	10 veces/día
Cuenta, servicio, cliente	Añadir el consumo de un servicio por parte de un cliente	Cliente.documento + cuenta.id + servicio.nombre	Write	80 veces/ día
Cuenta, servicio, cliente	Actualizar el consumo de un servicio por parte de un cliente	Cliente.documento + cuenta.id + servicio.nombre	Write	80 veces/ día

Check-out, Cliente	Consultar la salida de un cliente	Check-out	Read	10 veces/día
Check-out, Cliente	Añadir una nueva salida de un cliente	Check-out	Write	40 veces/día
Servicio, Habitación	Añadir el dinero recolectado por servicio en cada habitación	Servicio.nombre + Habitación.id + Servicio.costo	Write	80 veces/ día
Servicio, Habitación	Consultar el dinero recolectado por servicio en cada habitación	Servicio.nombre + Habitación.id + Servicio.costo	Read	10 veces/día
Habitación	Añadir el porcentaje de ocupación de una habitación	Habitación.porcentajeOcupacion	Write	5 veces /día
Habitación	Consultar el porcentaje de ocupación de una habitación	Habitacion.porcentajeOcupacion	Read	1 vez/día
Cliente, Check-in, Check-out	Consultar el consumo por un cliente en un rango de fechas	Check-in.dia + Check-out.dia + Cliente.id + Cuenta.consumidos	Read	10 veces/día
Cliente, Check-in, Check-out	Consultar aquellos clientes que realizan estancias en HotelAndes al menos una vez por trimestre.	Hoteles + Clientes	Read	12 veces/año

Lista de entidades

- Hotel: Esta entidad modela los distintos establecimientos que ofrecen alojamiento y servicios en varios países con diferentes categorías.

- Plan consumo: Esta entidad representa el conjunto de servicios y características de un paquete de alojamiento en un hotel.
- Servicio: Representa cualquier actividad o beneficio disponible en un hotel. Esta entidad puede referirse a servicios de habitación, limpieza, acceso a internet y otras áreas de recreación o instalaciones extras.
- Cuenta: Es la representación del estado financiero de un cliente. Esta entidad indica el consumo total del mismo y los servicios a los que destino recursos.
- Habitación: Esta entidad modela las características y tipo de una habitación.
- Empleado: Representa a las personas que trabajan en el hotel, también indica distintos datos de identificación y cargos dentro del establecimiento.
- Cliente: Modela la persona que se aloja en un hotel y usa los servicios de este. Esta entidad también modela a los acompañantes.
- Check-out: Esta entidad modela el proceso en el que se registra un cliente y sus acompañantes al ingresar a un hotel con su respectiva fecha.
- Check-in: Esta entidad modela el proceso de salida de un cliente y sus acompañantes, en esta se especifica el estado de la cuenta del cliente y la respectiva fecha de salida.
- Reserva: Representa la solicitud anticipada de adquirir los servicios de hospedaje en un hotel.

Relaciones entre entidades

1. Cuenta-Cliente: es una relación 1 a 1, lo cual significa que cada cliente tiene una única cuenta de consumo.
2. Cuenta-Habitación: es una relación 1 a 1, es decir, cada habitación tiene una cuenta.
3. Cuenta-servicios: es una relación muchos a muchos, es decir, que muchas cuentas pueden tener muchos servicios.
4. Servicio-Hotel: es una relación 1 a muchos, es decir, un hotel tiene muchos servicios.
5. Check_in-Habitación: es una relación 1 a 1, es decir, cada check-in tiene una sola habitación.
6. Check_in-Empleado: es una relación 1 a muchos, es decir, un empleado puede realizar muchos check-ins.
7. Check_in-Cliente: es una relación 1 a 1, es decir, cada check-in tiene a un cliente asociado.
8. Hotel-Plan de Consumo: relación 1 a muchos, es decir, cada hotel tiene diferentes planes de consumo disponibles.
9. Hotel-Cliente: es una relación 1 a muchos, lo que quiere decir que un hotel tiene muchos clientes.
10. Hotel-Habitación: es una relación 1 a muchos, es decir, un hotel tiene muchas habitaciones.
11. Hotel-Empleado: es una relación 1 a muchos, lo que significa que un hotel tiene muchos empleados.
12. Cliente-Reserva: es una relación 1 a 1, es decir, un cliente tiene una única reserva.

13. Cliente-Check_out: es una relación 1 a 1, lo que quiere decir que un cliente tiene un check-out asociado.
14. Cliente-Habitación: es una relación 1 a 1, es decir, un cliente tiene una única habitación asociada.
15. Empleado-Check_out: es una relación uno a muchos, pues un empleado puede realizar múltiples check-outs.

Análisis de selección de esquema de asociación

- Cuenta-Cliente: Referenciado

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

- Cuenta-Habitación: Referenciado

Guideline name	Question	Embedded	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a “has-a”, “contains”, or a similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are there pieces of information updated together?	No	No

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

- Cuenta-servicios: Referenciado

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

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

- Servicio-Hotel: Embebido

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

- Check_in-Habitación: Referenciado

Guideline name	Question	Embedded	Reference

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

- Check_in-Empleado: Referenciado

Guideline name	Question	Embedded	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	No	Yes
Go Together	Do the pieces of information have a “has-a”, “contains”, or a similar relationship?	No	No

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

- Check_in-Cliente: Embebido

Guideline name	Question	Embedded	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	Yes
Go Together	Do the pieces of information have a “has-a”, “contains”, or a similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are there pieces of information updated together?	Yes	No
Archival	Should the pieces of information be archived at the same time?	Yes	Yes
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	No
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	No

Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	No
Document Growth	Would the embedded piece grow without bound?	Yes	No
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes

- Hotel-Plan de Consumo: Embebido

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

- Hotel-Cliente: Referenciado

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

- Hotel-Habitación: Embebido

Guideline name	Question	Embedded	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a “has-a”, “contains”, or a similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are there pieces of information updated together?	Yes	No

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

- Hotel-Empleado: Referenciado

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

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

- Cliente-Reserva: Embebido

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

- Cliente-Check_out: Embebido

Guideline name	Question	Embedded	Reference

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

- Cliente-Habitación: Referenciado

Guideline name	Question	Embedded	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	No	Yes
Go Together	Do the pieces of information have a “has-a”, “contains”, or a similar relationship?	No	Yes
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are there pieces of information updated together?	No	No
Archival	Should the pieces of information be archived at the same time?	No	No

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

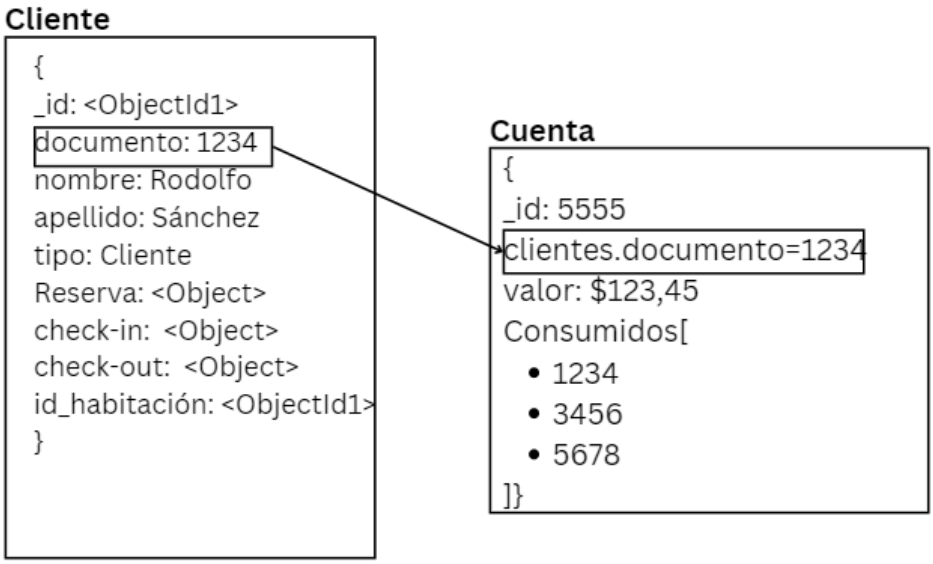
- Empleado-Check_out: Referenciado

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

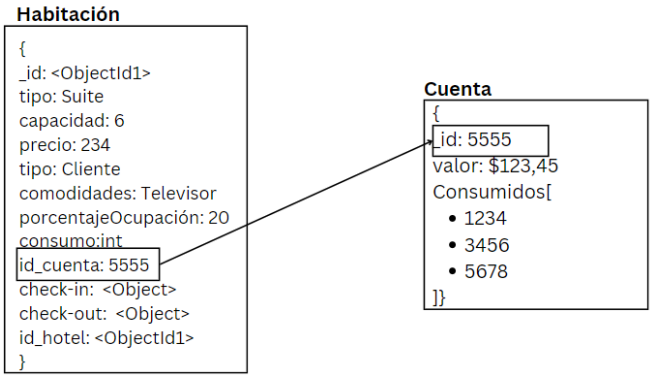
Workload	Are the pieces of information written at different times in a write-heavy workload?	Yes	No
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes

Descripción gráfica

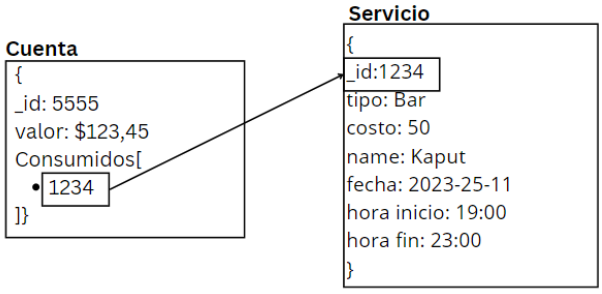
1. Cuenta-Cliente: Referenciado



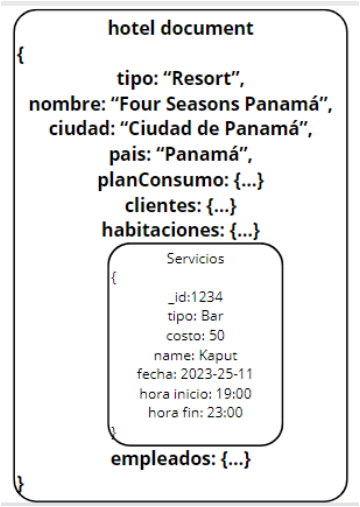
2. Cuenta-Habitación: Referenciado



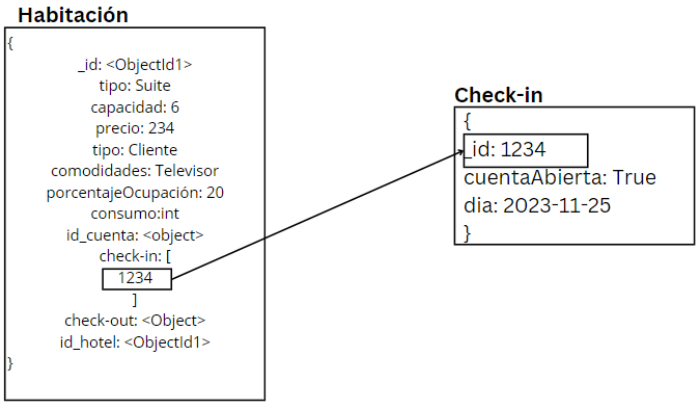
3. Cuenta-servicios: Referenciado



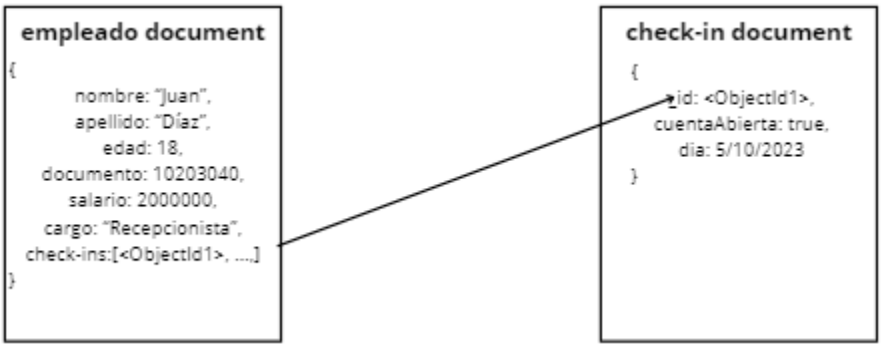
4. Servicio-Hotel: Embebido



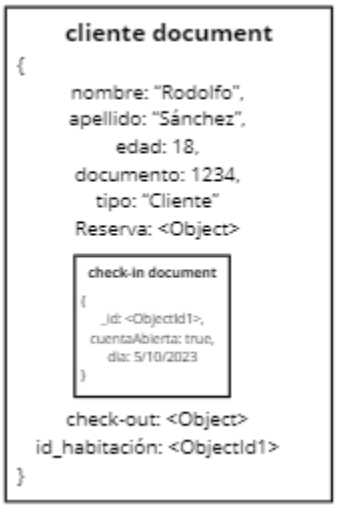
5. Check_in-Habitación: Referenciado



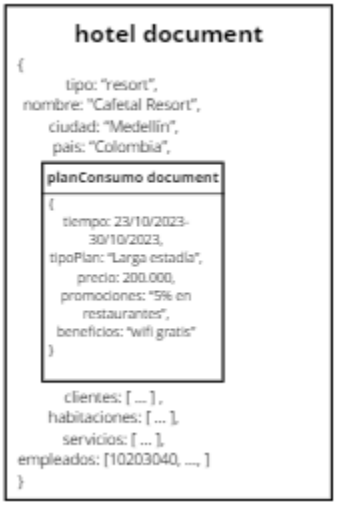
6. Check_in-Empleado: Referenciado



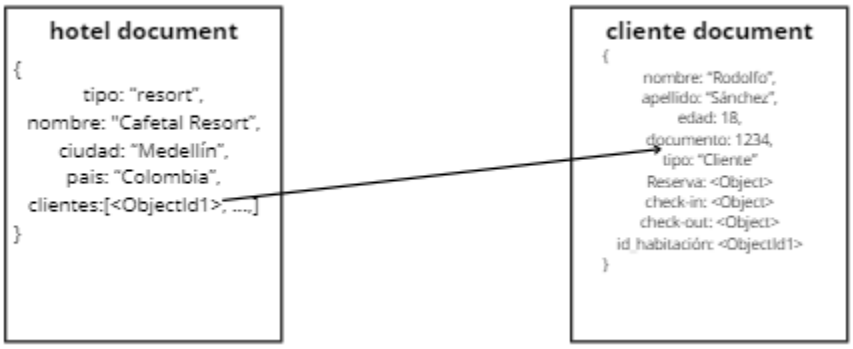
7. Check_in-Cliente: Embebido



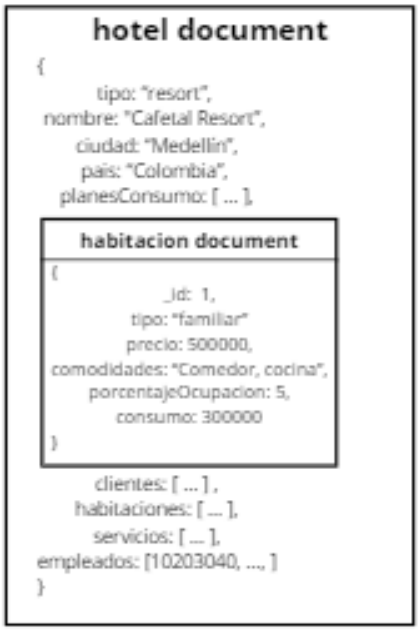
8. Hotel-Plan de Consumo: Embebido



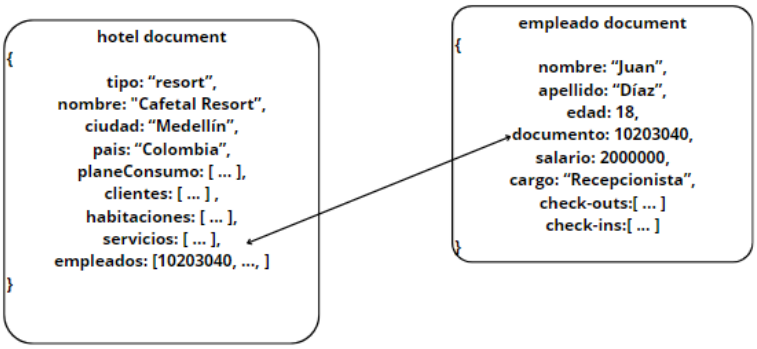
9. Hotel-Cliente: Referenciado



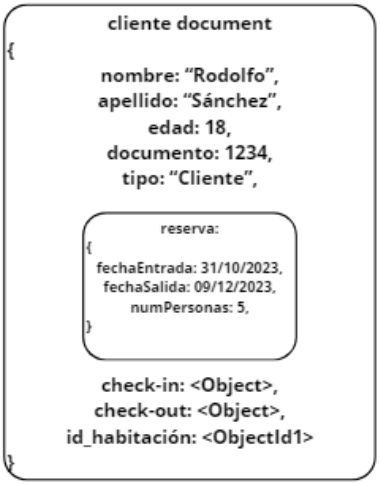
10. Hotel-Habitación: Embebido



11. Hotel-Empleado: Referenciado

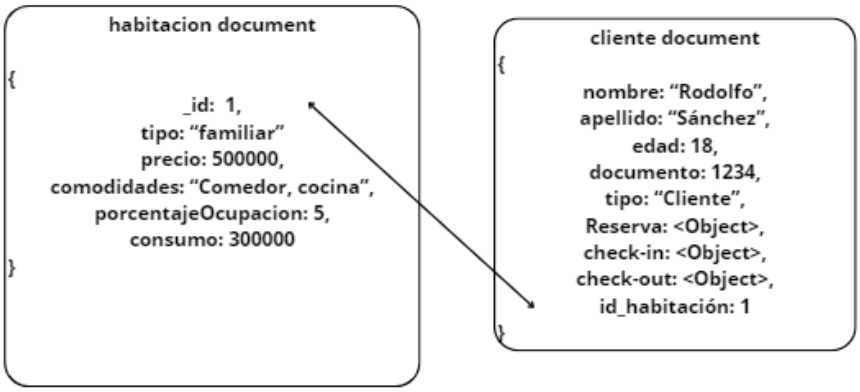


12. Cliente-Reserva: Embebido

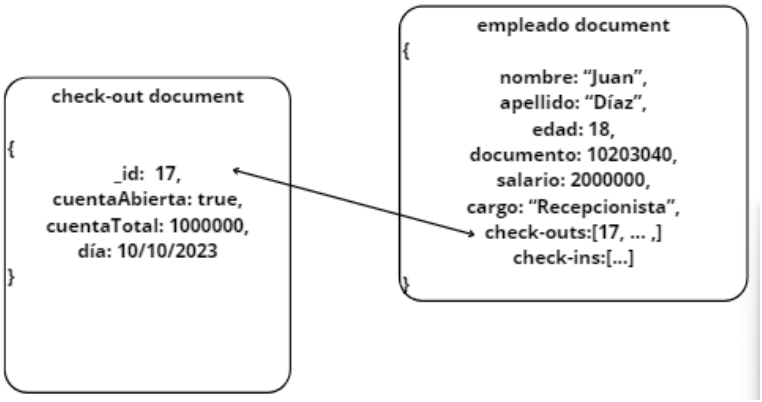


13. Cliente-Check_out: Embebido

14. Cliente-Habitación: Referenciado



15. Empleado-Check_out: Referenciado



Requerimientos

Funcionales:

RF1:

- Insertar un tipo de habitación

```
db.Hoteles.updateOne(
  {nombre: "Marriot Bogota", "habitaciones.numero": 1 },
  {$set: {"habitaciones.$.tipo": "suite" } }
);
```

- Encontrar el tipo de habitación

```
db.Hoteles.findOne(
  {nombre: "Marriot Bogota", "habitaciones": { $elemMatch: { numero: 1 } } },
  { id: 0, "habitaciones.$": 1 }
);
```

- Actualizar el tipo de habitación

```
db.Hoteles.updateOne(
  {nombre: "Marriot Bogota", "habitaciones.numero": 1 },
  {$set: {"habitaciones.$.tipo": "suite" } }
);
```

- Borrar un tipo de habitación

```
db.Hoteles.updateOne(
  {nombre: "Marriot Bogota"},
  {$pull: {habitaciones: {tipo: "doble"}}}
);
```

RF2:

- Insertar una habitación

```
db.Hoteles.updateOne(
  { nombre: "Marriot Bogota" },
  {
    $push: {
      habitaciones: {
        numero: 6,
```

```
        tipo: "familiar",
        precio: 460000,
        comodidades: "Comedor, cocina",
        porcentajeOcupacion: 8,
        consumo: 300000
      }
    }
  }
);
```

- Encontrar una habitación

```
db.Hoteles.find(
  { nombre: "Marriot Bogota", "habitaciones.numero": 1 },
  { _id: 0, "habitaciones.$": 1 }
);
```

- Actualizar una habitación

```
db.Hoteles.updateOne(
  {nombre: "Marriot Cancun", "habitaciones.numero": 5 },
  {$set: {"habitaciones.$.precio": 67000 } }
);
```

- Borrar una habitación

```
db.Hoteles.updateOne(
  {nombre: "Marriot Bogota"},
  {$pull: {habitaciones: {numero: 5}}});
```

RF3:

- Insertar un servicio del hotel

```
db.Hoteles.updateOne(
  { nombre: "Four Seasons Panama" },
  {
```

```
    $push: {
      servicios: {
        id: 6,
        tipo: "Restaurante",
        costo: 80,
        nombre: "Corral",
        fecha: "2023-25-11",
        horaInicio: "19:00",
        horaFin: "23:00"
      }
    }
  }
);
```

- Encontrar un servicio del hotel

```
db.Hoteles.find({nombre: "Four Seasons Panama", servicios:{$elemMatch:{id: 5}}}, {nombre: 1, "servicios.$" :1});
```

- Actualizar un servicio del hotel

```
db.Hoteles.updateOne(
  {nombre: "Four Seasons Panama", "servicios.id": 5 },
  {$set: {"servicios.$.costo": 100 } }
);
```

- Borrar un servicio del hotel

```
db.Hoteles.updateOne(
  {nombre: "Four Seasons Panama"},
  {$pull: {servicios: {id: 5}}}
);
```

RF4:

- **Insertar una reserva**

db.clientes.updateOne({documento: 234567890}, {$push: { reservas:{ "_id": 1, "fechaEntrada": "2023-12-03", "fechaSalida": "2023-12-07", "numPersonas": 10 }}})

- **Actualizar reserva**

  db.clientes.updateOne({documento: 234567890, "reservas._id": 1}, {$set:{"reservas.$.fechaEntrada": "2023-12-05"}})

  db.clientes.updateOne({documento: 234567890, "reservas._id": 1}, {$set:{“reservas.$.fechaSalida”: "2023-12-10"}})

  db.clientes.updateOne({documento: 234567890, "reservas._id": 1}, {$set:{“reservas.$.numPersonas”: 7}})

- **Borrar reserva**

  db.clientes.updateOne({documento: 1000987653}, {$unset: {“reservas”:1 }})

- **Encontrar una reserva**

  db.clientes.find(

  { "documento": 1000987653, "reservas._id":1 },{"reservas.$": 1}

  );

RF5:

- **Insertar un consumo de un servicio del hotel**

  Db.servicios.insertOne({

  “_id”: 1234,

  “tipo”: “Bar”,

  “costo”: 50,

  “nombre”: Kaput,

  “fecha”: 2023-11-25,

  “horaInicio”: 19:00,

  “horaFin”: 23:00

  })

  db.cuentas.updateOne({“clientes.documento”: 1000987653}, {$push: { servicios: 1234}}})

- **Borrar un consumo de un servicio del hotel**

  db.cuentas.updateOne({documento: 1000987653}, {$unset: {“servicios”: 1234}})

- **Encontrar un consumo de un servicio del hotel**

  db.cuentas.findOne ({documento: 1000987653}, {“servicios.$”: 1234})

RF6:

- **Insertar un checkin**

  db.clientes.updateOne({documento: 1000987653}, {$push: { checkins:{

  “_id”: 1,

  “cuentaAbierta”: true,

  “dia”: 2023-12-07

  }}})

- **Actualizar checkin**

  db.clientes.updateOne({documento:1000237894}, {$set:{“checkin.cuentaAbierta”: false}})

- **Borrar checkin**

  db.Clientes.updateOne(

  { "documento": 1112223334},

  { $unset: { "checkin": 1 } }

  );

- **Encontrar un checkin**

  db.clientes.find({documento: 1000987653, “checkins.id”: 18})

RF7:

- **Insertar un checkout**

```
db.Clientes.updateOne(
  { "documento": 1234593 },
  {
    $set: {
      "checkout": {
        "_id": 17,
        "cuentaAbierta": true,
```

```
        "cuentaTotal": 1850000,
        "dia": "2023-05-20"
      }
    }
  }
);
```

- **Actualizar un checkout**

```
db.Clientes.updateOne(
  { "documento": 185739900 },
  { $set: { "checkout.cuentaTotal": 1500000 } }
);
```

- **Eliminar un checkout**

```
db.Clientes.updateOne(
  { "documento": 1112223334},
  { $unset: { "checkout": 1 } }
);
```

- **Consultar un checkout**

```
db.Clientes.find(
  { "documento": 1000987653, "checkout.id": 15 },
  { "checkout.$": 1 }
);
```

Consultas básicas

RF1:

```
db.clientes.aggregate([
  {
    $match: {
      "reserva.fechaEntrada": {
```

```
            $lte: ISODate("2023-12-15T00:00:00.000Z"),
            $gte: ISODate("2023-02-05T00:00:00.000Z")
          },
          "reserva.fechaSalida": {
            $gte: ISODate("2023-02-05T00:00:00.000Z"),
            $lte: ISODate("2023-12-15T00:00:00.000Z")
          }
        }
      },
      {
        $project: {
          id_habitacion: 1,
          cuentaTotal: "$checkout.cuentaTotal"
        }
      },
      {
        $group: {
          _id: "$id_habitacion",
          totalCuenta: { $sum: "$cuentaTotal" }
        }
      }
]);
```

RF2:

```
db.clientes.aggregate([
  {
    $lookup: {
      from: "hoteles",
      localField: "id_habitacion",
```

```
            foreignField: "habitaciones.numero",
            as: "listaHabitacionesClientes"
        }
    },
    {
        $match: {
            "reserva.fechaEntrada": {
                $lte: ISODate("2023-12-15T00:00:00.000Z"),
                $gte: ISODate("2023-02-05T00:00:00.000Z")
            },
            "reserva.fechaSalida": {
                $gte: ISODate("2023-02-05T00:00:00.000Z"),
                $lte: ISODate("2023-12-15T00:00:00.000Z")
            }
        }
    },
    {
        $project: {
            "habitaciones.numero": 1,
            "habitaciones.porcentajeOcupacion": 1
        }
    }
]);
```

```
db.hoteles.find({"habitaciones.porcentajeOcupacion": {$gte: 0, $lte: 100}},
{"habitaciones.numero":1, "habitaciones.porcentajeOcupacion":1})
```

RF3:

```
db.clientes.aggregate([
  {
```

```
    $match: {
      "reserva.fechaEntrada": {
        $lte: ISODate("2023-12-15T00:00:00.000Z"),
        $gte: ISODate("2023-02-05T00:00:00.000Z")
      },
      "reserva.fechaSalida": {
        $gte: ISODate("2023-02-05T00:00:00.000Z"),
        $lte: ISODate("2023-12-15T00:00:00.000Z")
      }
    }
  },
  {
    $project: {
      documento: 1,
      cuentaTotal: "$checkout.cuentaTotal"
    }
  },
  {
    $group: {
      _id: "$documento",
      totalCuenta: { $sum: "$cuentaTotal" }
    }
  }
]);
```

Consultas avanzadas

```
db.Clientes.aggregate([{$lookup: {from: “Cuentas”, localField: “documento”, foreignField: “clienteDocumento”, as: “cuentaCliente”}}, {$match: {"reserva.fechaEntrada": {
        $lte: ISODate("2023-12-15T00:00:00.000Z"),
        $gte: ISODate("2023-02-05T00:00:00.000Z")
```

```
        },
        "reserva.fechaSalida": {
           $gte: ISODate("2023-02-05T00:00:00.000Z"),
           $lte: ISODate("2023-12-15T00:00:00.000Z")
        },
        {
        “$cuentaCliente.consumidos”: {$not: 2}}
},
{
$project: {nombre: 1}
}
]);
```