Facultad de Ingeniería

Universidad Nacional de Mar del Plata

Sistemas de contratación y servicios de seguridad

Cátedra: Programación III

Grupo 3

Estudiantes:

- Arias Iñaki
- Arias Pedro
- Delgado Facundo
- Rasso Micaela

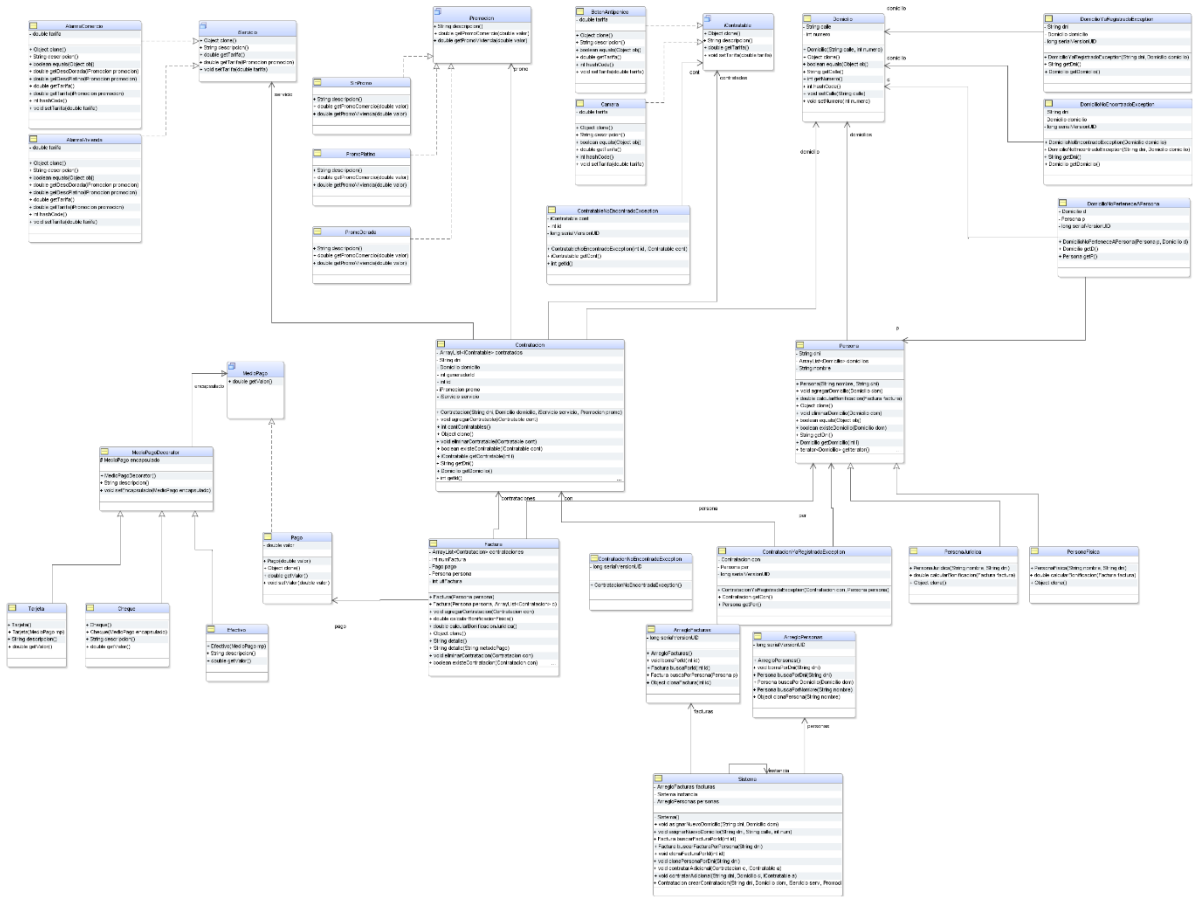
URL: Repositorio GitHub

Análisis de las soluciones elegidas

En primer lugar, se planteó un sistema de clases que incluyese todas las funcionalidades pedidas por la consigna del trabajo práctico. A medida que se avanzó en la escritura del código y la implementación de los algoritmos, se modificó esta primera estructura de manera que pudiese manejar situaciones que no se habían tenido en consideración en un principio.

En lugar de seguir heredando clases, decidimos implementar unas interfaces que llamamos *iContratable*, *iServicio*, *iMedioPago* e *iPromocion*. Nos pareció más práctico hablar de interfaces en vez de herencia, porque nos permitía hacer las cosas más intuitivamente. A través de ellas se pudo crear el primer bosquejo del sistema. Así, logramos tener una estructura más dinámica y flexible.

De esta manera, llegamos a la estructura siguiente:



Vista general de las clases del sistema, a través de diagrama UML
URL: DiagramaUML

En la consigna del informe se pedía implementar ciertas características o funcionalidades mediante distintos patrones de diseño. Por lo tanto, para la aplicación de promociones sobre los servicios contratados, se utilizó **Double Dispatch** y para la modificación del valor total de la factura por el medio de pago, se utilizó el patrón **Decorator**.

Para la clase *Sistema* se optó por que siga el patrón **Singleton** ya que no se necesita declarar más instancias de ese tipo. Una de las funciones de la clase sistema es crear las instancias de *Persona*, *Factura*, *Contratación* y *Contratables*. En la próxima entrega planeamos incluir el patrón **Factory** para encapsular esta funcionalidad y delegar responsabilidades en cada clase.

Aspectos más relevantes

Implementación de excepciones

A través de toda la estructura del código creado, se realizó la implementación de excepciones particulares y descriptivas de cada posible error. De esta manera se posibilita una mejor comprensión de los problemas en la ejecución del programa. Una de las excepciones, por ejemplo, se lanza cuando un domicilio no registrado a nombre de una persona quiere incluirse en una factura. Mediante los carteles y los parámetros de la misma se puede identificar el dni de la persona involucrada y cuál es el domicilio que causó el lanzamiento de la excepción.

Clonación

Al implementar la clonación de objetos, se tuvo en cuenta la relación que cada objeto tenía dentro de la clase. De esta manera, pudimos realizar el tipo de clonación que correspondía a cada objeto en particular.

Clases de arreglos

Se crearon clases que heredan del tipo *ArrayList* de *Persona* y *Factura* para poder hacer consultas buscando por Id's en el caso de *Factura* y por campo dni en el caso de la clase *Persona*, lo cual facilita el acceso por parte del sistema a objetos específicos con los que se quiera realizar operaciones.

Problemas

Utilización de Git y Github

En el desarrollo del proyecto, se tuvo dificultades al utilizar Git y Github, debido a que ninguno de los miembros del equipo estaba familiarizado con la plataforma. Como resultado, nos encontramos en la situación de trabajar de forma secuencial, esperando a que otro compañero hiciera push a la rama principal (Main), lo que ha retrasado significativamente el progreso del desarrollo de nuestra aplicación. A partir de esta experiencia, nos dimos cuenta de la importancia de tener un buen conocimiento de las herramientas que ofrecen estas tecnologías. Por lo tanto, hemos acordado dedicar tiempo a familiarizarnos más con ellas, para evitar futuros retrasos en nuestro proyecto.

Expectativas

En cuanto a las expectativas para la próxima entrega de este informe, el grupo espera poder implementar el patrón **Factory** en las clases necesarias. Además, se espera que, tras haber aprendido más sobre Git y Github, se pueda trabajar de forma más colaborativa y fluida en el

desarrollo del proyecto. En términos generales, el objetivo es seguir mejorando la estructura y funcionalidades de la aplicación, y garantizar una ejecución eficiente y sin errores a través de la implementación de excepciones y la clonación adecuada de objetos.