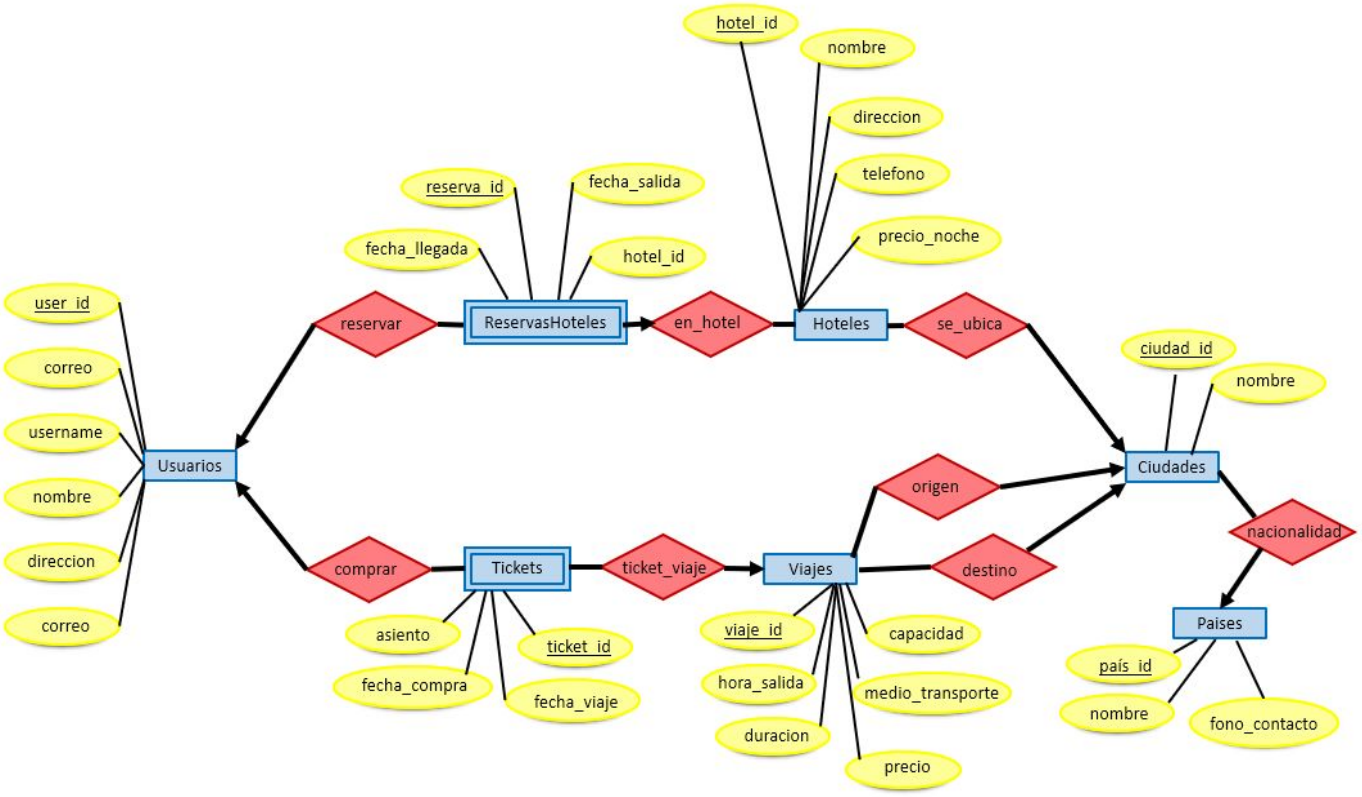
# INFORME ENTREGA 2

**Christian Luer**
**Tomas irarrazaval**

## 2.1 Esquema y Modelo:




**Entidades**

Usuarios(user_id int PK, username varchar(50), nombre varchar(50), correo varchar(50), direccion varchar(100))

ReservasHoteles(reserva_id int PK, fecha_llegada date, fecha_salida date)

Hoteles(hotel_id int PK, nombre varchar(30), dirección varchar(100), teléfono varchar(30), precio_noche money)

Ciudades(ciudad_id int PK, nombre varchar(30))

Paises(pais_id int PK, nombre varchar(30), fono_contacto varchar(30))

Viajes(viaje_id int PK, hora_salida time, duración int, medio_transporte varchar(30), capacidad int, precio money)

Tickets(ticket_id int PK, asiento int, fecha_compra timestamp, fecha_viaje date)

**Relaciones**
Todas las relaciones cuentan con una llave primaria que incluye toda la información de dicha fila. Esto se debe a que las llaves primarias tienen que ser únicas y estas tablas nunca tienen dos filas iguales por construcción.

reservar(user_id int FK, reserva_id int FK) (PRIMARY KEY (user_id, reserva_id))

en_hotel(reserva_id int FK, hotel_id int FK)  (PRIMARY KEY (reserva_id, hotel_id))

se_ubica(hotel_id int FK, ciudad_id int FK)  (PRIMARY KEY (hotel_id, ciudad_id))

nacionalidad(ciudad_id int FK, pais_id int FK)  (PRIMARY KEY (ciudad_id, pais_id))

origen(viaje_id int FK, ciudad_id int FK)  (PRIMARY KEY (viaje_id, ciudad_id))

destino(viaje_id int FK, ciudad_id int FK)  (PRIMARY KEY (viaje_id, ciudad_id))

ticket_viaje(ticket_id int FK, viaje_id int FK)  (PRIMARY KEY (ticket_id viaje_id))

comprar(user_id int FK, ticket_id int FK)   (PRIMARY KEY (user_id, ticket_id))

## 2.2 Justificar Modelo:

Dependencias funcionales por entidad:

Usuarios: user_id → username, nombre, correo, direccion

> Cada usuario tiene un id único. Es fácil ver que dos personas pueden tener un mismo nombre y no ser el mismo usuario (existen muchos Nicolas Rodriguez). Esto también podría ocurrir con la dirección (dos usuarios que viven en la misma casa), con el correo (dos familiares que usan la misma cuenta (madre e hijo), o alguna institución que representa a muchas personas), username podría no estar limitado a que cada usuario tenga uno distinto. Esto entrega mayor libertad para los clientes a la hora de elegir qué username desean usar. Cada user_id, corresponde a un único usuario específico.

Ciudades: ciudad_id → nombre

> Pueden existir dos ciudades distintas (en distintos países por ejemplo) con un mismo nombre. (Dato: existen 25 ciudades en el mundo llamadas San Pedro). Cada ciudad_id, corresponde a una única ciudad específica.

Paises: pais_id → nombre, fono_contacto

Se decide usar la llave pais_id a pesar de que no pueden haber dos países con un mismo nombre. Se decidió modelar de esta forma por el issue #138.

Viajes: viaje_id → hora_salida, duración, medio_transporte, capacidad, precio

Nada evita que existan más de un viaje que comparta la hora de salida, duración , medio, capacidad y el precio. Todas las variables anteriores son independientes entre sí ya que solo dependen del id de viaje.

Tickets: ticket_id → asiento, fecha_compra, fecha_viaje

Los atributos asiento, fecha de compra y fecha de viaje son independientes entre sí, pero todos dependen del id del ticket.

Hoteles: hotel_id → nombre, direccion, teléfono, precio

Pueden existir dos hoteles con un mismo nombre, dos hoteles distintos que ocupen un mismo número telefónico (en algunas ocasiones dos de una misma cadena que están al lado).

ReservasHoteles: reserva_id → fecha_llegada, fecha_salida

Es fácil ver que pueden existir muchas reservas distintas con una misma fecha de llegada y fecha de salida. Se decidió hacer ReservasHoteles como una entidad y no una relación entre hoteles y usuarios por el motivo que un usuario puede hacer más de una reserva en un mismo hotel en fechas distintas.

Todas las relaciones (origen, destino, comprar, reservar, ticket_viaje, nacionalidad, se_ubica, hotel) cuentan con una llave primaria que incluye toda la información de dicha fila. Esto se debe a que las llaves primarias tienen que ser únicas y estas tablas nunca tienen dos filas iguales por construcción.
Nuestro modelo está en 3NF ya que para toda dependencia funcional no trivial X → Y, X es llave o Y es parte de una llave minimal. Esto ocurre por ejemplo con la tabla paises.

**Consultas:**

1) Username junto con su correo se obtiene de la tabla Usuarios:

```
SELECT username, correo FROM usuarios
```

2) Todas las ciudades con país de nombre ‘ingresado’ en donde la agencia tiene presencia:

```
SELECT ciudades.nombre FROM ciudades, paises, nacionalidad where
LOWER(paises.nombre) LIKE LOWER('%$nombre_pais%') AND
paises.pais_id = nacionalidad.pais_id AND ciudades.ciudad_id =
nacionalidad.ciudad_id;
```

3) Todos los países en donde una persona con el username ‘ingresado’ se ha hospedado:

```
SELECT paises.nombre FROM usuarios, reservar, en_hotel, se_ubica,
paises, nacionalidad WHERE LOWER(usuarios.username) LIKE
LOWER('%$tipo%') AND usuarios.user_id = reservar.user_id AND
en_hotel.reserva_id = reservar.reserva_id AND se_ubica.hotel_id =
en_hotel.hotel_id AND nacionalidad.ciudad_id = se_ubica.ciudad_id
AND paises.pais_id = nacionalidad.pais_id;
```

4)

```
SELECT usuarios.nombre, sum(viajes.precio) FROM usuarios, viajes,
comprar, ticket_viaje WHERE $id_usuario = usuarios.user_id AND
comprar.user_id = usuarios.user_id AND ticket_viaje.ticket_id =
comprar.ticket_id AND viajes.viaje_id = ticket_viaje.viaje_id
GROUP BY usuarios.nombre;
```

5)

```
SELECT usuarios.user_id, usuarios.nombre,
reservashoteles.fecha_llegada, reservashoteles.fecha_salida,
hoteles.nombre FROM usuarios, reservashoteles, hoteles, en_hotel,
reservar WHERE reservashoteles.fecha_llegada >= '2020-01-01' AND
reservashoteles.fecha_salida <= '2020-03-31' AND
reservashoteles.reserva_id = en_hotel.reserva_id AND
reservashoteles.reserva_id = reservar.reserva_id AND
usuarios.user_id = reservar.user_id;
```

6)

```
SELECT usuarios.user_id, usuarios.nombre, sum(viajes.precio) FROM
usuarios, tickets, viajes, comprar, ticket_viaje WHERE
date(tickets.fecha_compra) >= '$fecha_ini' AND
date(tickets.fecha_compra) <= '$fecha_fini' AND comprar.ticket_id
= tickets.ticket_id AND usuarios.user_id = comprar.user_id AND
ticket_viaje.ticket_id = tickets.ticket_id AND viajes.viaje_id =
ticket_viaje.viaje_id GROUP BY usuarios.user_id;
```

codigos:

**DATOS**
**ciudades**

**CREATE TABLE ciudades(ciudad_id int PRIMARY KEY, nombre varchar(30));**

**usuarios**

**CREATE TABLE usuarios(user_id int PRIMARY KEY, username varchar(50), nombre varchar(50) , correo varchar(50), direccion varchar(100));**

**reservashoteles**

**CREATE TABLE reservashoteles(reserva_id int PRIMARY KEY, fecha_llegada date, fecha_salida date);**

**hoteles**

**CREATE TABLE hoteles(hotel_id int PRIMARY KEY, nombre varchar(30), direccion varchar(100), telefono varchar(30), precio money);**
**CREATE TABLE**

**paises**

**grupo109=> CREATE TABLE paises(pais_id int PRIMARY KEY, nombre varchar(30), fono_contacto varchar(30));**

**viajes**

**CREATE TABLE viajes(viaje_id int PRIMARY KEY, hora_salida time, duracion int, medio_transporte varchar(30), capacidad int, precio money);**
**CREATE TABLE**

**grupo109=> \COPY viajes FROM 'VIAJES.csv' DELIMITER ',' CSV HEADER ENCODING 'windows-1251';**

**tickets**

**grupo109=> CREATE TABLE tickets(ticket_id int PRIMARY KEY, fecha_compra timestamp, fecha_viaje date);**

**RELACIONES**

reservar:

CREATE TABLE reservar(user_id int REFERENCES usuarios(user_id) ON DELETE CASCADE, reserva_id int REFERENCES reservashoteles(reserva_id) ON DELETE CASCADE, PRIMARY KEY(user_id, reserva_id));

en_hotel:

CREATE TABLE en_hotel(reserva_id int REFERENCES reservashoteles(reserva_id) ON DELETE CASCADE, hotel_id int REFERENCES hoteles(hotel_id) ON DELETE CASCADE, PRIMARY KEY(reserva_id, hotel_id));

se_ubica

CREATE TABLE se_ubica(hotel_id int REFERENCES hoteles(hotel_id) ON DELETE CASCADE, ciudad_id int REFERENCES ciudades(ciudad_id) ON DELETE CASCADE, PRIMARY KEY(hotel_id, ciudad_id));

nacionalidad

CREATE TABLE nacionalidad(ciudad_id int REFERENCES ciudades(ciudad_id) ON DELETE CASCADE, pais_id int REFERENCES paises(pais_id) ON DELETE CASCADE, PRIMARY KEY(ciudad_id, pais_id));

origen

CREATE TABLE origen(viaje_id int REFERENCES viajes(viaje_id) ON DELETE CASCADE, ciudad_id int REFERENCES ciudades(ciudad_id) ON DELETE CASCADE, PRIMARY KEY(viaje_id, ciudad_id));

destino

CREATE TABLE destino(viaje_id int REFERENCES viajes(viaje_id) ON DELETE CASCADE,  ciudad_id int REFERENCES ciudades(ciudad_id) ON DELETE CASCADE, PRIMARY KEY(viaje_id, ciudad_id));