Entrega #2 – Proyecto

Integrantes:

Gabriela Soler – 202123744

Silvana Sandoval – 202123682

Santiago Celis – 202111131

Diseño de las consultas:

RF1:

```
//Requerimiento funcional 1
Select habitaciones.hoteles_nombre, habitaciones.consumo from INNER JOIN habitaciones on checkins.habitacion = habitacion.id Where
checkins.fecha between(01/01/2023 AND 31/12/2023);
```

RF2:

```
//Requerimiento Funcional 2
Select *, COUNT(bares.nombre), COUNT(pools.id), COUNT(restes.nombre), COUNT(spas.id), COUNT(servicios.tipo), COUNT(supers.nombre),
COUNT(salones.id), COUNT(prestamos.id), COUNT(tiendas.nombre)
FROM INNER JOIN Semanas on pools.semana = semanas.numeroSemanas, INNER JOIN Semanas on bares.semana = semanas.numeroSemanas,
INNER JOIN Semana on rests.semana = semanas.numeroSemanas, INNER JOIN Semana on spas.semana = semanas.numeroSemanas, INNER JOIN Semanas on salones.semana =
semanas.numeroSemanas,
INNER JOIN Semanas on supers.semana = semanas.numeroSemanas, INNER JOIN tiendas.semanas = semanas.numeroSemanas, INNER JOIN prestamos.semanas =
semanas.numeroSemanas
Where rests.fecha between (fecha_1 And fecha_2) And pools.fecha between (fecha_1 And fecha_2) AND bares.fecha between (fecha_1 And fecha_2) AND
spas.fecha between (fecha_1 And fecha_2) AND salones.fecha between (fecha_1 And fecha_2) AND supers.fecha between (fecha_1 And fecha_2)
AND tiendas.fecha between (fecha_1 And fecha_2) AND prestamos.fecha between (fecha_1 And fecha_2) And MAX(COUNT(bares.nombre)) AND MAX(COUNT(pools.id)) AND
MAX(COUNT(restes.nombre)) AND MAX(COUNT(spas.id)) AND MAX(COUNT(servicios.tipo)) AND MAX(COUNT(supers.nombre)) AND MAX(COUNT(salones.id)) AND
MAX(COUNT(prestamos.id)) AND MAX(COUNT(tiendas.nombre));
```

RF3

```
//CONSULTAS NUEVAS:
@Query(value = "SELECT habitaciones.id, Habitaciones.tipo, COUNT(Reservas.id), "+
    "(SUM(ABS(Reservas.fechasalida - Reservas.fechaentrada)) / 365.0)"+
    "FROM habitaciones LEFT JOIN reservas ON habitaciones.id = Reservas.id_habitacion "+
    "WHERE reservas.fechaentrada >= '01-01-2023' AND reservas.fechaentrada < '01-01-2024' ", nativeQuery = true)
Collection<Habitacion> darIndicesHabitaciones();
```

RF4:

```
//Requerimiento 4
Select * FROM INNER JOIN Semanas on pools.semana = semanas.numeroSemanas, INNER JOIN Semanas on bares.semana = semanas.numeroSemanas,
INNER JOIN Semana on rests.semana = semanas.numeroSemanas, INNER JOIN Semana on spas.semana = semanas.numeroSemanas, INNER JOIN Semanas on salones.semana =
semanas.numeroSemanas,INNER JOIN Semanas on supers.semana = semanas.numeroSemanas, INNER JOIN tiendas.semanas = semanas.numeroSemanas, INNER JOIN
prestamos.semanas = semanas.numeroSemanas WHERE
```

RF5:

```
@Query(value = "SELECT cliente.documento, cliente.name, cuenta.consumo FROM cuentas, clientes"+
"Where cliente := cliente"
+"INNER JOIN cuentas on clientes.documento = cuentas.cliente.documento", nativeQuery=true)
Collection<Cuenta> darCuentasCliente(@Param("cliente") Integer cliente);
```

RF6:

```
@Query(value = "SELECT fecha, COUNT(*) as ocupacion\r\n" +
    "FROM checkin\r\n" +
    "GROUP BY fecha\r\n" +
    "ORDER BY ocupacion DESC\r\n" +
    "LIMIT 1", nativeQuery = true)
Date darFechaMayorOcupacion();

@Query(value = "SELECT fecha, COUNT(*) as ocupacion\r\n" +
"FROM checkin\r\n" +
"GROUP BY fecha\r\n" +
"ORDER BY ocupacion ASC\r\n" +
"LIMIT 1", nativeQuery = true)
Date darFechaMenorOcupacion();

@Query(value = "SELECT fecha, SUM(consumo) as ingresos\r\n" +
    "FROM cuenta\r\n" +
    "INNER JOIN checkin ON cuenta.documento = checkin.documento\r\n" +
    "GROUP BY fecha\r\n" +
    "ORDER BY ingresos DESC\r\n" +
    "LIMIT 1", nativeQuery = true)
Date darFechaMayoresIngresos();
```

RF7:

```
@Query(value = "SELECT clientes.*, cuentas.consumo, reservas.fechaentrada FROM clientes INNER JOIN reservas.clientes_documento = clientes.documento "+
"INNER JOIN clientes.documento = cuentas.clientes_documento "+
"WHERE ((reservas.fechasalida - reservas.fechaentrada) >= 14) OR (cuentas.consumo > 15000000 AND reservas.fechaentrada > '01-01-2023') ", nativeQuery = true)
Collection<Cliente> darBuenosClientes();
```

RF8:

```
//Requerimiento Funcional 8
Select *, COUNT(bares.nombre), COUNT(pools.id), COUNT(restes.nombre), COUNT(spas.id), COUNT(servicios.tipo), COUNT(supers.nombre),
COUNT(salones.id), COUNT(prestamos.id), COUNT(tiendas.nombre)
FROM INNER JOIN Semanas on pools.semana = semanas.numeroSemanas, INNER JOIN Semanas on bares.semana = semanas.numeroSemanas,
INNER JOIN Semana on rests.semana = semanas.numeroSemanas, INNER JOIN Semana on spas.semana = semanas.numeroSemanas, INNER JOIN Semanas on salones.semana =
semanas.numeroSemanas,
INNER JOIN Semanas on supers.semana = semanas.numeroSemanas, INNER JOIN tiendas.semanas = semanas.numeroSemanas, INNER JOIN prestamos.semanas =
semanas.numeroSemanas Where MIN(COUNT(pools.id)) AND MIN(COUNT(restes.nombre)) AND MIN(COUNT(spas.id)) AND MIN(COUNT(servicios.tipo)) AND
MIN(COUNT(supers.nombre)) AND MIN (COUNT(salones.id)) AND MIN(COUNT(prestamos.id)) AND MIN(COUNT(tiendas.nombre));
```

RF9:

```
//Requerimiento 9
@Query(value = "SELECT * FROM clientes where hotel_nombre = :hotel_nombre"
+ "INNER JOIN clientes on clientes.documento = cuentas.clientes_documento"
+ "INNER JOIN cuentasspas on cuentas.id = cuentasspas.cuentas_id"
+ "INNER JOIN cntasbares on cuentas.id = cntasbartes.cuentas_id"
+ "INNER JOIN cntasredes on cuentas.id = cntasredes.cuentas_id"
+ "INNER JOIN cntasres on cuentas.id = cntasres.cuentas_id"
+ "INNER JOIN cntassuper on cuentas.id = cntassuper.cuentas_id"
+ "INNER JOIN cntastiendas on cuentas.id = cntastiendas.cuentas_id"
+ "INNER JOIN cuentasspas on spas.id = cuentasspas.spas_id"
+ "INNER JOIN cntasbares on bares.nombre = cntasbares.bares_nombre"
+ "INNER JOIN cntasredes on internets.id =cntasredes.internets_id"
+ "INNER JOIN cntasres on restaurantes.rests_id = cntasres.restaurantes_rests_id"
+ "INNER JOIN cntassuper on supers.nombre = supers.supers_nombre"
+ "INNER JOIN cntastiendas on tiendas.nombre = cntastiendas.tiendas_nombre"
+ "AND (cuentasspas.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntasbares.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntasredes.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntasres.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntassuper.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntastiendas.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "GROUPBY caracteristica", nativeQuery = true)
Collection<Cliente> darClientesCaracteristica(@Param("hotel_nombre") String hotel_nombre);
```

RF10:

```
//Requerimiento 10
@Query(value = "SELECT * FROM clientes where hotel_nombre = :hotel_nombre"
+ "INNER JOIN clientes on clientes.documento = cuentas.clientes_documento"
+ "INNER JOIN cuentasspas on cuentas.id = cuentasspas.cuentas_id"
+ "INNER JOIN cntasbares on cuentas.id = cntasbartes.cuentas_id"
+ "INNER JOIN cntasredes on cuentas.id = cntasredes.cuentas_id"
+ "INNER JOIN cntasres on cuentas.id = cntasres.cuentas_id"
+ "INNER JOIN cntassuper on cuentas.id = cntassuper.cuentas_id"
+ "INNER JOIN cntastiendas on cuentas.id = cntastiendas.cuentas_id"
+ "INNER JOIN cuentasspas on spas.id = cuentasspas.spas_id"
+ "INNER JOIN cntasbares on bares.nombre = cntasbares.bares_nombre"
+ "INNER JOIN cntasredes on internets.id =cntasredes.internets_id"
+ "INNER JOIN cntasres on restaurantes.rests_id = cntasres.restaurantes_rests_id"
+ "INNER JOIN cntassuper on supers.nombre = supers.supers_nombre"
+ "INNER JOIN cntastiendas on tiendas.nombre = cntastiendas.tiendas_nombre"
+ "AND (cuentasspas.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntasbares.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntasredes.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntasres.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntassuper.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "OR (cntastiendas.fecha BETWEEN (fecha_inicial =: fecha_inicial AND fecha_final =: fecha_final))"
+ "AND cuenta.valor = 0"
+ "GROUPBY caracteristica", nativeQuery = true)
Collection<Cliente> darClientesCaracteristica2(@Param("hotel_nombre") String hotel_nombre);
```

RF12:

```
//Requerimiento 12
@Query(value = "SELECT * FROM clientes WHERE hotel_nombre = :hotel_nombre"
+ "INNER JOIN checkins ON clientes.checkins_id = checkins.id"
+ "INNER JOIN checkouts ON clientes.checkouts_id = checkouts.id"
+ "INNER JOIN cuentas on clientes.documento = cuentas.clientes_documento"
+ "INNER JOIN cuentasspas on cuentas.id = cuentasspas.cuentas_id"
+ "AND (((SELECT dia from checkins) Between ((1/01/2023 And 31/03/2023) OR (1/04/2023 And 30/06/2023) OR (1/07/2023 And 30/09/2023) OR (1/10/2023 And 31/12/2023)))"
+ "AND ((SELECT dia from checkouts) Between ((1/01/2023 And 31/03/2023) OR (1/04/2023 And 30/06/2023) OR (1/07/2023 And 30/09/2023) OR (1/10/2023 And 31/12/2023))))"
+ "OR (SELECT consumo from cuentas where consumo > 300000)"
+ "OR (clientes.documento = cuentas.clientes_documento AND cuentas.id = cuentasspas.cuentas_id)", nativeQuery=true)
Collection<Cliente> darClientesHotel(@Param("hotel_nombre") String hotel_nombre);
```

Análisis:

- En respuesta al diseño existente, hemos analizado el impacto de la introducción de los nuevos requerimientos en el modelo conceptual (diagrama de clases UML) presentado anteriormente, obteniendo consideraciones positivas acerca de los cambios para la optimización de las consultas. Hemos tenido en cuenta los comentarios proporcionados por los monitores durante las sustentaciones del anterior taller identificando así aspectos a mejorar. Como resultado de este análisis, se realizaron ajustes necesarios en el modelo conceptual, actualizando clases específicas para reflejar con precisión los cambios introducidos por los nuevos requerimientos. Además, para reflejar estos ajustes en la implementación concreta, se han realizado modificaciones en el modelo relacional de la base de datos, esto con el fin de garantizar la coherencia entre el modelo conceptual y el modelo relacional.

  Por otro lado, para la siguiente entrega, se modifican las tablas que estén relacionadas con los servicios con el fin de agruparlas. Este cambio busca optimizar las consultas sintetizando toda la información en un solo lugar.

Índices:

RF1: Para optimizar esta consulta se usarán índices sobre el atributo de fechas en la tabla checkins ya que esta permitiría organizar y buscar las fechas de estos de manera más eficiente porque la sentencia solo deberá leer las que estén en ese rango.

RF2: En esta sentencia se usarán índices sobre las fechas de los servicios ya que esto podrá agilizar las consultas al estas estar organizadas. De resto, los demás datos usados en la consulta ya se encuentran indexados, por lo que no tiene sentido crear índices nuevos.

RF3: En esta consulta se indexará por la fecha de las reservas para facilitar la búsqueda de las fechas mayores a la propuesta como límite inferior y menores que la propuesta de límite superior.

RF4: Para este requerimiento tendría sentido hacer índices en los diferentes atributos que tienen las tablas que no han sido indexados. Por ejemplo, hacer una indexación por rango de precio, de fechas, tipo de servicio, etc.

RF5: Esta, al ser una consulta tan sencilla, no tiene sentido hacer indexación ya que se requiere mostrar el consumo de un usuario dado y este ya se encuentra indexado por su documento de identidad.

RF6: Los índices a usar para este requerimiento son:

1. Índice por fechas sobre la tabla de check-ins y de servicios ya que esto permite explorar los consumos de las habitaciones, las fechas de mayor y menor ocupación, etc.

RF7: Para este requerimiento tiene sentido indexar las tablas de los servicios por fecha, al igual que la tabla de cuentas por consumo para poder encontrar fácilmente ese cliente que ha tenido un alto consumo en el último año.

RF8: Para este requerimiento se hace la indexación a partir del número de las semanas para qu e su consulta sea más rápida.

RF9: La indexación se haría a partir de las fechas de las tablas que llevan las cuentas de los diferentes servicios ya que son aquellas en las que se hacen comparaciones. Esto haría que las consultas fueran óptimas.

RF10: La indexación ser haría sobre cuentas en el atributo de consumo para poder encontrar aquellos clientes que no han consumido nada. Adicionalmente, se haría indexación en las tablas que llevan las cuentas de los diferentes servicios para poder optimizar la búsqueda de las fechas.

RF11: En este caso, se indexaría sobre la tabla reservas por los campos de fecha de inicio y finalización de la reserva y por el número de habitación asociada. También se haría sobre el servicio para poder hacer más eficiente la consulta respecto al funcionamiento de este.

RF12: La indexación se haría a partir de las fechas de check-in y check-out para hacer las consultas más rápidas buscando el rango de fechas que se desea. Adicionalmente, se indexarían los servicios por su precio y servicios como spa o salones por su duración.

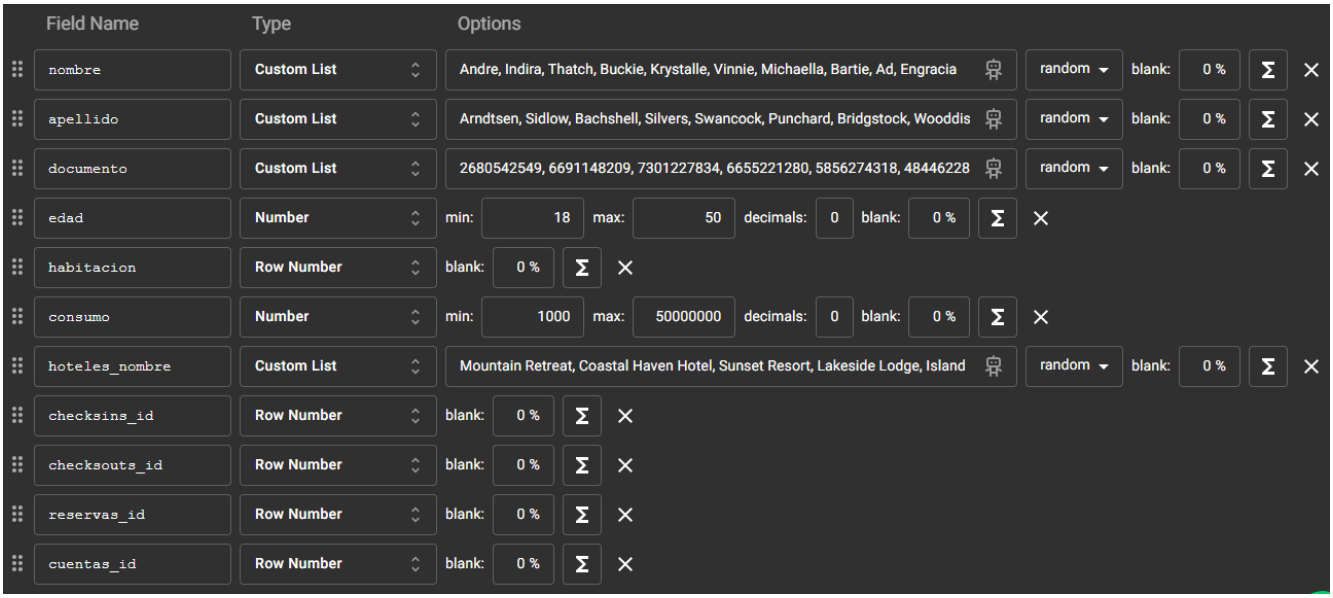
Carga de datos:

Antes de cargar los datos, se realizaron ajustes tanto en el modelo UML como en el modelo relacional con el objetivo de mejorar la estructura de la base de datos y optimizar las consultas. Los datos fueron cargados teniendo en consideración estas modificaciones. En total, la carga de datos incluye alrededor de 510,000 registros distribuidos en varias tablas. La decisión de alcanzar esta cantidad de registros se basa en mantener un valor cercano al límite máximo de datos. Sin embargo, debido a restricciones de tiempo, algunas tablas contienen menos datos que otras.

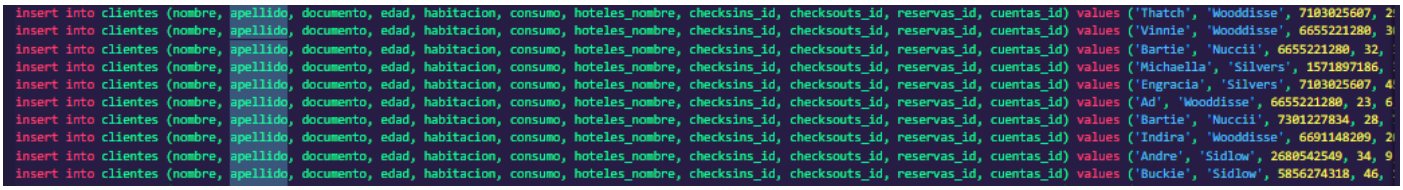
La carga de datos se llevó a cabo mediante la utilización de una aplicación web denominada Mockaroo. Esta plataforma facilita la introducción de los atributos (columnas) de una tabla, permitiendo especificar el tipo de dato asociado a cada uno. De esta manera, se logra generar la información requerida de manera eficiente.

Ejemplo: (Datos para cliente)

Proceso en Mockaroo (https://www.mockaroo.com/)



Resultado



Además, está nueva carga de datos tiene en mente la posible utilización de índices en la base de datos y, por consiguiente, en las consultas. Este aspecto es importante porque asegura una eficiencia óptima en la gestión de grandes cantidades de información.

(Estos datos están en la carpeta NewData)