Sistema de Contratacion de Servicios de Seguridad

INTEGRANTES:

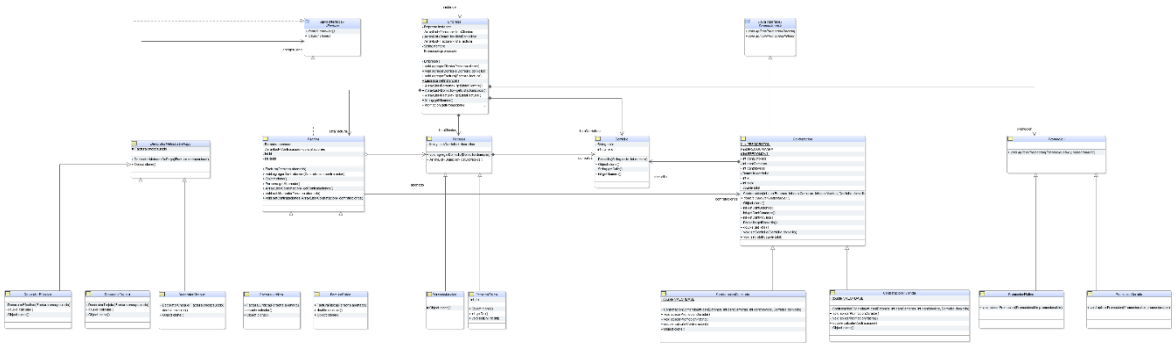
-Gonzalo Leon

-Tomas Civiero

-Ignacio Cantando Flego

-Gabriel Echeverria

ANALISIS BREVE: Antes de haber empezado con el trabajo, lo primero y más importante que se hizo fue la planificación del trabajo, como debería armarse, como debe funcionar, que resultados debemos obtener y que errores pueden encontrarse.



En la imagen se puede destacar el uso de herencias múltiples, interfaces y patrón decorator.

Las herencias para no tener que escribir 2 o 3 veces las mismas características de las clases.

La implementación de interfaces para solo usar métodos que comparten varias clases.

El uso de patrón decorator para una mejor elección de una clase.

Una vez hecho el esquema, recién se podía empezar a escribir el código.

ASPECTOS RELEVANTES:

El uso de doublé dispatch en las promociones nos permite enviar un mensaje para ver que promoción usar, para recibir como respuesta la promoción que se utilizara para el servicio.

para el uso de patrón decorator, nos enfocamos en los métodos de pago (Cheque, Tarjeta, Efectivo), debido a las cantidades de posibilidades que puede haber con las facturas:

- Tendremos 2 tipos de facturas, dependiendo de si la persona es física o jurídica, y además tendremos 3 formas de pagar. Se tendría 6 clases hijos, una persona física con 3 métodos de pago, y una jurídica con 3 métodos de pago. Para su sencillez sin sacrificar el funcionamiento, separamos facturas por un lado y los métodos por otro.
- Además, el patrón decorator permite extender la funcionalidad en tiempos de ejecución sin depender de la herencia y a la vez se implementa una interfaz en vez de un objeto concreto, permitiendo más flexibilidad. De esta forma se consiguen aplicar 2 principios de diseño de SOLID:
    - Principio de inversión de dependencias al depender de una abstracción (interfaz) en vez de algo concreto.
    - Principio Abierto-Cerrado al estar abierto a la extensión, pero cerrado a la modificación, en este caso la clase Factura.

El uso de una interface Ifactura permite que se pueda usar el método calcular() tanto en facturas como en métodos de pago para obtener un resultado.

Excepciones: Un detalle a notar es que las únicas excepciones que existen son para el caso de que exista persona jurídica, y para la factura correspondiente a la persona juridica.

DIFICULTADES Y COMO SE RESOLVIERON:

Se discutió en grupo ideas sobre cómo resolver el tema, utilizando UML se discutieron la forma de relacionar las clases para crear un sistema flexible, fácil de mantener y que a la vez cumplía con las condiciones dadas por la cátedra.

La mayor dificultad del trabajo proviene desde el principio; la planeación.

La principal fue el uso del patrón decorator y su relación con las facturas y los metodos de pago. Un mal uso provocaría que termináramos con un programa muy complejo. Como se vio antes, se resolvió separando los métodos de pago con la factura.

Otra cosa que complico un poco fue el concepto de servicios, es decir, si el agregado de cámaras, botones o móviles contaba como un servicio. La principal incógnita se debía a una de las condiciones de la factura jurídica (a partir del tercer servicio, se le aplica un descuento). Se concluyó que incluir estos como un servicio mas no solo complicaría más las cosas, sino que tampoco tendría mucho sentido en cuanto a promociones.

Una vez se tenía una imagen clara de cómo debería funcionar el programa se pudo continuar con normalidad con el trabajo hasta su finalización.