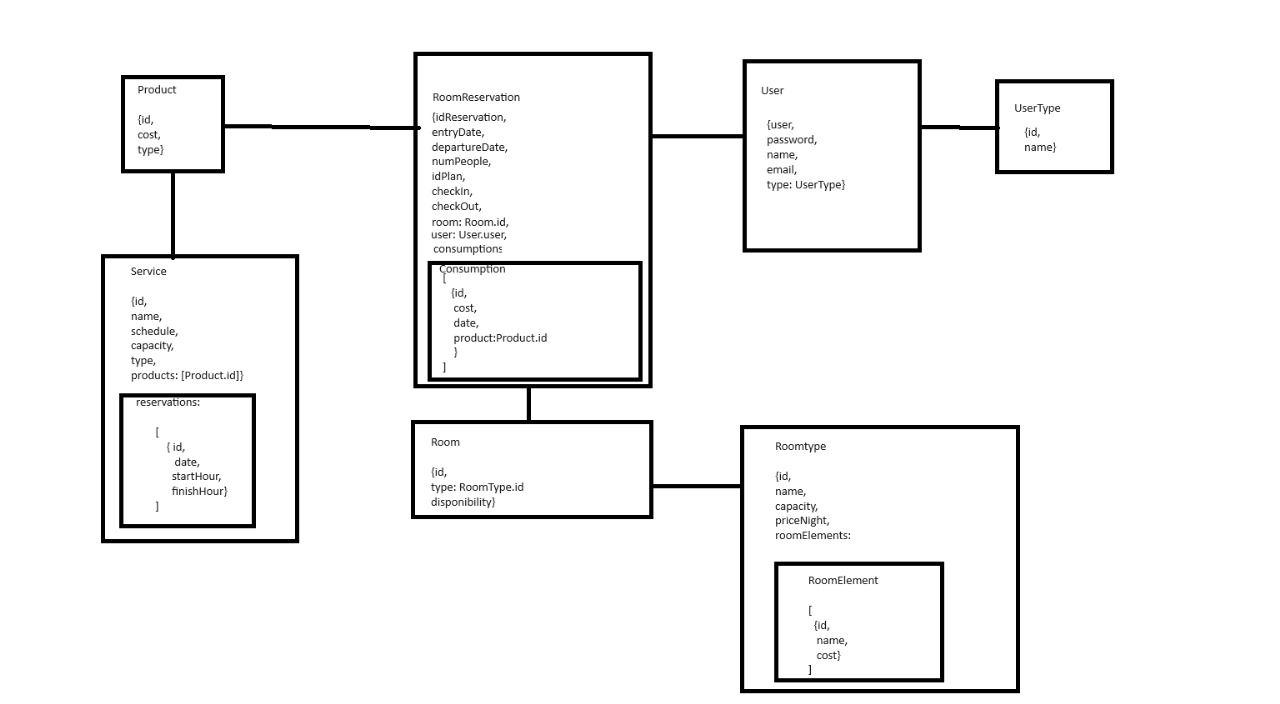
**Proyecto Caso de Estudio E3**

**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.



**Diseño de la base de datos**

1. Análisis de la carga de trabajo (workload). Para ello, presenten lo siguiente:
2. Identifiquen entidades y sus atributos

1. Product:

   - id

   - cost

   - type

2. Service:

   - id

   - name

   - schedule

   - capacity

   - type

   - products: [Product.id]

3. RoomReservation:

   - idReservation

   - entryDate

   - departureDate

   - numPeople

   - plan

   - checkIn

   - checkOut

   - room: Room.id

   - user: User.user

   - consumptions

4. Consumption:

   - id

   - cost

   - date

   - product: Product.id

5. Room:

   - id

   - type: RoomType.id

   - disponibility

6. RoomType:

   - id

   - name

   - capacity

   - priceNight

   - roomElements

7. RoomElement:

   - id

   - name

   - cost

8. User:

   - user

   - password

   - name

   - email

   - type: UserType

9. UserType:

   - id

   - name

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 | Cantidad de registros esperados |
| RoomType | 20 |
| Room | 200 |
| Service | 35 |
| RoomReservation | 50000 / 3 años |
| RoomReservation (Checkin/Checkout) | 50000 / 3 años |
| Consumption | 250000 / 3 años |

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 qué información se accederá de manera conjunta.

|  |  |  |  |
| --- | --- | --- | --- |
| Entidades | Operaciones | Información necesaria | Tipo |
| RoomType | R | N/A | Lectura |
| RoomType | C,U,D | Detalles del tipo de habitación | Escritura |
| Room | R | N/A | Lectura |
| Room | C,U,D | Detalles de la habitación | Escritura |
| Room | Ver ocupaciónes | Reservas que tenga la habitación | Lectura |
| Service | R | N/A | Lectura |
| Service | C,U,D | Detalles servicio + productos | Escritura |
| Service | Clientes que consumen | Detalles de clientes que consumen el servicio | Lectura |
| RoomReservation | R | N/A | Lectura |
| RoomReservation | C,U,D | Detalles reserva + habitación | Escritura |
| Consumption | R | N/A | Lectura |
| Consumption | C,U,D | Detalles consumo | Escritura |
| Consumption | Consumos de un cliente | Detalles consumo de ese cliente | Lectura |
| RoomReservation | R | Detalles de checkin/checkout sobre reserva | Lectura |
| RoomReservation | C,U,D | Detalles de checkin/checkout sobre reserva | Escritura |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entidades | Operaciones | Información necesaria | Tipo | Tasa |
| RoomType | R | Ninguna | Lectura | 1/semana |
| RoomType | C,U,D | Detalles del tipo de habitación | Escritura | 1/semana |
| Room | R | Ninguna | Lectura | 2/semana |
| Room | C,U,D | Detalles de la habitación | Escritura | 2/semana |
| Room | Ver ocupación | Reservas que tenga la habitación | Lectura | N/A |
| Service | R | Ninguna | Lectura | 2/semana |
| Service | C,U,D | Detalles servicio + productos | Escritura | 2/semana |
| Service | Clientes que consumen | Detalles de clientes que consumen el servicio | Lectura |  |
| RoomReservation | R | Ninguna | Lectura | 50/día |
| RoomReservation | C,U,D | Detalles reserva + habitación | Escritura | 100/día |
| Consumption | R | Ninguna | Lectura | 250/día |
| Consumption | C,U,D | Detalles consumo | Escritura | 500/día |
| Consumption | Consumos de un cliente | Detalles consumo de ese cliente | Lectura |  |
| RoomReservation (Checkin y Checkout ) | R | Detalles de checkin/checkout sobre reserva | Lectura | 50/día |
| RoomReservation (Checkin y Checkout ) | C,U,D | Detalles de checkin/checkout sobre reserva | Escritura | 100/día |

2. 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.

Product: Representa bienes o artículos que el hotel ofrece a los huéspedes. Cada producto tiene un costo asociado y un tipo que lo categoriza.

Service: Esta entidad representa los servicios que el hotel ofrece a sus clientes, como la piscina o el gimnasio. Incluye detalles como horarios, capacidad y tipos de servicios, así como una relación con los productos que se pueden requerir para la prestación del servicio.

RoomReservation: Encapsula toda la información relacionada con la reserva de una habitación. Esto incluye la fecha de entrada y salida, el número de personas, la información del plan seleccionado, y cualquier consumo de productos que el huésped haya hecho durante su estancia.

Consumption: Relacionado con los productos consumidos por los huéspedes, esta entidad lleva un registro de los costos e identifica qué productos se han consumido, asociándolos con una reserva específica.

Room: Representa una habitación individual en el hotel y está asociada con un tipo específico de habitación. También incluye información sobre la disponibilidad de la habitación.

RoomType: Categoriza las habitaciones por tipo, como suite presidencial, doble, o sencilla. Incluye atributos como la capacidad, el precio por noche y los elementos específicos de la habitación que definen su categoría y comodidades.

RoomElement: Son los elementos o características individuales que pueden estar presentes en una habitación, como una televisión, minibar o cafetera, cada uno con su propio costo.

User: Representa a una persona que utiliza el sistema de reservas del hotel, ya sea un huésped o un empleado. Incluye información de autenticación y contacto.

UserType: Define el rol o tipo de usuario dentro del sistema, lo que puede ayudar a determinar los niveles de acceso y permisos dentro del sistema de gestión del hotel.

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

Product (1:M) Service: Una relación de muchos a muchos indicando que un servicio puede involucrar (vender) varios productos, y un producto solo se vende en algún servicio.

Service (1:M) Reservations: Una relación de uno a muchos indicando que un servicio puede tener varias reservaciones.

RoomReservation (N:1) Room: Una relación de muchos a uno indicando que muchas reservaciones pueden estar asociadas con una sola habitación.

RoomReservation (N:1) User: Una relación de muchos a uno indicando que un usuario puede tener muchas reservaciones.

RoomReservation (1:N) Consumption: Una relación de uno a muchos indicando que una reservación puede tener múltiples consumos registrados.

Room (N:1) RoomType: Una relación de muchos a uno indicando que varias habitaciones pueden pertenecer a un solo tipo de habitación.

RoomType (1:N) RoomElement: Una relación de uno a muchos indicando que un tipo de habitación puede incluir varios elementos de habitación.

User (N:1) UserType: Una relación de muchos a uno indicando que varios usuarios pueden tener un solo tipo de usuario.

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.

RoomType - Room

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Guia | Pregunta | Embeber (yes/no) | Referenciar  (yes/no) |
| Simplicity | Would keeping the pieces of information together lead to a simpler data modeland code ? | Yes | No |
| Go Together | Do the pieces of information have a “has-a”, “contains” or similar relationship ? | Yes | No |
| Query Atomicity | Does the application query the pieces of information together ? | Yes | Yes |
| Update Complexity | Are the pieces of information updated together ? | Yes | Yes |
| 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? | No | No |
| Document Size | Would the combined size of the pieces of information take too much memory or transfer bondwith for the application ? | Yes | No |
| Document Growth | Would the embedded pieces 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 ? | Yes | Yes |

Room - RoomReservation

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Guia | Pregunta | Embeber (yes/no) | Referenciar  (yes/no) |
| Simplicity | Would keeping the pieces of information together lead to a simpler data modeland code ? | Yes | No |
| Go Together | Do the pieces of information have a “has-a”, “contains” or similar relationship ? | Yes | No |
| Query Atomicity | Does the application query the pieces of information together ? | Yes | Yes |
| Update Complexity | Are the pieces of information updated together ? | Yes | Yes |
| 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? | No | No |
| Document Size | Would the combined size of the pieces of information take too much memory or transfer bondwith for the application ? | Yes | No |
| Document Growth | Would the embedded pieces grow without bound ? | Yes | Yes |
| 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 ? | Yes | Yes |

Reservations-Services

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Guia | Pregunta | Embeber (yes/no) | Referenciar  (yes/no) |
| Simplicity | Would keeping the pieces of information together lead to a simpler data modeland code ? | Yes | No |
| Go Together | Do the pieces of information have a “has-a”, “contains” or similar relationship ? | Yes | No |
| Query Atomicity | Does the application query the pieces of information together ? | Yes | Yes |
| Update Complexity | Are the pieces of information updated together ? | Yes | Yes |
| 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? | No | No |
| Document Size | Would the combined size of the pieces of information take too much memory or transfer bondwith for the application ? | Yes | No |
| Document Growth | Would the embedded pieces 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 ? | Yes | Yes |

RoomsReservations-Consumptions

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Guia | Pregunta | Embeber (yes/no) | Referenciar  (yes/no) |
| Simplicity | Would keeping the pieces of information together lead to a simpler data modeland code ? | No | No |
| Go Together | Do the pieces of information have a “has-a”, “contains” or similar relationship ? | No | No |
| Query Atomicity | Does the application query the pieces of information together ? | No | No |
| Update Complexity | Are the 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? | No | No |
| Document Size | Would the combined size of the pieces of information take too much memory or transfer bondwith for the application ? | Yes | No |
| Document Growth | Would the embedded pieces grow without bound ? | Yes | Yes |
| 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 ? | Yes | Yes |

Consumptions-Products

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Guia | Pregunta | Embeber (yes/no) | Referenciar  (yes/no) |
| Simplicity | Would keeping the pieces of information together lead to a simpler data modeland code ? | Yes | No |
| Go Together | Do the pieces of information have a “has-a”, “contains” or similar relationship ? | Yes | No |
| Query Atomicity | Does the application query the pieces of information together ? | Yes | Yes |
| Update Complexity | Are the pieces of information updated together ? | Yes | Yes |
| 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? | No | No |
| Document Size | Would the combined size of the pieces of information take too much memory or transfer bondwith for the application ? | Yes | No |
| Document Growth | Would the embedded pieces 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 |

Services-Reservations

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Guia | Pregunta | Embeber (yes/no) | Referenciar  (yes/no) |
| Simplicity | Would keeping the pieces of information together lead to a simpler data modeland code ? | Yes | No |
| Go Together | Do the pieces of information have a “has-a”, “contains” or similar relationship ? | Yes | No |
| Query Atomicity | Does the application query the pieces of information together ? | Yes | Yes |
| Update Complexity | Are the pieces of information updated together ? | Yes | Yes |
| 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? | No | No |
| Document Size | Would the combined size of the pieces of information take too much memory or transfer bondwith for the application ? | Yes | No |
| Document Growth | Would the embedded pieces 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 ? | Yes | Yes |

Users-RoomsReservations

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Guia | Pregunta | Embeber (yes/no) | Referenciar  (yes/no) |
| Simplicity | Would keeping the pieces of information together lead to a simpler data modeland code ? | Yes | No |
| Go Together | Do the pieces of information have a “has-a”, “contains” or similar relationship ? | Yes | No |
| Query Atomicity | Does the application query the pieces of information together ? | Yes | Yes |
| Update Complexity | Are the pieces of information updated together ? | Yes | Yes |
| 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? | No | No |
| Document Size | Would the combined size of the pieces of information take too much memory or transfer bondwith for the application ? | Yes | No |
| Document Growth | Would the embedded pieces 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 ? | Yes | Yes |

RoomElements-RoomTypes

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Guia | Pregunta | Embeber (yes/no) | Referenciar  (yes/no) |
| Simplicity | Would keeping the pieces of information together lead to a simpler data modeland code ? | Yes | No |
| Go Together | Do the pieces of information have a “has-a”, “contains” or similar relationship ? | Yes | No |
| Query Atomicity | Does the application query the pieces of information together ? | Yes | Yes |
| Update Complexity | Are the pieces of information updated together ? | Yes | Yes |
| 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? | No | No |
| Document Size | Would the combined size of the pieces of information take too much memory or transfer bondwith for the application ? | Yes | No |
| Document Growth | Would the embedded pieces 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 ? | Yes | Yes |

RoomTypes-Rooms

Embeber: La información sobre el tipo de habitación y la habitación misma se mantiene junta para facilitar la lectura y reducir la necesidad de realizar múltiples consultas para obtener información completa sobre una habitación.

Rooms-Reservations

Embeber: Al embeber la información de la reserva en la habitación, se facilita la obtención de detalles de la reserva junto con la información de la habitación, lo que mejora la eficiencia de las consultas comunes.

Reservations-Services

Embeber: Al embeber la información del servicio en la reserva, se puede acceder fácilmente a los detalles del servicio relacionado con la reserva sin la necesidad de realizar consultas adicionales.

RoomsReservations-Consumptions

Referenciar: La relación entre reservaciones y consumos se referenció para evitar la duplicación de datos y reducir la complejidad en caso de cambios en los consumos asociados a múltiples reservaciones.

Consumptions-Products

Embeber: Al embeber la información del producto en el consumo, se garantiza que la información completa sobre un consumo esté disponible sin necesidad de buscar en otra colección.

Services-Reservations

Embeber: Al embeber la información de la reserva en el servicio, se facilita la obtención de detalles de la reserva relacionada con un servicio sin realizar consultas adicionales.

Users-RoomsReservations

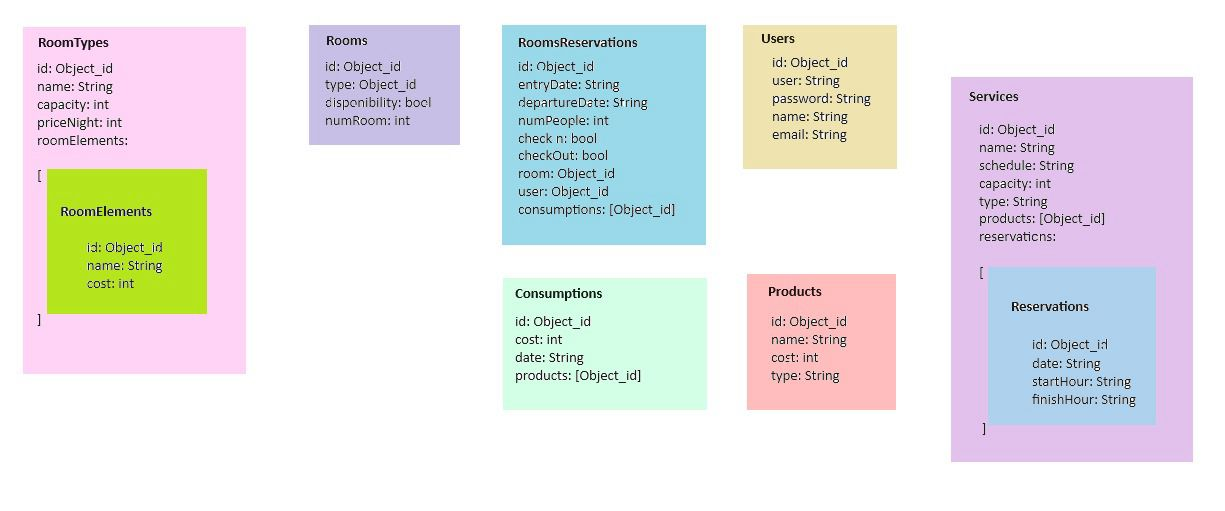
Embeber: Al embeber la información de la reservación en el usuario, se permite un acceso más rápido a las reservas asociadas a un usuario sin necesidad de realizar consultas adicionales.

RoomElements-RoomTypes

Embeber: La información de los elementos de la habitación se embebe en el tipo de habitación para simplificar la obtención de detalles completos sobre un tipo de habitación, evitando consultas adicionales.

Estas decisiones se basan en consideraciones como la simplicidad del modelo de datos, la eficiencia de las consultas y la gestión de la duplicación de datos.

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.



{

  "RoomTypes": {

"id": "RT001",

"name": "Suite Presidencial",

"capacity": 4,

"priceNight": 300,

"roomElements": ["RE001", "RE002", "RE003"]

  },

  "Rooms": {

"id": "RM001",

"type": "RT001",

"disponibility": true,

"numRoom": 101

  },

  "RoomsReservations": {

"id": "RR001",

"entryDate": "2023-12-01",

"departureDate": "2023-12-05",

"numPeople": 2,

"checkIn": true,

"checkOut": false,

"room": "RM001",

"user": "US001",

"consumptions": ["CN001", "CN002"]

  },

  "Consumptions": {

"id": "CN001",

"cost": 50,

"date": "2023-12-02",

"products": ["PD001", "PD002"]

  },

  "Products": {

"id": "PD001",

"name": "Agua Mineral",

"cost": 3,

"type": "Bebida"

  },

  "Services": {

"id": "SV001",

"name": "Gimnasio",

"schedule": "06:00-22:00",

"capacity": 20,

"type": "Fitness",

"products": ["PD003", "PD004"],

"reservations": ["RS001"]

  },

  "Reservations": {

"id": "RS001",

"date": "2023-12-02",

"startHour": "09:00",

"finishHour": "11:00"

  },

  "Users": {

"id": "US001",

"user": "juan\_velez",

"password": "password",

"name": "Juan Velez",

"email": "juan.velez@hotmail.com"

  },

  "RoomElements": {

"id": "RE001",

"name": "Televisión",

"cost": 1000

  }

}

Documentacion de los requerimientos:

RF1 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR TIPO DE HABITACIÓN

CRUD de los tipos de habitaciones de los cuales dispone el hotel con su dotación correspondiente. Considere inicialmente por lo menos 3 tipos de habitaciones.

// CREATE

    RoomType save(RoomType tipoHabitacion);

// READ

    @Query("{ 'name' : ?0 }")

    RoomType findByName(String name);

    @Query("{}")

    List<RoomType> findAllRoomTypes();

// DELETE

    @Query(value = "{ 'name' : ?0 }", delete = true)

    void deleteByName(String name);

    @Query(value = "{ '\_id' : ?0 }", delete = true)

    void deleteById(String id);

    @Query("{\_id: ?0}")

    @Update("{$push:{roomElements:{name:?1, cost:?2}}}")

    void updateById(String id\_roomType, String name, Integer cost);

RF2 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR HABITACIÓN

CRUD de las instancias de las habitaciones de las cuales dispone el hotel. Considere inicialmente por lo menos 3 habitaciones de cada tipo de habitación disponible

// CREATE

       Room save(Room room);

       // READ

       @Query("{ 'numRoom' : ?0 }")

       Room findByNumero(Integer numRoom);

       @Query("{ '\_id' : ?0 }")

        Optional<Room> findById(String id);

       @Query("{}")

       List<Room> findAllRooms();

       // DELETE

       @Query(value = "{ '\_id' : ?0 }", delete = true)

       void deleteByNumero(String id);

       // UPDATE

      @Query("{ 'numRoom' : ?0 }")

       Room updateRoom(int numRoom, Room room);

RF3 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR UN SERVICIO DEL HOTEL

CRUD de la descripción de los servicios de los cuales dispone el hotel. Considere inicialmente por lo menos 1 servicio de cada uno de los tipos de servicio descritos en el enunciado (piscina, spa, bar, … El menú de los bares y restaurantes debe tener por lo menos tres productos).

//CREATE

    Service save(Service service);

    // READ

    @Query("{ '\_id' : ?0 }")

    Service findByid(String id);

    @Query("{}")

    List<Service> findAllServices();

    // DELETE

    @Query(value = "{ '\_id' : ?0 }", delete = true)

    void deleteByid(String id);

    //UPDATE

    @Query("{ '\_id' : ?0 }")

    Service updateServicio(String id, Service service);

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

// CREATE

    RoomReservation save(RoomReservation roomReservation);

    // READ

    @Query("{}")

    List<RoomReservation> findByRoomsReservations();

    // Buscar por fecha de salida

    @Query("{ 'departureDate' : ?0 }")

    List<RoomReservation> findByDepartureDate(String departureDate);

    // Buscar por fecha de entrada

    @Query("{ 'entryDate' : ?0 }")

    List<RoomReservation> findByEntryDate(String entryDate);

    // Buscar por fecha de entrada y salida

    @Query("{ 'entryDate' : { $gte: ?0 }, 'departureDate' : { $lte: ?1 }}")

    List<RoomReservation> findByDateRange(String startDate, String endDate);

    // Buscar reservaciones por n usuario

    @Query("{ '\_id' : ?0 }")

    RoomReservation findByRoomReservation(String idReservation);

    // Buscar reservaciones por nombre de usuario

    @Query("{ 'user.name' : ?0 }")

    List<RoomReservation> findByNameUser(String user);

    // Buscar reservaciones por numero de habitación

    @Query("{ 'room.numRoom' : ?0 }")

    List<RoomReservation> findByNumRoom(Integer room);

    // Eliminar por ID de reservación

    @Query(value = "{ '\_id' : ?0 }", delete = true)

    void deleteById(String idReservation);

    // // UPDATE

    @Query(value = "{ '\_id' : ?0 }")

    RoomReservation updateRoomReservation(int idReservation, RoomReservation

    roomReservation);

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

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

 // READ

    @Query("{ '\_id' : ?0 }")

    Consumption findByid(String id);

    // DELETE

    @Query(value = "{ '\_id' : ?0 }", delete = true)

    void deleteByid(String id);

RF7 - REGISTRAR / ACTUALIZAR / BORRAR / CONSULTAR LA SALIDA DE UN CLIENTE

CRUD del registro de la salida de un cliente al hotel, con todo lo que eso implica.

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

// RQFC1

    @Aggregation(pipeline = {

            "{ $lookup: { from: 'Consumptions', localField: 'consumptions', foreignField: '\_id', as: 'consumptions' } }",

            "{ $lookup: { from: 'Rooms', localField: 'room', foreignField: '\_id', as: 'room1' } }",

            "{ $unwind: '$consumptions' }",

            "{ $unwind: '$room1' }",

            "{$project: {fechaConsumo: {$cond: {if: {$gt: ['$consumptions.date', { $dateToString: { format: '%Y-%m-%d', date: { $subtract: ['$now', { $multiply: [365, 24, 60, 60, 1000] }] } } }] }, then: {$toDate: '$consumptions.date'}, else: null}}, numRoom: '$room1.numRoom', costo: '$consumptions.cost'}}",

            "{$match: {fechaConsumo: {$ne: null}}}",

            "{$group: {\_id: '$numRoom', totalCost: {$sum: '$costo'}}}"

    })

    List<RespuestaGrupo2> findRoomConsumptionCostInLastYear();

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

 // RQFC2

    @Aggregation({

            "{$project: {" +

                    "  entryDate: {" +

                    "    $cond: {" +

                    "      if: { $gte: ['$entryDate', { $dateToString: { format: '%Y-%m-%d', date: { $subtract: [new Date(), { $multiply: [365, 24, 60, 60, 1000] }] } } }] },"

                    +

                    "      then: { $toDate: '$entryDate' }," +

                    "      else: null" +

                    "    }" +

                    "  }," +

                    "  departureDate: { $toDate: '$departureDate' }," +

                    "  room: 1" +

                    "}}",

            "{$match: {" +

                    "  entryDate: { $ne: null }" +

                    "}}",

            "{$group: {" +

                    "  \_id: '$room'," +

                    "  occupancyRate: { $sum: { $multiply: [{ $divide: [{ $subtract: ['$departureDate', '$entryDate'] }, { $multiply: [365, 24, 60, 60, 1000] }] }, 100] } }"

                    +

                    "}}",

            "{$set: {" +

                    "  occupancyRate: { $toInt: '$occupancyRate' }" +

                    "}}",

            "{$project: {" +

                    "  \_id: 0," +

                    "  occupancyRate: 1," +

                    "  room: '$\_id'" +

                    "}}"

    })

    List<RespuestaGrupo> calculateRoomOccupancyRates();

    @Aggregation(pipeline = {

            "{ $lookup: { from: 'Users', localField: 'user', foreignField: '\_id', as: 'userInfo' } }",

            "{ $unwind: { path: '$userInfo' } }",

            "{ $match: { 'userInfo.user': ?0 } }",

            "{ $lookup: { from: 'Consumptions', localField: 'consumptions', foreignField: '\_id', as: 'consumos' } }",

            "{ $unwind: { path: '$consumos' } }",

            "{ $match: { 'consumos.date': { $gte: ?1, $lte: ?2 } } }",

            "{ $lookup: { from: 'Products', localField: 'consumos.products', foreignField: '\_id', as: 'productos' } }",

            "{ $unwind: { path: '$productos' } }",

            "{ $lookup: { from: 'Rooms', localField: 'room', foreignField: '\_id', as: 'r' } }",

            "{ $unwind: { path: '$r' } }",

            "{ $project: { service\_date: '$consumos.date', room: '$r.numRoom', cost: '$productos.cost', serviceName: '$productos.type' } }"

    })

    List<RespuestaConsumoUsuario> findCustomAggregation(String userName, String startDate, String endDate);

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

  // RQFC3

    @Aggregation(pipeline = {

            "{ $lookup: { from: 'Users', localField: 'user.id', foreignField: '\_id', as: 'userInfo' } }",

            "{ $unwind: '$userInfo' }",

            "{ $match: { 'userInfo.user': ?0 } }",

            "{ $unwind: '$consumptions' }",

            "{ $match: { 'consumptions.\_id.date': { $gte: ?1, $lte: ?2 } } }",

            "{ $lookup: { from: 'products', localField: 'consumptions.\_id.product', foreignField: '\_id', as: 'producto' } }",

            "{ $unwind: '$producto' }",

            "{ $lookup: { from: 'services', localField: 'producto.service', foreignField: '\_id', as: 'servicio' } }",

            "{ $unwind: '$servicio' }",

            "{ $project: { service\_date: '$consumptions.\_id.date', room: '$room.numRoom', cost: '$producto.cost', serviceName: '$servicio.name' } }"

    })

    List<RespuestaConsumoUsuario> darConsumosCliente(String username, String startDate,

            String endDate);

    @Query(value = "{ '\_id' : ?0 }", fields = "{  'entryDate': 1, 'departureDate': 1, '\_id' : 0}")

    datesDetail findDatesById(ObjectId idReservation);

}