3/20/2018

UNIVERSIDAD DE LOS ANDES
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

SISTEMAS TRANSACCIONALES
CLAUDIA LUCIA JIMENEZ

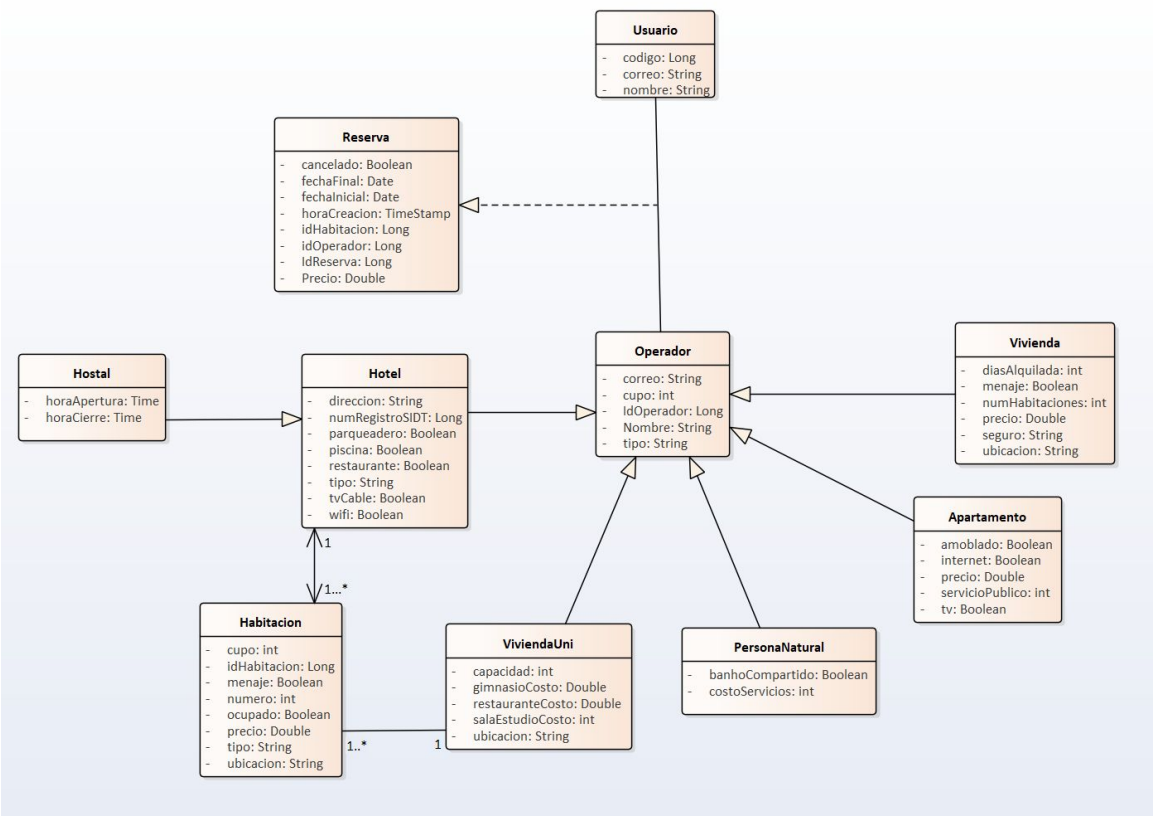
REPORTE DE ENTREGA

ITERACION No. 1

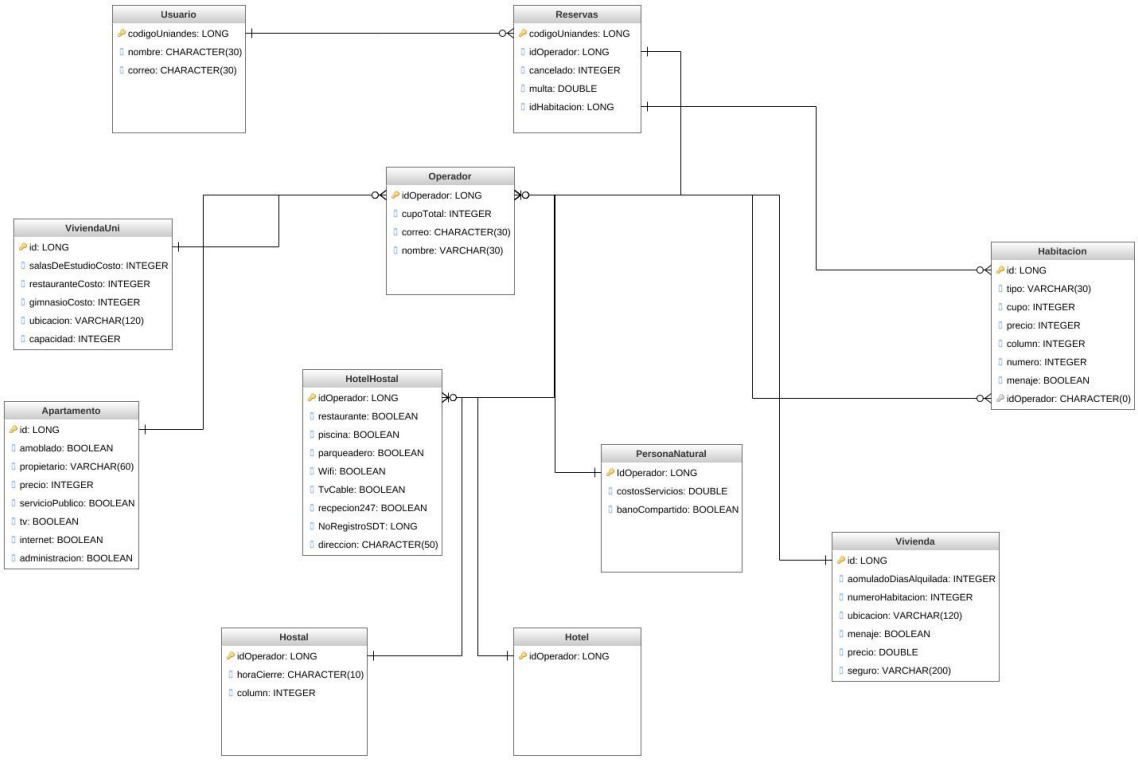
INTEGRANTES DEL GRUPO

GREGORIO OSPINA - 201631760
ANDRES SILVA - 201632266

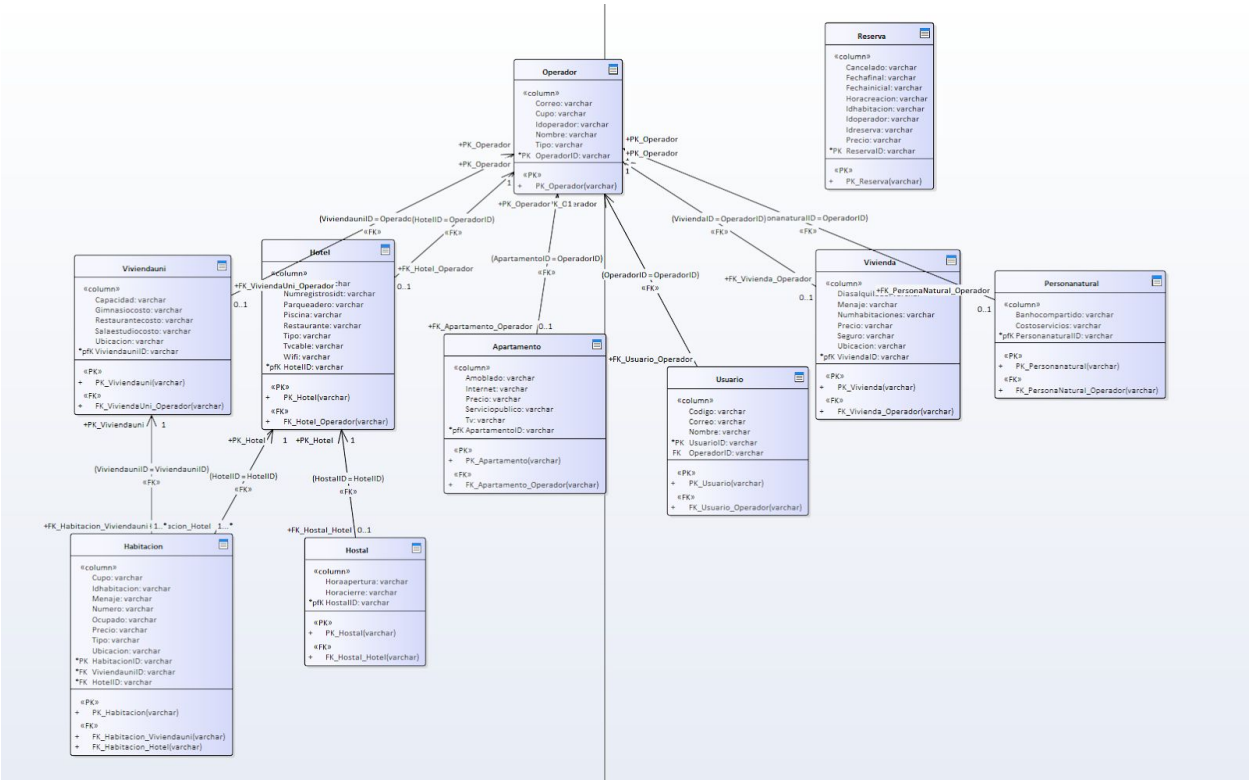
Modelo UML



Modelo generado por nosotros en GenMyModel



Modelo generado por Enterprise Architect.



** en el archivo del proyecto están los proyectos en mejor tamaño y formato original.

Comparacion modelo auto generado.

En el modelo generado por E.A es notable que a la clase de reserva se le pierden las relaciones y las llaves foráneas, esto crearía muchos problemas al no conectar la reserva ni con un Usuario ni con un Operador. Por otro lado vemos que el modelo generado tiene muchas llaves foráneas compuestas a diferencia del modelo que creamos nosotros. De resto se puede ver que hemos tomado buenas decisiones con un modelo que tiene sentido y que es muy compatible con el modelo que diseñó Enterprise Architect.

Reporte de recursos.

	Operadores	Habitacion	Reserva	Usuario
GET	Funcional para los 6 tipos de operadores, tanto el getAll, como el getById, getByTipoDeOperador	Funcional para todos los metodos (getAll, getById, getDisponible, getDisponibleByOperador)	Todos los metodos estan funcionales (getAll, getById, getByUsuarioOperador)	Funcional para todos los metodos (getAll, getByCodigo)
PUT	Funcional en todos los tipos de operador.	Funcional	Funcional	Funcional
POST	Funcional en todos los tipos de operador.	Funcional	Funcional	Funcional
DELETE	Esta funcional para los 6 tipos de operadores teniendo en cuenta que no hayan reservas	Elimina satisfactoriamente	Elimina satisfactoriamente	Elimina satisfasctoriamente

Reglas de negocio adicionales

Encontramos en el caso de estudio, varias reglas de negocio adicionales a las propuestas, que son importantes para mantener la transaccionalidad y la concurrencia de este:

- Una habitacion involucrada en una reserva activa no puede ser alquilada otravez
- Cuando se elimina un operador, se deben borrar las habitaciones de manera cascada.
- Ni la fecha de inicio ni el tiempo de creación de una reserva se puede modificar.
- El correo del usuario debe estar en el formato adecuado.

Pruebas con Postman.

En el archivo collections del proyecto están todas las pruebas de Postman con los recursos del proyecto.

Aclaraciones para los RFC1 y RFC2

-- RFC1
- Con la manera como poblamos las tablas, es necesario cambiar la instancia donde dice "CURRENT_DATE - 365" a por lo menos "CURRENT_DATE - 800" ya que los datos que le insertamos a las tablas son de años menores a un año desde el current time.

-- RFC2
-- Nosotros consideramos que la oferta mas popular es aquella habitacion de un mismo operador que tenga la mayor cantidad de apariciones en las reservas. La condicion de que solo sean 20 es restringida en el DAO donde solo se itera sobre los primeros 20 Resultsets.