Programación/Laboratorio 3

Trabajo Practico FINAL



**Integrantes**: Agustin Dominguez - Azul Gottero - Agustin Caceres

**Asignatura**: Técnico Superior en Programación

Facultad regional UTN Mar del Plata Unidad Académica

Mar del Plata 26 de junio de 2018

1. **Resumen**

Se llevo a cabo el desarrollo de una aplicación para la gestión y manejo de un hotel, que tiene como objetivo garantizar la organización del mismo, desde que el cliente ingresa y se retira del establecimiento.

Se empezó el proyecto trabajando con una lista, tanto de clientes como empleados (incluyendo administradores), donde los últimos nombrados, tendrán la función de realizar checkin y checkout del pasajero que asista al hotel, una vez que tenga la correspondiente reserva.

Por otra parte, los administradores, serán los encargados de la creación de empleados y pasajeros en el sistema, y de otras funciones como realizar un backup de los datos del hotel.

1. **Informe Técnico**

Se utilizaron diversas clases para organizar el proyecto, las cuales son:

Tipos de usuario:

* **Persona** (contiene nombre y apellido, y por eso, es la clase padre de pasajero, admin y recepcionista, ya que comparten esos datos).
* **Pasajero** (Es la persona que ocupa la habitación. Contiene datos necesarios para registrarse en el hotel, como lo son el teléfono, dirección, email y DNI. En cuanto a la creación de los mismos, se crearon 20 pasajeros con sus respectivos datos en una lista del tipo correspondiente, para su serialización y deserialización).
* **Administrador** (Su objetivo es realizar funciones administrativas del sistema como la creación de usuarios, ya sea cliente o empleado, y, en caso de ser necesario, realizar un backup en un archivo).
* **Recepcionista** (Es el encargado de todo tipo de operación para con el cliente, estas incluyen, crear reservas (que incluye tanto la búsqueda de la misma como de la habitación correspondiente), registrar checkin y checkout (cambia el estado de la habitación).

**Habitación**: (Esta clase representa los cuartos del hotel, que tendrán como datos, un numero de habitación, un **Tipo** y un **Estado**).

* **Tipo** (Es un enum, cuyos posibles valores son: INDIVIDUAL, DOBLE, TRIPLE O SUITES).
* **Estado** (Es un enum creado como clase, cuyos posibles valores son: DISPONIBLE, NO\_DISPONIBLE Y OCUPADO).

**Hotel**: (En esta clase, se van a definir los datos que caracterizan al hotel, como el nombre, cantidad de estrellas y contacto (email-teléfono).

**Reserva**: (Esta clase nos genera un numero de reserva para el pasajero(aleatorio), la entrada y salida del mismo del hotel, y la habitación a ocupar. La entrada será la fecha al llamar la función de alquiler, mientras que la salida será de entre 1 semana y 1 mes.).

**Main**: (Implementación de menú (Recepcionista, Admin, Pasajero), cada uno con sus correspondientes funciones.

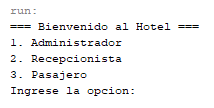
**Exceptions**: Se utilizaron 2 Exceptions para este programa, los cuales son:

* **PersonaNoEncontradaException**
* **PersonaDuplicadaException**

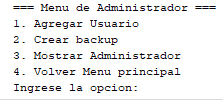
**Utilización de la herramienta GIT, para mayor organización.**

**(github.com/azulws/TpFinalLab3)**

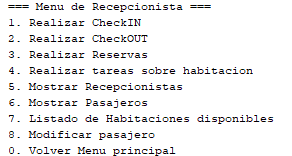
1. **Manual de Usuario**



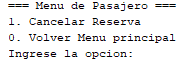
1. **Administrador:** Agrega usuarios, crea backup y muestra los admin creados.
2. **Recepcionista:** Realiza checkin-checkout, reservas, muestra los recepcionistas creados y el listado de habitaciones disponible.
3. **Pasajero:** Solamente podrá cancelar la reserva**.**



1. **Agregar usuario:** Esta función, permite crear usuarios de cualquier tipo (admin, pasajero, recepcionista) y los almacena en una lista del tipo al que corresponda.
2. **Crear backup: Esta función, invoca a “serializar” que nos pregunta si deseamos hacer un backup de pasajeros o empleados, y, utilizando la librería de Gson, serializa la lista que le indiquemos.**
3. **Mostrar Administrador:** Como su nombre lo indica, muestra la lista de administradores.
4. **Volver al menú principal:** Volvemos al menú principal del hotel**.**



1. **Realizar CheckIN:** Cambia el estado de la habitación que pasemos por parámetro, de Disponible a Ocupada**.**
2. **Realizar CheckOUT:** Cambia el estado de la habitación que pasemos por parámetro, de Ocupada a No-Disponible.
3. **Realizar Reservas:** Realiza la reserva de una determinada habitación, y se la asigna al pasajero, utilizando su DNI.
4. **Realizar tareas sobre habitación:** Realiza operaciones de mantenimiento (Limpieza-Desinfección-Reparaciones) sobre una habitación determinada, antes de poner la misma en estado Disponible.
5. **Mostrar Recepcionistas:** Muestra los recepcionistas de la lista.
6. **Mostrar Pasajeros:** Muestra los pasajeros de la lista y los nuevos que vaya agregando el administrador.
7. **Listado de habitaciones disponibles**: Nos redirecciona al menú de habitaciones.
8. **Modificar pasajero**: Modifica los datos de los pasajeros, en caso de que se hayan cargado mal en el sistema.
9. **Volver al Menú principal:** Volvemos al menú principal del hotel**.**



1. **Cancelar Reserva:** Cancela la reserva del pasajero.
2. **Volver al Menú principal:** Volvemos al menú principal del hotel**.**
3. **Errores**

* Excepcion nullpointerexception.
* Error al invocar función de alquiler en el menú.

1. **Matriz de Soluciones**

* Rehacer los constructores de las clases en la que estaban los métodos que nos tiraban esta excepción.
* Para solucionar este error tuvimos que crear una reserva donde guardamos el alquiler para luego pasarlo a lista de reservas.

1. **Diario de Trabajo**

* **Semana 1: Comenzamos con los constructores, los métodos getter y setter y realizamos el UML del trabajo practico.**
* **Semana 2: Creamos las clases y programamos los primeros métodos como son los diferentes buscar, ver si el estado de una habitación es disponible y el mostrar habitaciones disponibles.**
* **Semana 3: Se programó las últimas funciones del programa el checkin, checkout y el alquiler de habitaciones (reservas), se probó las funciones del programa, comenzamos a programar el menú.**
* **Semana 4: Terminamos de programar el menú y las funciones adicionales, que son utilizadas en el menú.**

1. **Bibliografía**

[**https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html**](https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html)

[**https://java2blog.com/gson-example-read-and-write-json/**](https://java2blog.com/gson-example-read-and-write-json/)

[**https://stackoverflow.com/questions/19459082/read-and-write-data-with-gson**](https://stackoverflow.com/questions/19459082/read-and-write-data-with-gson)

[**https://futurestud.io/tutorials/gson-model-annotations-how-to-ignore-fields-with-expose**](https://futurestud.io/tutorials/gson-model-annotations-how-to-ignore-fields-with-expose)

[**https://stackoverflow.com/questions/11486764/error-casting-typetoken-to-type-when-deserializing-with-gson**](https://stackoverflow.com/questions/11486764/error-casting-typetoken-to-type-when-deserializing-with-gson)

[**https://futurestud.io/tutorials/gson-mapping-of-arrays-and-lists-of-objects**](https://futurestud.io/tutorials/gson-mapping-of-arrays-and-lists-of-objects)

[**http://pragmateek.com/javajson-mapping-with-gson/**](http://pragmateek.com/javajson-mapping-with-gson/)