LABORATORIO DE COMPUTACI ÓN **III**

*“RESTO-GRAM”*



Desarrolladores:

Benvenuti Mariana -

Bertuccio Nicolas -

Fullone Lautaro -

Perez Rodrigo -

Ciclo 2020

***INFORME TÉCNICO***

Nuestro sistema está orientado a la organización y funcionamiento diario de un establecimiento gastronómico en la ciudad de Mar del Plata.

Esta idea surgió luego de varias conversaciones entre los diferentes integrantes del grupo. Quienes, desde su punto de vista, aportaron diversos factores para llegar a establecer esta idea como la que se terminó realizando.

En un principio, la idea se baso en desarrollar tanto la parte organizacional como también administrativa (contable) del lugar. Pero, finalmente solo se desarrolló la primer parte, dejando lo demás para una posible futura expansión del sistema.

Diseño de Clases

En cuanto al diseño de clases, podemos comenzar por el pilar del proyecto. En lo más alto del sistema tenemos a la clase RESTAURANT, la cual posee en su estructura a las demás clases. Esta contiene las siguientes cualidades:

* LISTA CLIENTES: Es un listado de CLIENTES (extiende de PERSONA), ya sean VIRTUALES o PRESENCIALES. En este caso se utilizó la colección *ArrayList<CLIENTES>.*
* LISTA MESAS: Es un listado de MESAS, lugar el cual será ocupado por los CLIENTES PRESENCIALES. En este caso se utilizó la colección *ArrayList<MESAS>.*
* CARTA COMIDAS: Es una clase la cual posee dos colecciones como atributos. Por un lado, tenemos LISTA DE COMIDAS y por el otro SET DE COMBOS.
* LISTA COMIDAS: Es un listado de COMIDAS, que pueden ser comidas en si, como PLATO PRINCIPAL, BEBIDA, GUARNICION o POSTRE. En este caso se utilizó la colección *ArrayList<COMIDAS>.*
* SET COMBOS: Es un listado de COMBOS, es decir un conjunto de COMIDAS que forman cada COMBO. En este caso se utilizó la colección *HashSet<COMBOS>.*

En cada uno de esos casos de utilizo la colección que pensamos mejor adecuarse a las necesidades que el proyecto demandaba. De igual manera el sistema esta modularizado de cierta manera que el cambio de una de estas colecciones por otra, no debería significar una gran complicación.

Interfaces

En este sistema solo se aplicaron dos sencillas interfaces, pero no por eso poco relevantes.

Una de ellas es **IOpBasicas** la cual, siendo genérica, encapsula los métodos principales de todas las listas de objetos del sistema (listar, agregar y eliminar).

En el otro extremo tenemos a **IPrecio** que cuenta con un único método, que se aplica en las diversas clases que la implementan, con el objetivo de retornar el valor/precio/costo de un producto y/o pedido.

Excepciones

El sistema cuenta con una larga fila de excepciones muy variadas. Listadas a continuación y con su respectiva documentación en el archivo JavaDoc.

* **CapacidadMaximaException**
* **ClienteInexistenteException**
* **ComboNoExistenteException**
* **ComidaInexistenteException**
* **DniExistenteException**
* **DniNOExistenteException**
* **IdInexistenteMesaException**
* **IngresoInvalidoException**
* **MesasLibresException**
* **PrecioNegativoException**
* **SentarPersonaException**
* **SinMesasException**

Archivos

Para este caso solo se vio necesario utilizar dos clases “archivadoras” para todo el sistema.

* Clase ARCHIVOS, encargado de almacenar en memoria los datos/información de la clase RESTAURANT. Esta consta de ocho métodos estáticos uno de lectura y otro de escritura para cada una de las cuatro colecciones de objetos (Lista de Mesas, Lista de Comidas, Set de Combos y Lista de Clientes).
* Clase JSONUTILES, esta clase almacena en un archivo de texto plano toda la información recolectada por el sistema JSON del proyecto.

Uso de Herramientas Complementarias

Durante el transcurso del desarrollo de este proyecto utilizamos diferentes herramientas de comunicación/programación. Entre ellas destacamos las siguientes aplicaciones:

* **GITHUB**: Esta fue la herramienta complementaria base para la codificación del sistema. Esta nos permitió hacer un “feedback” de los distintos avances, compartidos por cada integrante del equipo.

Link: [*https://github.com/benvenutim93/Laboratorio-III*](https://github.com/benvenutim93/Laboratorio-III)

* **TRELLO**: Este tablero de tarjetas nos permitió organizar el día a día del desarrollo, llevando un diario semanal de las tareas. Nos facilitó la organización de las cosas hechas/por hacer desde el primer hasta el último día.

Link: [*https://trello.com/b/gLOKL82p/laboratorio-iii*](https://trello.com/b/gLOKL82p/laboratorio-iii)

* **DISCORD**: Todo el trabajo de codificación se realizó en conjunto y a distancia, mediante el uso de la aplicación. Se trabajo compartiendo pantalla en simultaneo, mientras se comentaba sobre las diferentes tareas a realizar.
* **DRAWIO**: Herramienta utilizada para la creación del archivo UML del sistema.

Link: *https://www.draw.io/*