Iteración 3

Pedro L. Lobato Barros
Universidad de los Andes, Bogotá, Colombia
{p.lobato}@uniandes.edu.co
Fecha de presentación: Mayo 06 de 2023

Tabla de contenido

1 Introducción

La aplicación que se desarrolló es una aplicación Java que utiliza el marco de trabajo *Java Data Objects* (*JDO*) para administrar operaciones transaccionales y de consulta en una base de datos relacional. *JDO* proporciona una capa de abstracción que permite al programador trabajar con objetos Java y dejar que el proveedor de *JDO* se encargue de la persistencia de los datos en la base de datos.

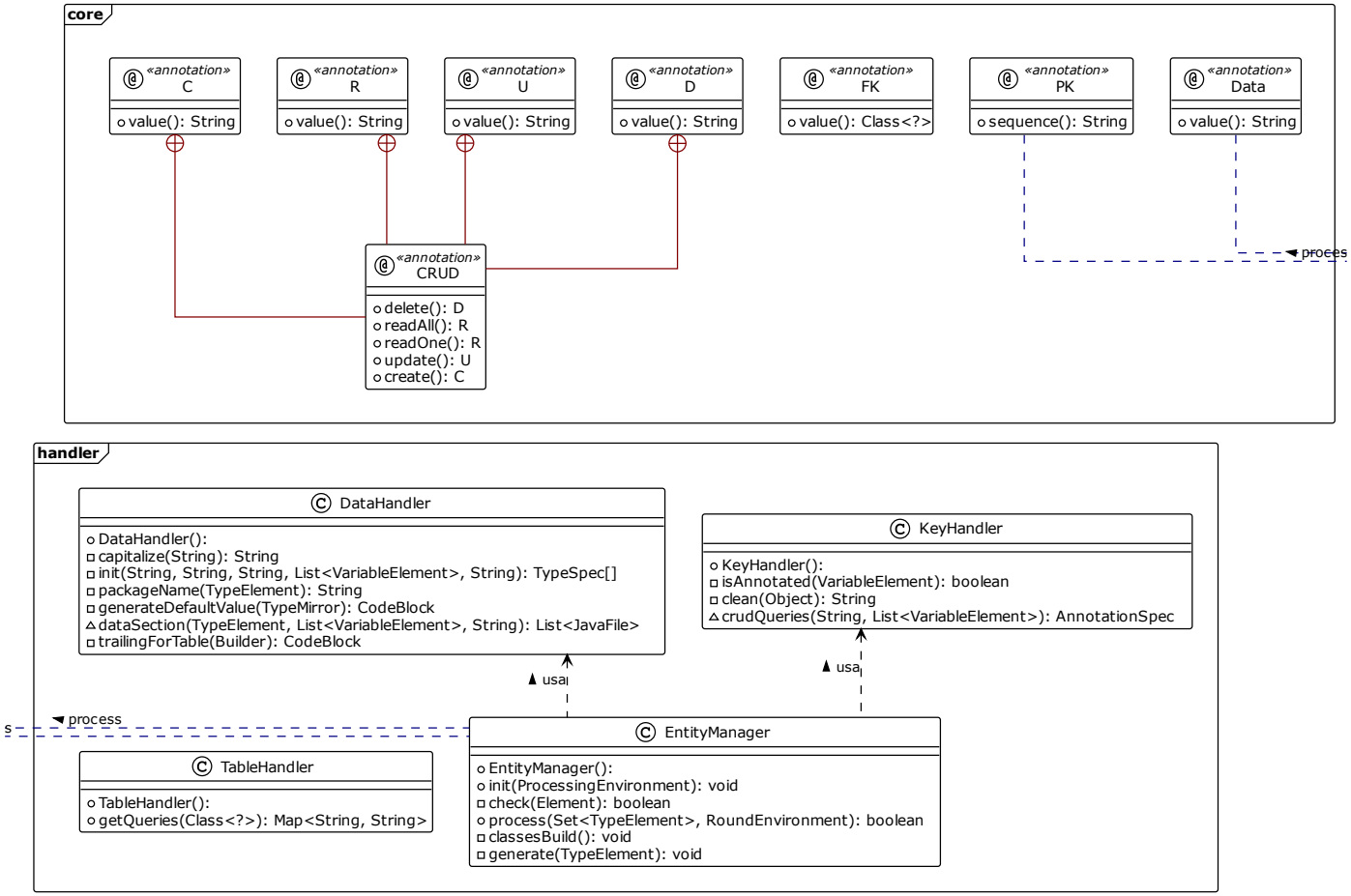
La aplicación utiliza operaciones *CRUD* (Crear, Leer, Actualizar y Eliminar) para interactuar con la base de datos, así como operaciones de consulta para responder a los requerimientos funcionales del usuario. Además, la aplicación utiliza código SQL embebido para realizar completar dichas operaciones más avanzadas en la base de datos.

2 Modelos

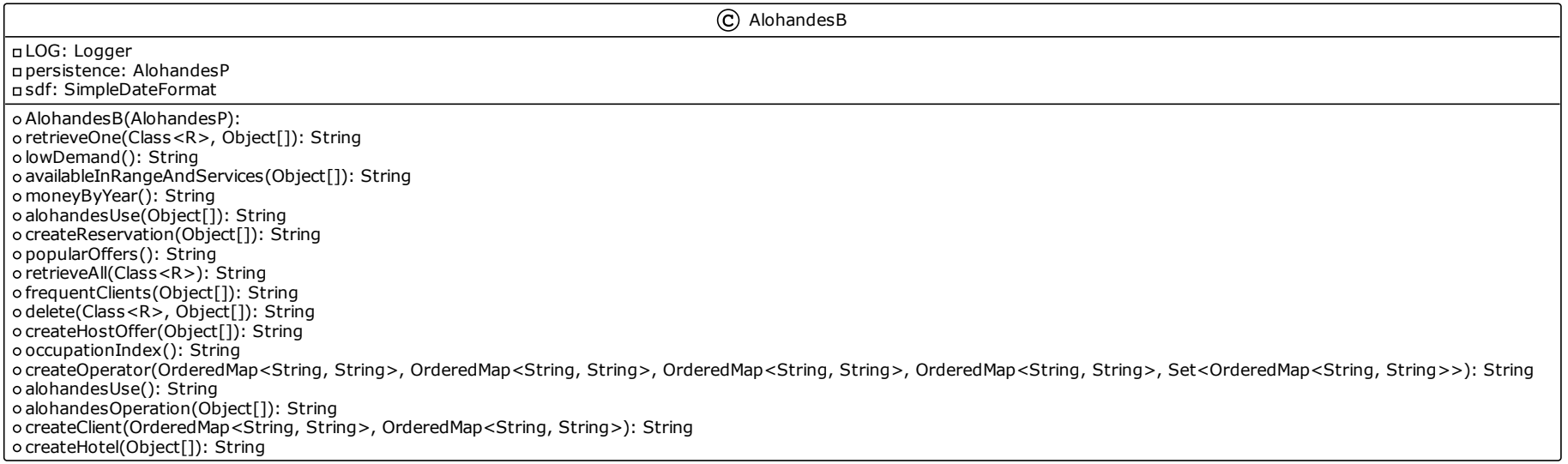
2.1 Modelo de clases

Para el desarrollo de una aplicación eficiente y escalable, es importante definir una arquitectura de solución clara y organizada. En este sentido, se ha dividido el proyecto en seis submódulos que se encargan de distintas tareas en el proceso de desarrollo:

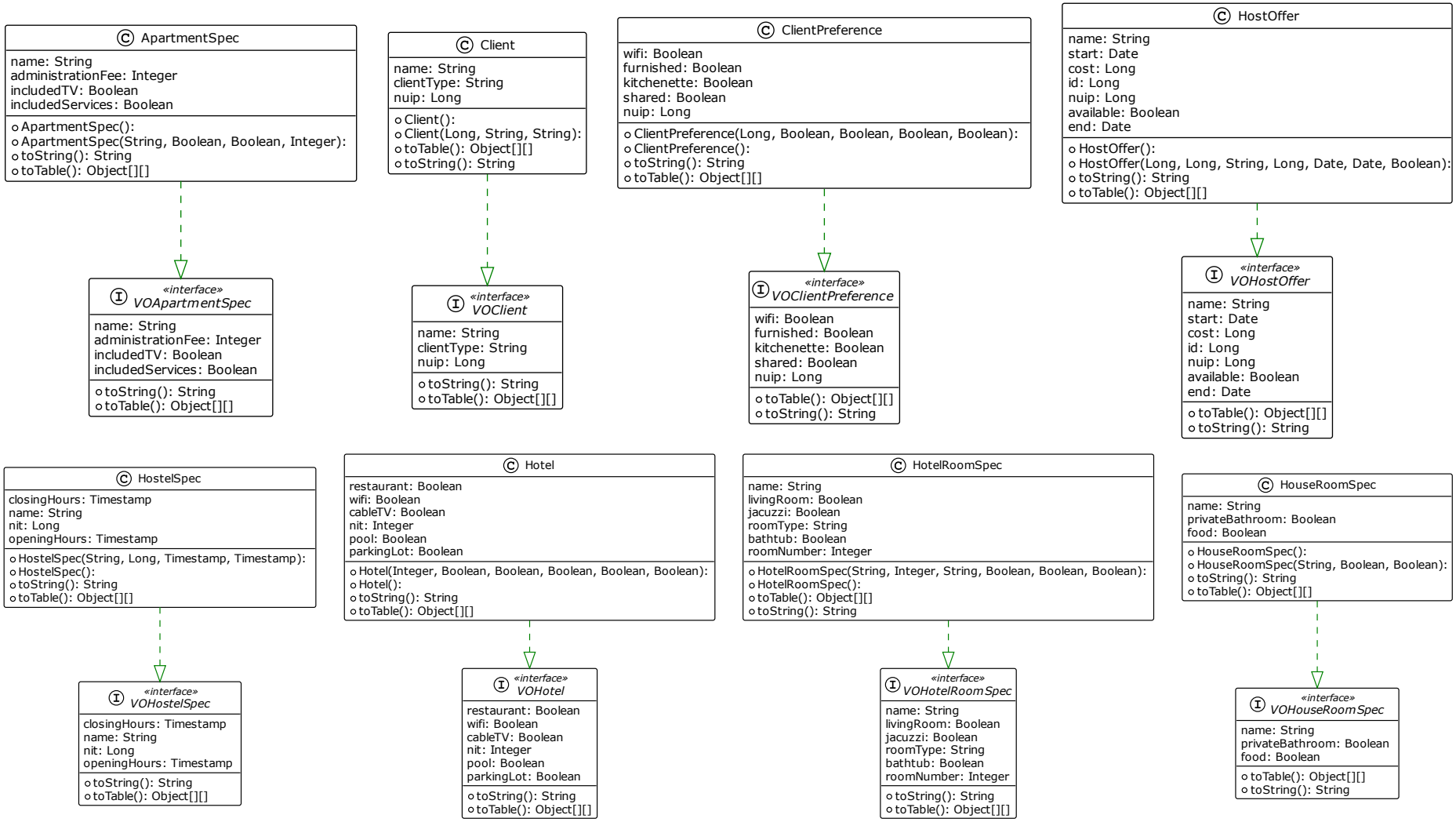
El primer submódulo, "annotation-magic", contiene un annotation-processor que facilita la creación de POJOs, VOs y Queries del CRUD mediante el uso de anotaciones.

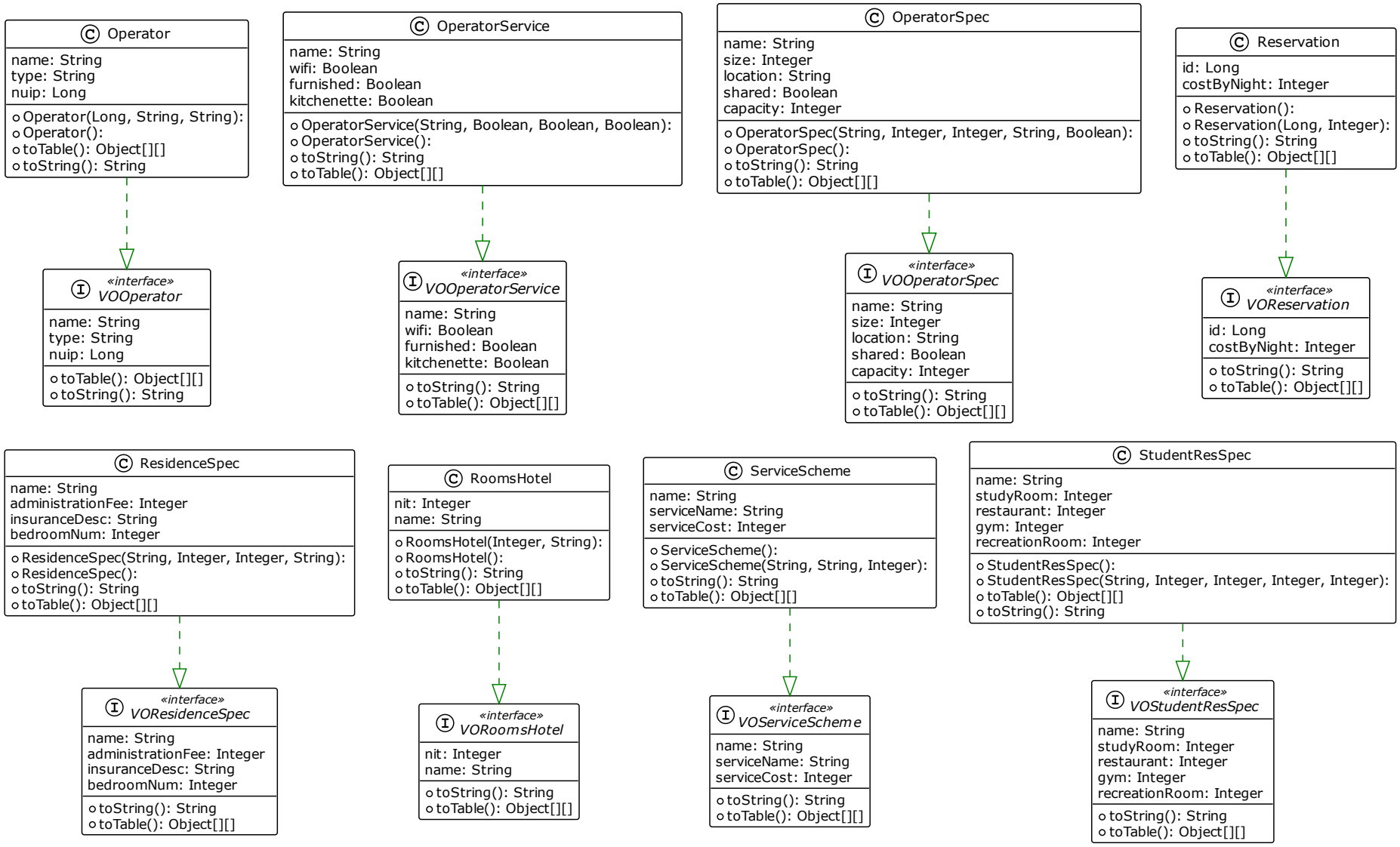

core
«annotation»
C
value(): String
«annotation»
R
value(): String
«annotation»
U
value(): String
«annotation»
D
value(): String
«annotation»
FK
value(): Class<?>
«annotation»
PK
sequence(): String
«annotation»
Data
value(): String
«annotation»
CRUD
delete(): D
readAll(): R
readOne(): R
update(): U
create(): C
proces
handler
DataHandler
DataHandler():
capitalize(String): String
init(String, String, String, List<VariableElement>, String): TypeSpec[]
packageName(TypeElement): String
generateDefaultValue(TypeMirror): CodeBlock
dataSection(TypeElement, List<VariableElement>, String): List<JavaFile>
trailingForTable(Builder): CodeBlock
KeyHandler
KeyHandler():
isAnnotated(VariableElement): boolean
clean(Object): String
crudQueries(String, List<VariableElement>): AnnotationSpec
usa
usa
process
s
TableHandler
TableHandler():
getQueries(Class<?>): Map<String, String>
EntityManager
EntityManager():
init(ProcessingEnvironment): void
check(Element): boolean
process(Set<TypeElement>, RoundEnvironment): boolean
classesBuild(): void
generate(TypeElement): void

El segundo submódulo, "api", corresponde a la capa de negocio de la aplicación, donde se realizan las operaciones transaccionales y de consulta.

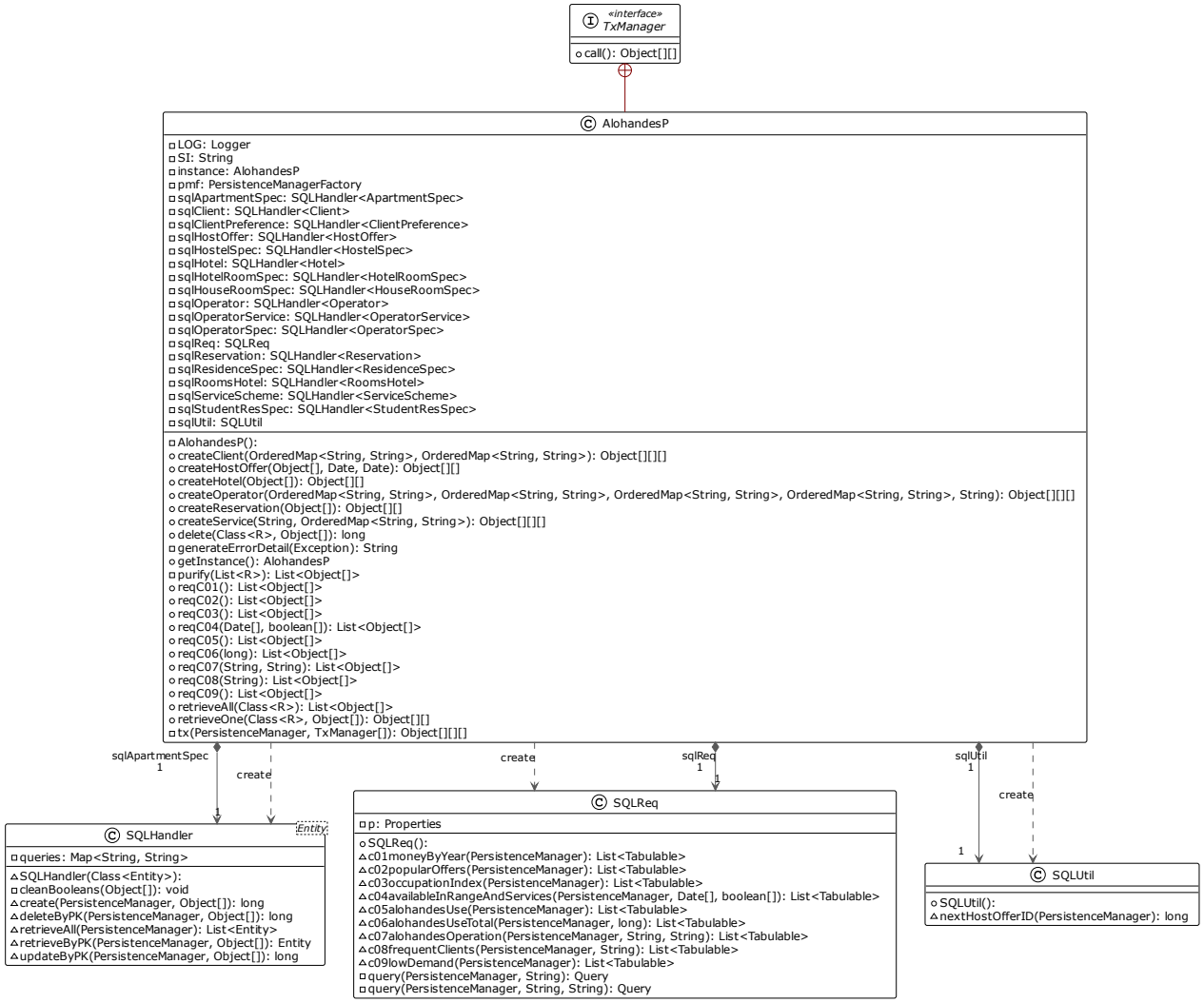


El tercer submódulo, "data", se encarga de contener los POJOs y VOs, así como los objetos de acceso a datos necesarios para la persistencia en la base de datos.

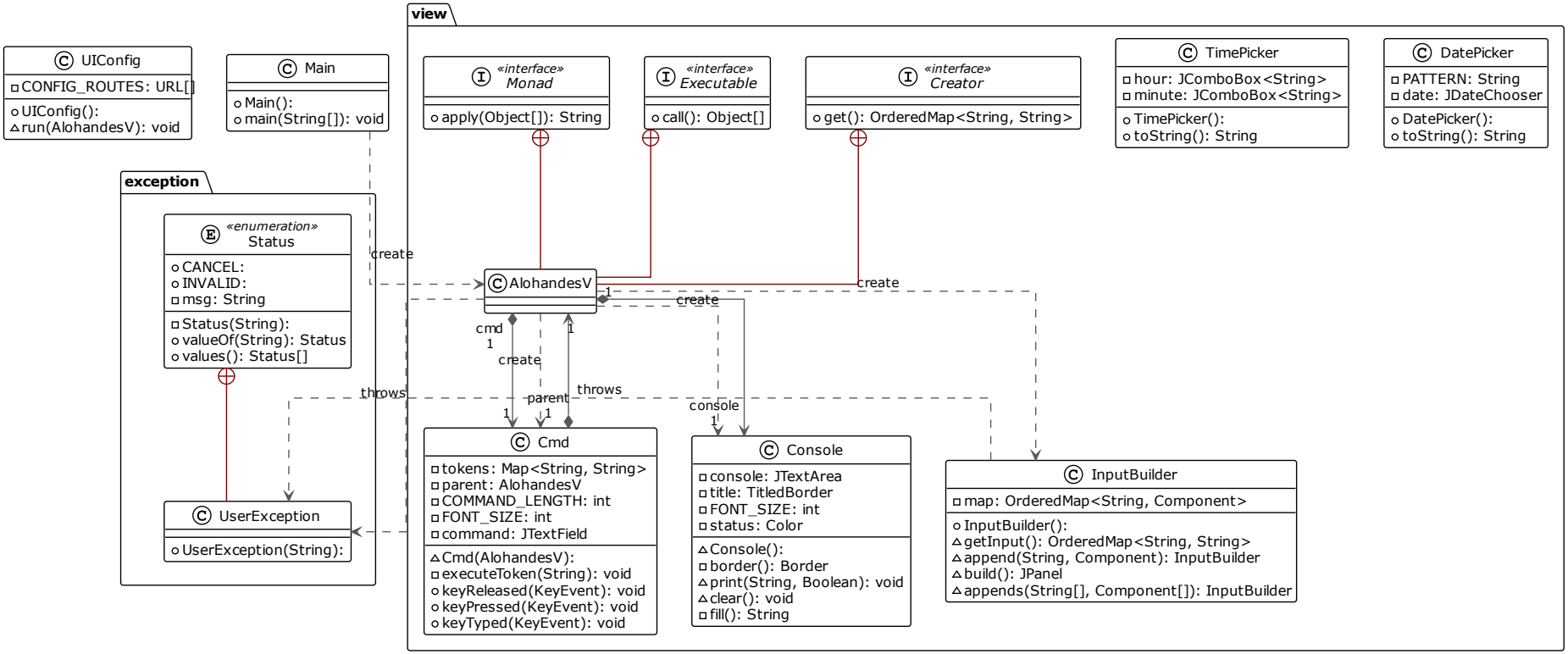



Operator
name: String
type: String
nuip: Long
Operator(Long, String, String):
Operator():
toTable(): Object[][]
toString(): String
«interface»
VOOperator
name: String
type: String
nuip: Long
toTable(): Object[][]
toString(): String
OperatorService
name: String
wifi: Boolean
furnished: Boolean
kitchenette: Boolean
OperatorService(String, Boolean, Boolean, Boolean):
OperatorService():
toString(): String
toTable(): Object[][]
«interface»
VOOperatorService
name: String
wifi: Boolean
furnished: Boolean
kitchenette: Boolean
toString(): String
toTable(): Object[][]
OperatorSpec
name: String
size: Integer
location: String
shared: Boolean
capacity: Integer
OperatorSpec(String, Integer, Integer, String, Boolean):
OperatorSpec():
toString(): String
toTable(): Object[][]
«interface»
VOOperatorSpec
name: String
size: Integer
location: String
shared: Boolean
capacity: Integer
toString(): String
toTable(): Object[][]
Reservation
id: Long
costByNight: Integer
Reservation():
Reservation(Long, Integer):
toString(): String
toTable(): Object[][]
«interface»
VOReservation
id: Long
costByNight: Integer
toString(): String
toTable(): Object[][]
ResidenceSpec
name: String
administrationFee: Integer
insuranceDesc: String
bedroomNum: Integer
ResidenceSpec(String, Integer, Integer, String):
ResidenceSpec():
toString(): String
toTable(): Object[][]
«interface»
VOResidenceSpec
name: String
administrationFee: Integer
insuranceDesc: String
bedroomNum: Integer
toString(): String
toTable(): Object[][]
RoomsHotel
nit: Integer
name: String
RoomsHotel(Integer, String):
RoomsHotel():
toString(): String
toTable(): Object[][]
«interface»
VORoomsHotel
nit: Integer
name: String
toString(): String
toTable(): Object[][]
ServiceScheme
name: String
serviceName: String
serviceCost: Integer
ServiceScheme():
ServiceScheme(String, String, Integer):
toString(): String
toTable(): Object[][]
«interface»
VOServiceScheme
name: String
serviceName: String
serviceCost: Integer
toString(): String
toTable(): Object[][]
StudentResSpec
name: String
studyRoom: Integer
restaurant: Integer
gym: Integer
recreationRoom: Integer
StudentResSpec():
StudentResSpec(String, Integer, Integer, Integer, Integer):
toTable(): Object[][]
toString(): String
«interface»
VOStudentResSpec
name: String
studyRoom: Integer
restaurant: Integer
gym: Integer
recreationRoom: Integer
toString(): String
toTable(): Object[][]

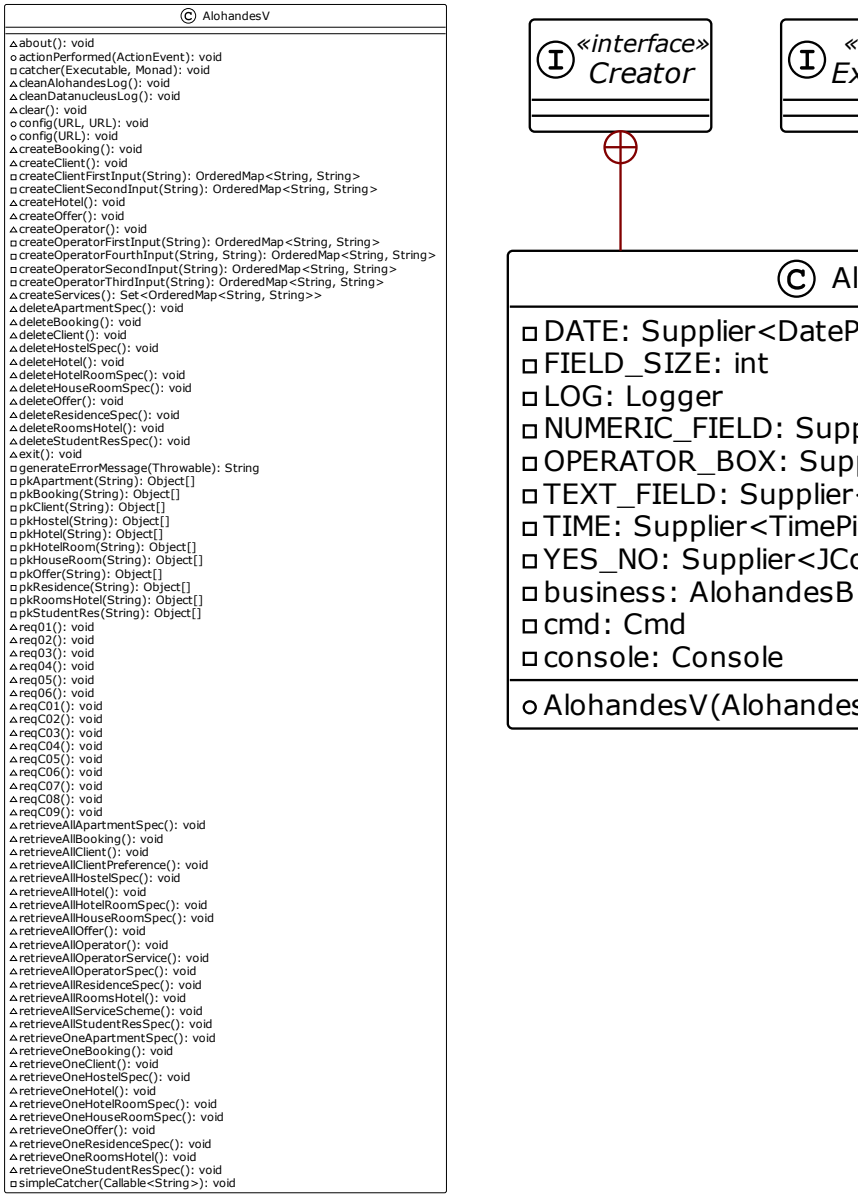
El cuarto submódulo, "persistence", se encarga de la capa que utiliza las queries y las operaciones de base de datos, garantizando la coherencia y consistencia de los datos almacenados.



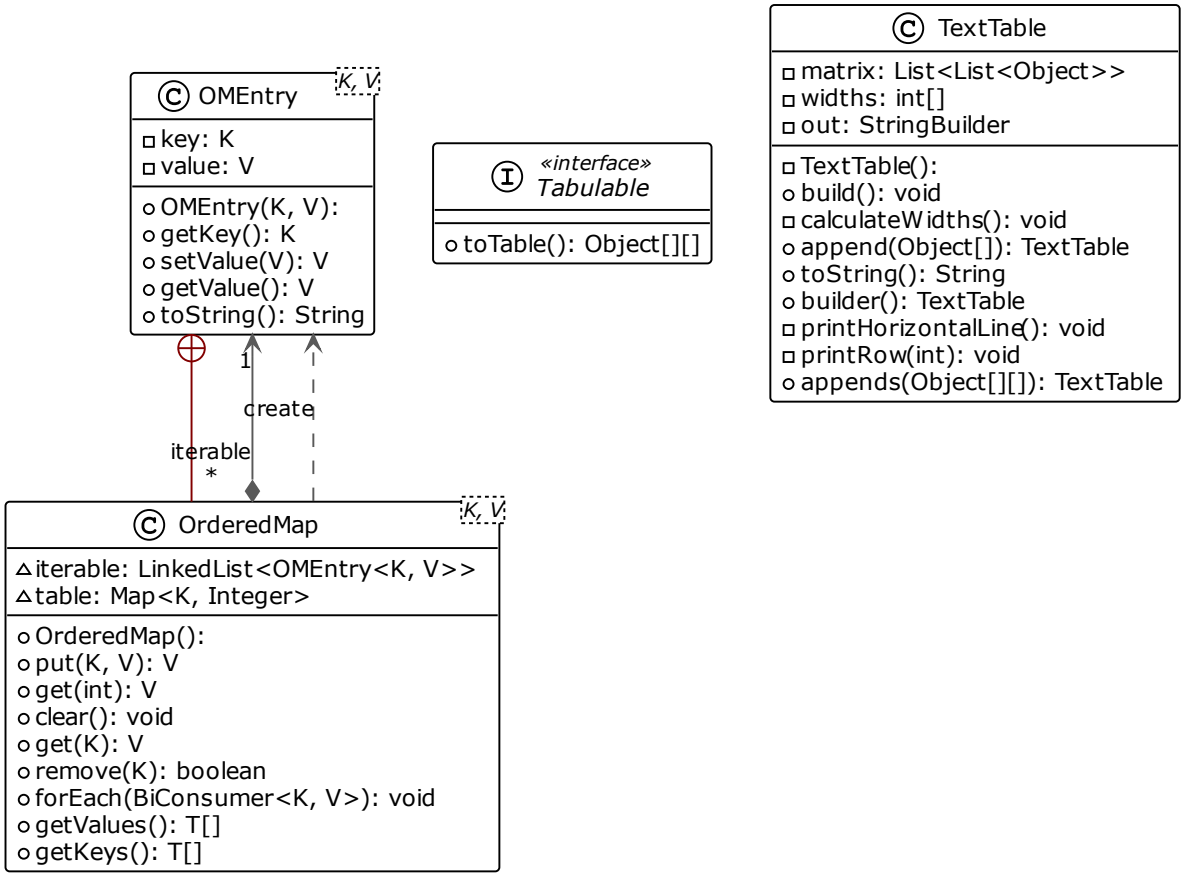
El quinto submódulo, "ui", se encarga de la interfaz de usuario, donde se realizan las operaciones de entrada y salida de datos, y se muestran los resultados de las consultas y operaciones realizadas.
Vista general del submodulo:



Vista de AlohandesV (Campos y constructor / El resto de metodos):

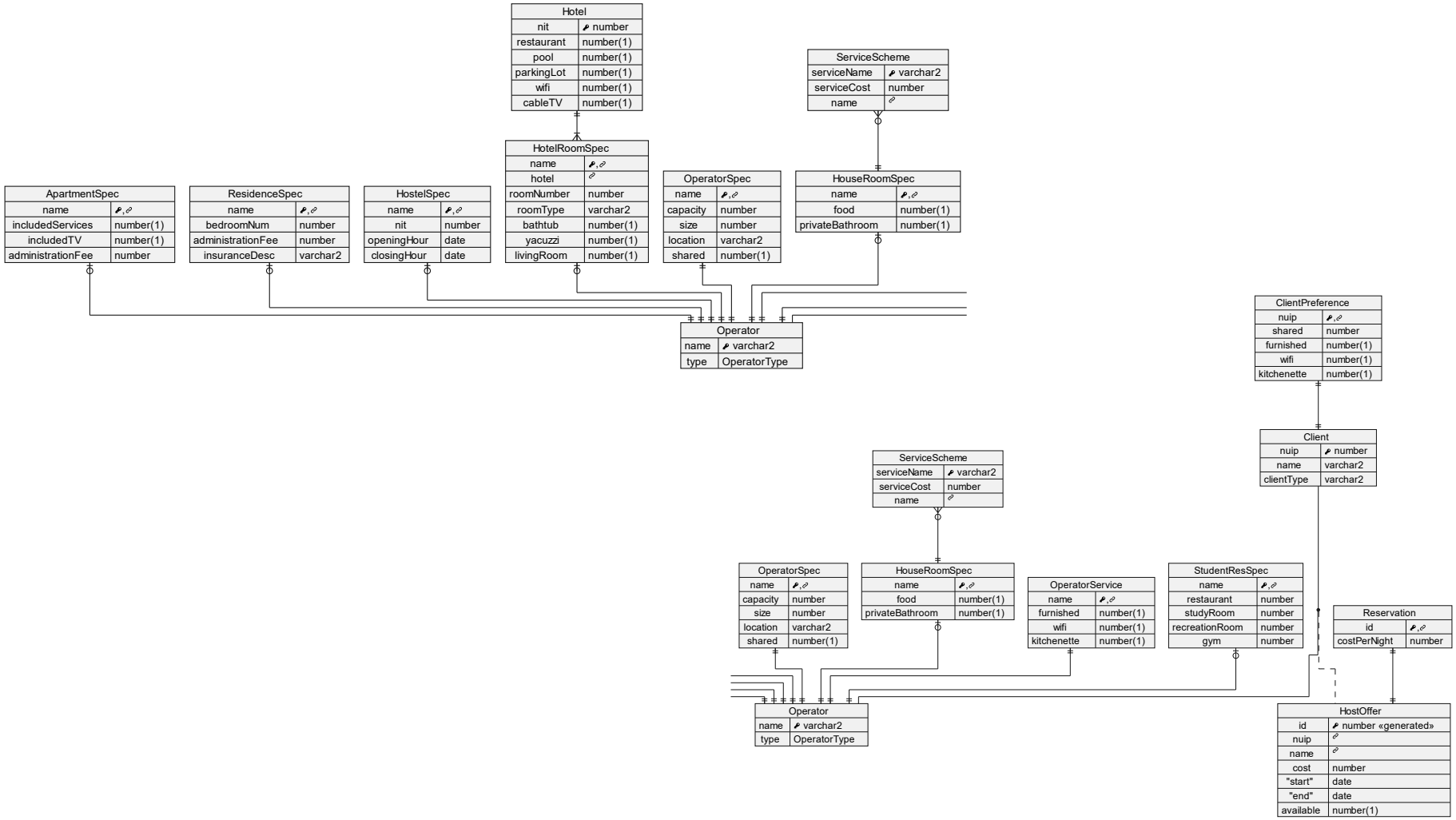


Por último, el submódulo "utils" contiene una serie de clases que otorgan utilidad y facilitan las operaciones para las otras capas.



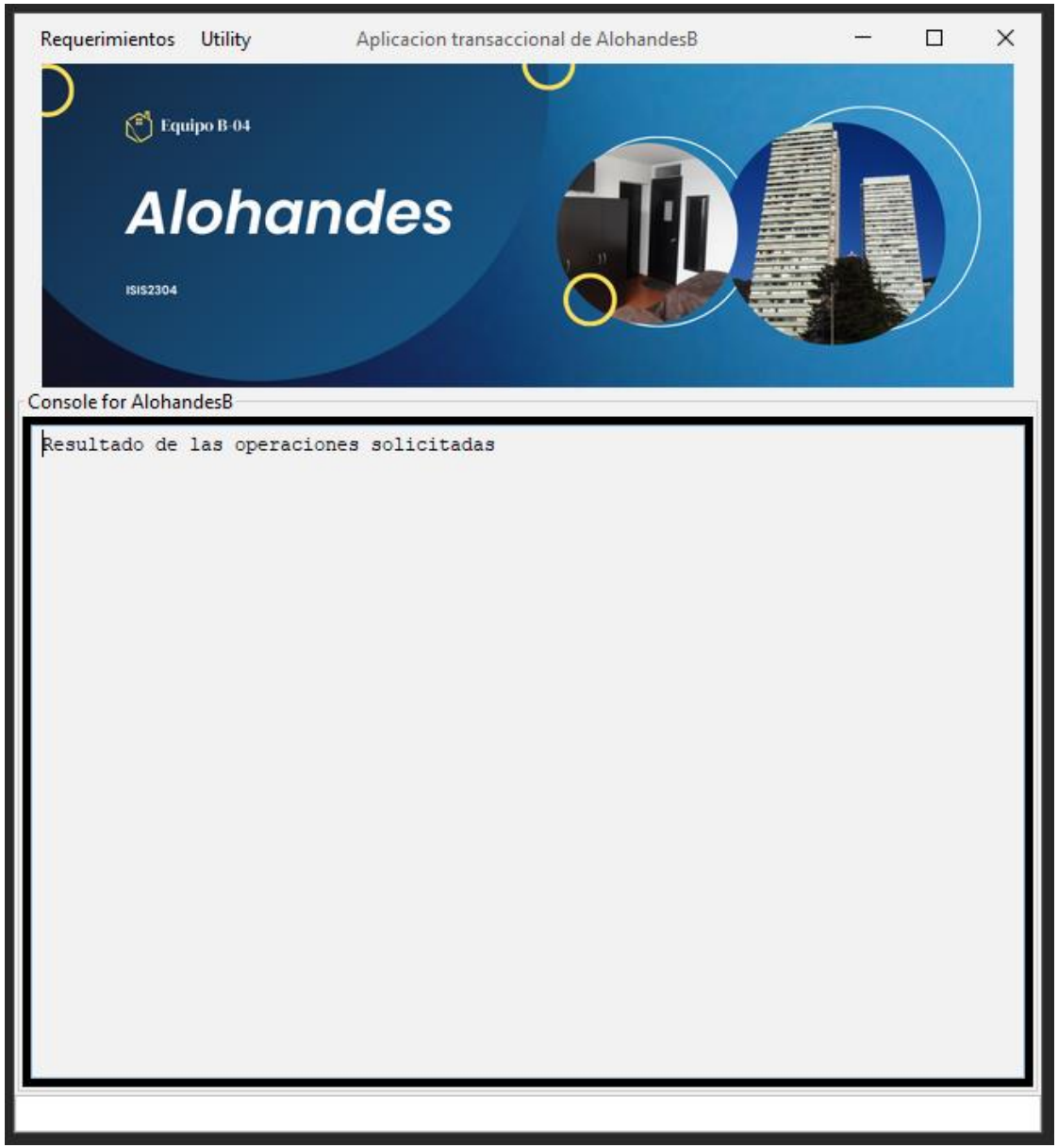
Cada submódulo se ha diseñado y desarrollado de manera independiente, pero trabajando en conjunto para cumplir con los requerimientos funcionales y de consulta del proyecto. A continuación, se presentarán seis diagramas UML que describen cada uno de estos submódulos en detalle.

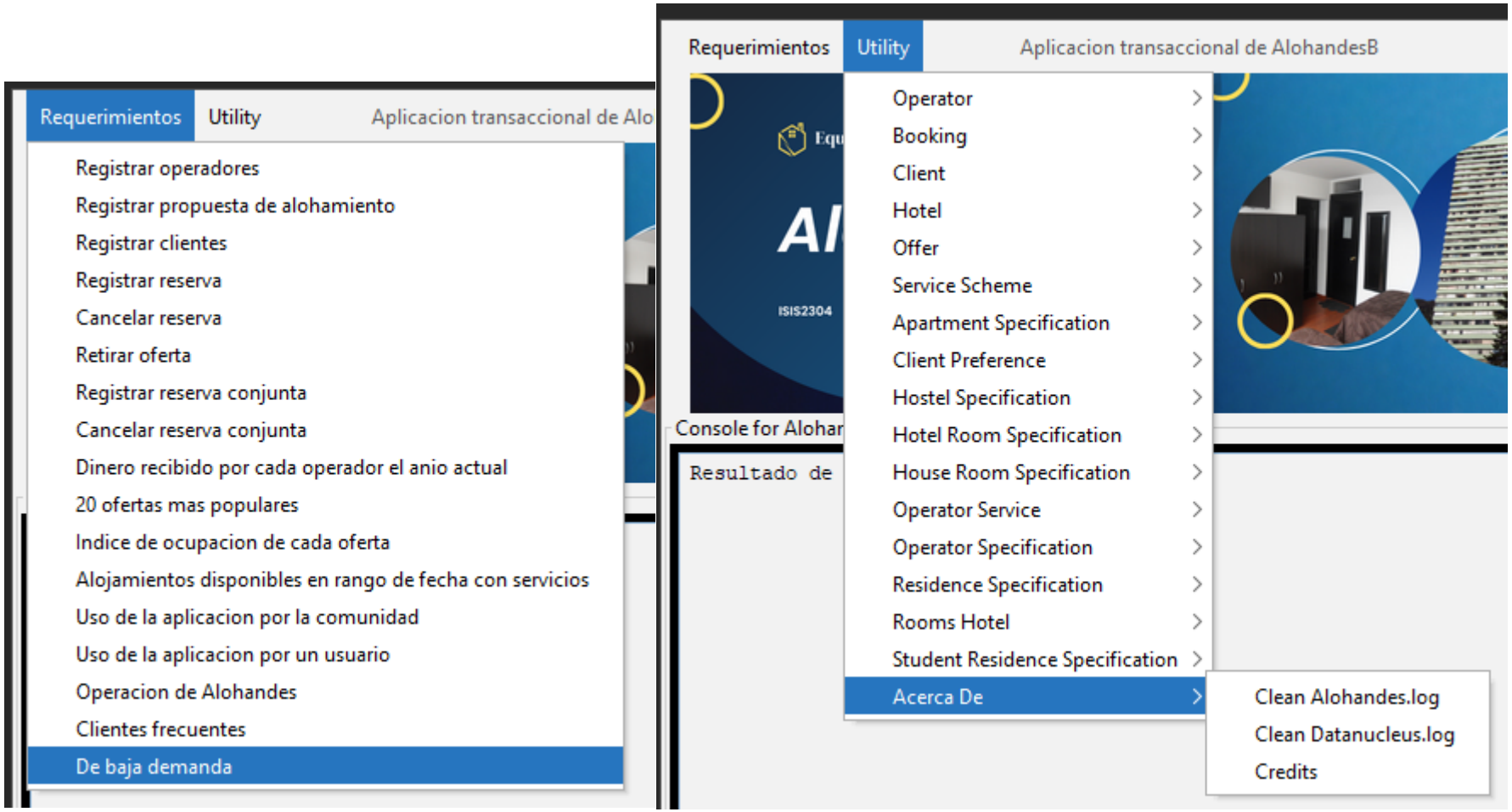
2.2 Modelo relacional

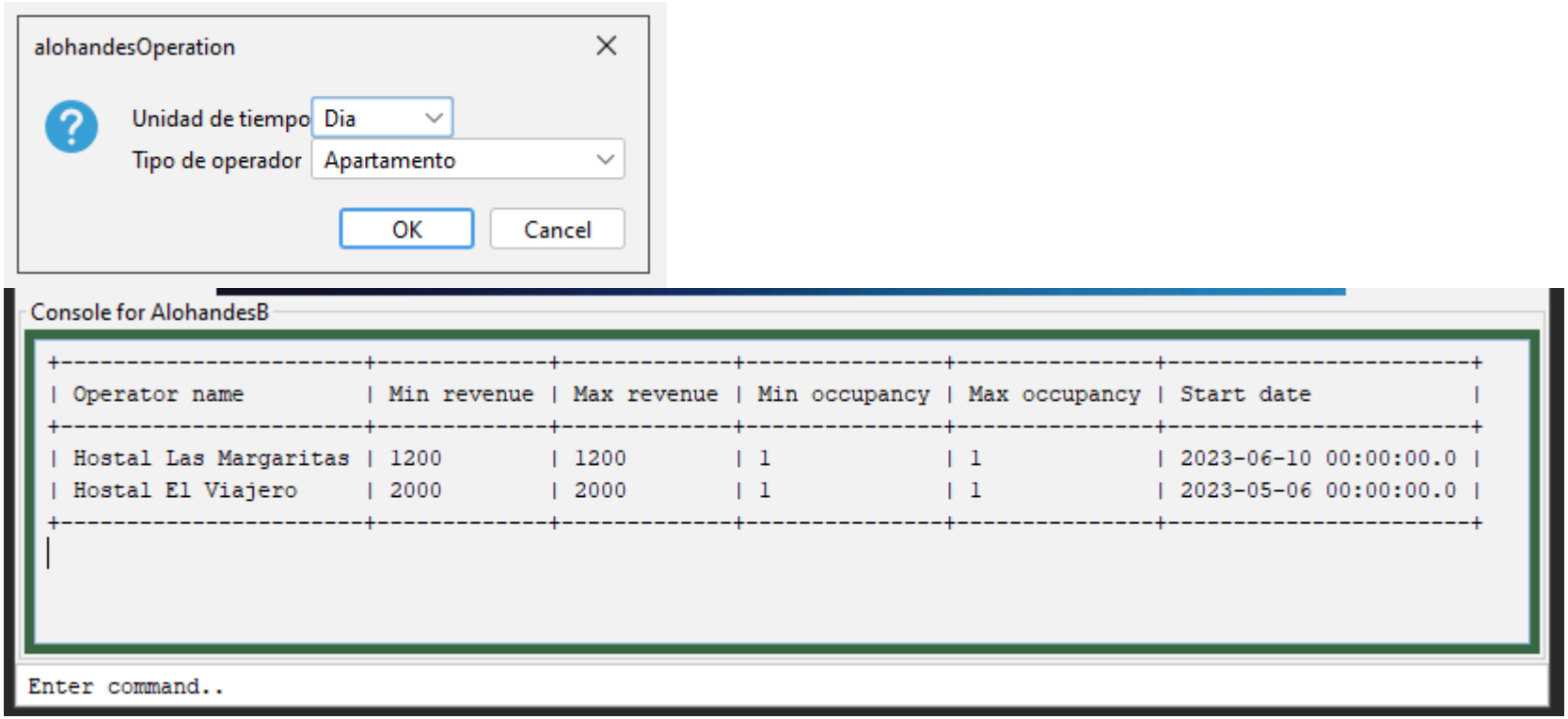

Hotel
nit number
restaurant number(1)
pool number(1)
parkingLot number(1)
wifi number(1)
cableTV number(1)
ServiceScheme
serviceName varchar2
serviceCost number
name
HotelRoomSpec
name
hotel
roomNumber number
roomType varchar2
bathtub number(1)
yacuzzi number(1)
livingRoom number(1)
OperatorSpec
name
capacity number
size number
location varchar2
shared number(1)
HouseRoomSpec
name
food number(1)
privateBathroom number(1)
ApartmentSpec
name
includedServices number(1)
includedTV number(1)
administrationFee number
ResidenceSpec
name
bedroomNum number
administrationFee number
insuranceDesc varchar2
HostelSpec
name
nit number
openingHour date
closingHour date
Operator
name varchar2
type OperatorType
ClientPreference
nuip
shared number
furnished number(1)
wifi number(1)
kitchenette number(1)
Client
nuip number
name varchar2
clientType varchar2
ServiceScheme
serviceName varchar2
serviceCost number
name
OperatorSpec
name
capacity number
size number
location varchar2
shared number(1)
HouseRoomSpec
name
food number(1)
privateBathroom number(1)
OperatorService
name
furnished number(1)
wifi number(1)
kitchenette number(1)
StudentResSpec
name
restaurant number
studyRoom number
recreationRoom number
gym number
Reservation
id
costPerNight number
Operator
name varchar2
type OperatorType
HostOffer
id number «generated»
nuip
name
cost number
"start" date
"end" date
available number(1)

3 Resultados

A continuación, se muestra el funcionamiento de la aplicación y cada requerimiento (Con información cargada previamente para los de consulta)


Requerimientos
Utility
Aplicacion transaccional de AlohandesB
Equipo B-04
Alohandes
ISIS2304
Console for AlohandesB
Resultado de las operaciones solicitadas


Requerimientos
Utility
Aplicacion transaccional de Alo
Registrar operadores
Registrar propuesta de alohamiento
Registrar clientes
Registrar reserva
Cancelar reserva
Retirar oferta
Registrar reserva conjunta
Cancelar reserva conjunta
Dinero recibido por cada operador el anio actual
20 ofertas mas populares
Indice de ocupacion de cada oferta
Alojamientos disponibles en rango de fecha con servicios
Uso de la aplicacion por la comunidad
Uso de la aplicacion por un usuario
Operacion de Alohandes
Clientes frecuentes
De baja demanda
Requerimientos
Utility
Aplicacion transaccional de AlohandesB
ISIS2304
Console for Alohar
Resultado de
Operator
Booking
Client
Hotel
Offer
Service Scheme
Apartment Specification
Client Preference
Hostel Specification
Hotel Room Specification
House Room Specification
Operator Service
Operator Specification
Residence Specification
Rooms Hotel
Student Residence Specification
Acerca De
Clean Alohandes.log
Clean Datanucleus.log
Credits


alohandesOperation
Unidad de tiempo
Dia
Tipo de operador
Apartamento
OK
Cancel
Console for AlohandesB
+----------------------+-------------+-------------+---------------+---------------+----------------------+
| Operator name | Min revenue | Max revenue | Min occupancy | Max occupancy | Start date |
+----------------------+-------------+-------------+---------------+---------------+----------------------+
| Hostal Las Margaritas | 1200 | 1200 | 1 | 1 | 2023-06-10 00:00:00.0 |
| Hostal El Viajero | 2000 | 2000 | 1 | 1 | 2023-05-06 00:00:00.0 |
+----------------------+-------------+-------------+---------------+---------------+----------------------+
Enter command..