Universidad Simón Bolívar, Sede Sartenejas.
República Bolivariana de Venezuela.
CI-3115 Ingeniería de Software.
Modalidad Semi-Intensiva: Septiembre-Octubre 2013.

INFORME: TAREA Nº 4

Profesor:
Carina Ferreira.

Elaborado por:
10-10108 Marcos Campos.
10-10196 John Delgado.
10-10239 Luis Fernandes.
10-10368 Oswaldo Jiménez.
10-10385 Gabriela Limonta.
10-10469 José Montenegro.
10-10666 Andrea Salcedo.
10-10812 Fernando D'Agostino.

Caracas, Sartenejas. 07 de Ocubre de 2013.

1 Índice

3 Introducción

El presente documento tiene como objetivo presentar un nuevo diseño con su respectiva funcionalidad de Software orientado por objetos desarrollado por el equipo fundador de INNOVA. Este software que se presenta como entrega mejorada de la tarea anterior no presenta cambios en cuanto a la base de datos se refiere, por lo que esta vez, todo lo relacionado a base de datos queda de lado y se hace un enfoque de gran amplitud en cuanto al diseño del software y la organización del equipo. Por tal motivo el informe presentará a continuación una estructura bastante clara basada en dos aspectos: Diseño del Software y Uso de la Herramienta Git.

Con respecto al diseño del software, se indicó que fuera reacomodado, principalmente para una mejor implementación de interfaz que sirva como fachada más definida y clara en cuanto a su función, que provea de sí distintos métodos que sirvan como filtro para un usuario que desee recibir detalles del subsistema en general, pero enfocados particularmente en los procesos que se llevan a cabo para la facturación de los productos, a su vez se desea hacer uso en el diseño de dos patrones que facilitan la estructura y la cohesión del subsistema, estos patrones conocidos como el patrón estrategia y el patrón decorador son propuestos para mejorar notablemente la estructura del software.

El punto relacionado a la herramienta Git no solo va enfocado a la herramienta como tal, que como ya es bien sabido, es utilizada para realmacenamiento periódico de paquetes y manejo de versiones; sino que también se hace referencia a la organización grupal del equipo, en cómo ha sido estructurada y bajo qué principios se fundamenta. Esto último es de vital importancia dentro del contexto de Git, ya que se prevé la realización de una gestión distribuida y un fuerte apoyo al desarrollo no lineal del software.

4 Diseño del Software

Para esta entrega de la tarea se han realizado ciertas modificaciones sobre el diseño del software, entre otras encontramos una nueva clase llamada Facturador que sirve como nueva fachada y que ha sido utilizada como una nueva estrategia para aumentar la cohesión y disminuir el acoplamiento entre las clases que conforman el software. Este patrón fachada implementado sobre esta clase se relaciona con la mayoría de las clases del subsistema que hemos llamado "Lógica del Negocio INNOVA, suministra información concisa sobre este subsistema, principalmente en relación a las facturas que son realizadas a los productos.

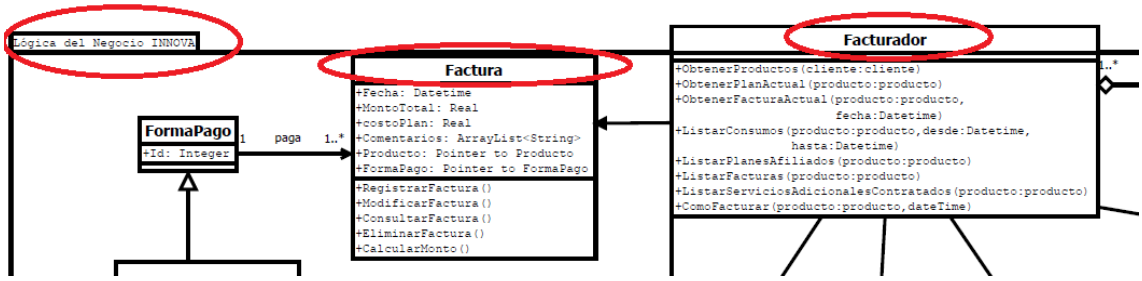

Ilustración 1 Clase Facturador

Los métodos que conforman esta clase de tipo fachada son los que realizan extracciones esenciales para proveer la interfaz, entre ellos tenemos el método que obtiene los productos de un cliente dado, el que lista los servicios adicionales contratados por un producto y el que muestra los consumos realizados por el producto dado en un intervalo de fechas, entre otros.

El siguiente patrón encontrado en el diseño del software se trata del patrón estrategia, el cual fue principalmente implementado por la necesidad de realizar un tipo de facturación para un producto dependiendo del plan al que éste esté afiliado, es decir, para los clientes con planes prepago, la factura se genera en el momento en el que se realice alguna recarga, además, cada vez que se realiza un consumo, el software debe cerciorarse de que el cliente posea un saldo suficiente y suspender el servicio en caso de no poseer saldo suficiente. Por otro lado, para los clientes con productos afiliados a planes postpago, los consumos realizados deben ser acumulados y facturados mensualmente.

Para poder aprovechar este recurso, se ha implementado en el diagrama la clase “ComoFacturar” que en realidad está contenida en la clase “Facturador” y hereda a dos clases en las cuales se encuentran los métodos que definirán la forma de emitir la factura para productos que estén afiliados a planes prepago y planes postpago.

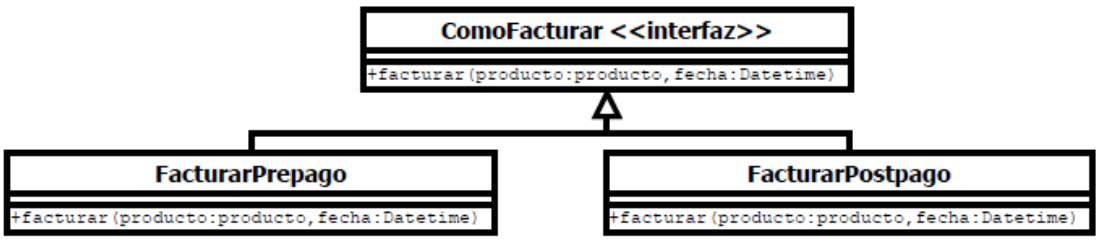

Ilustración 2 Clase ComoFacturar <<Interfaz>>

Un aspecto importante a resaltar sobre la implementación de los métodos de facturación es que, independientemente del plan que posea un producto que deba ser facturado, esta afiliación del producto con el plan se supondrá que ha sido hecha al principio del mes y que los consumos son calculados mensualmente, de tal forma que las fechas sean coincidentes y el cálculo de una factura sea mucho más fácil de realizar en un mes determinado.

El último patrón utilizado resaltante para realizar el diseño del software de esta entrega se trata del patrón decorador, el cual surge como alternativa para solventar el problema de poder añadir varios servicios adicionales a los productos contemplados en el subsistema. Para poder implementar el patrón decorador se ha hecho uso de una nueva clase a la cual se le ha nombrado “Facturable”, esta clase sirve para representar de una forma abstracta a todo elemento al cual se le pueda sacar una factura, a su vez, la clase Facturable hereda a una clase más concreta que ya es bien conocida: Producto; y también a la clase contenida llamada “Servicio Adicional Decorador”, ésta última representa a todos aquellos servicios adicionales que puedan ser añadidos a los productos en cuestión, para poder representar esta clase se ha hecho uso de unas clases que sirvan como ejemplo de servicios adicionales heredados, estos son: Buzón de Mensajes, Desvío de Llamadas y Conferencia Tripartita.

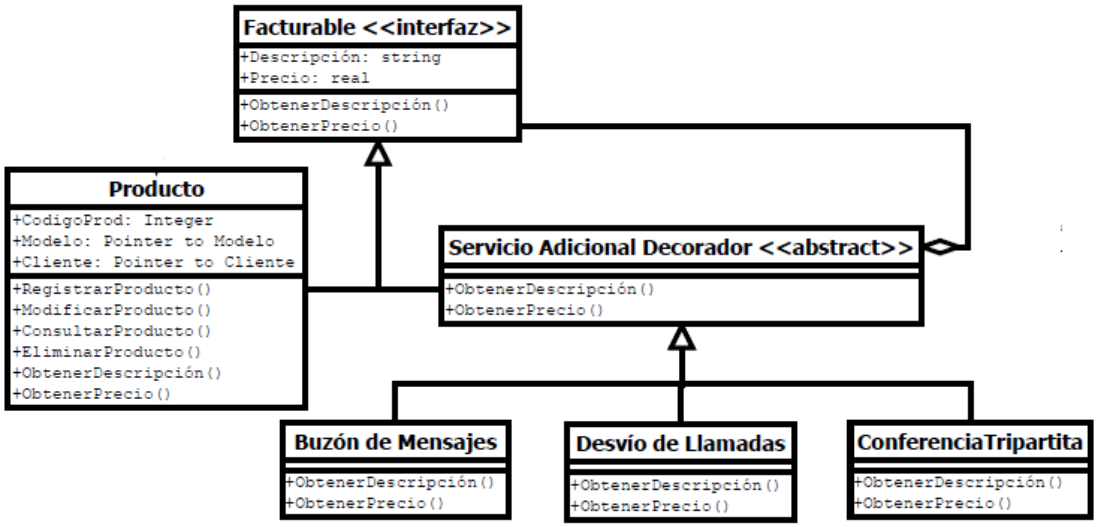

Ilustración 3 Clase Facturable <<Interfaz>>

5 Uso de la Herramienta Git

Para la implementación y organización del desarrollo del software se ha hecho un uso bastante concurrente de la herramienta Git, la cual ya se había comenzado a utilizar desde la entrega anterior, es decir, la tercera tarea como medio de manejo de versiones del software principalmente. Su uso fue propuesto desde un principio por Gabriela Limonta, quien ya había utilizado la herramienta desde hace un buen tiempo y todos los miembros restantes del equipo dieron su aprobación. Ya los miembros del equipo poseían cuentas en Github para aquel entonces por lo que solo se tuvo que invitar a los miembros a participar en el repositorio creado por la misma Gabriela Limonta.

Como distribución principal de las versiones se usó a la rama master como principal rama del manejador de versiones del repositorio, para esta entrega se organizó el equipo según las parejas originales de trabajo y se distribuyeron labores particulares para cada una de las parejas, principalmente se distribuyó la implementación de cada patrón mencionado en la sección anterior del documento y se mantuvo una pareja reservada para la implementación de las pruebas unitarias, de esta forma se garantiza una mayor fiabilidad en las pruebas por parte de personas que no formaron parte en la implementación de las clases y se mantiene una repartición de trabajo equitativo en el grupo.

A la hora de realizar las integraciones se concluyó en que cada pareja de trabajo crease una rama en la cual podrían trabajar independientemente de las demás y poder realizar versiones de las

mismas sin tener que escatimar en posibles esperas de los trabajos de otras parejas. Posteriormente, cuando cada patrón estuviera finalizado, se integrarían a la rama principal (master) y tanto John Delgado como Gabriela Limonta se encargarían de resolver posibles conflictos que estas integraciones pudieran traer consigo, aunque la posibilidad de que esto suceda es mínima dada la independencia tan notable de las clases que maneja cada rama.

El manejo de Git fue algo nuevo para algunos de los miembros del equipo por lo que se tuvo algunos problemas menores en la utilización de la herramienta y perjudicó una buena integración en un momento, sin embargo ha sido de vital importancia para todos el buen aprendizaje de la herramienta y cada miembro se mostró motivado y comprometido a aprender sobre ella, para esto también fue muy importante una de las clases de laboratorio en la cual se hizo bastante énfasis en explicar la utilización de la herramienta, lo que esclareció algunas dudas y también proporcionó nuevos conocimientos para todo el equipo.

La herramienta ha facilitado muchos aspectos para el desarrollo de este software, la capacidad de manejar versiones del software ha sido fundamental, sobre todo cuando se debe volver a un estado previo ya sea para consultar o para arreglar algo que ha salido mal en las versiones más recientes. Un aspecto que ha sido bastante importante es la utilización de las ramas, que como se dijo previamente, ha permitido al grupo poder dividir actividades y poder ejecutarlas en simultáneo para luego poder integrarlas sin tener consecuencias negativas, también ha facilitado bastante el ahorro del tiempo al no tener que hacer las integraciones a través de cada cambio que se haga por una persona en particular, línea por línea, y resulta asombroso la forma tan efectiva de integración que provee la herramienta.

En lo que a la organización se refiere, el equipo no posee un líder predefinido que coordine el trabajo, se ha desempeñado el papel de líder individualmente por parte de varios de los miembros del equipo en distintas oportunidades y para convocar a reuniones se está usando como vía principal el correo electrónico, por medio del cual cualquiera de los miembros puede solicitar reuniones presenciales y consultar a los demás su disponibilidad. Las decisiones importantes que rigen los planes de acción del equipo de trabajo están arraigadas a los acuerdos de los miembros, éstas se aprueban, se niegan o se dejan a espera según la voluntad del equipo en las reuniones presenciales.

6 Conclusión

Luego de realizar y probar el software para la presente entrega se ha concluido favorablemente en varios aspectos, entre ellas se destaca la buena organización y la comunicación entre los miembros del equipo, que desde un comienzo se fundamentó en reuniones para llevar a cabo planes de acción y dividir tareas de manera equitativa. Este aspecto fue fundamental para el desarrollo del software y sin el mismo no habría sido posible realizarlo de manera amena y menos dificultosa. Otro aspecto importante a resaltar es el uso de la herramienta Git que permitió establecer integraciones adecuadas a las implementaciones por parte de varios desarrolladores en forma simultánea, éstos basados en el uso del control de versiones que provee la herramienta.

En cuanto al diseño del software, se puede ver como los patrones a seguir para ciertos componentes del modelo han facilitado y organizado con consecuencias muy positivas el diseño, por su parte, el patrón fachada implementado para la clase facturador ha logrado resolver problemas de acoplamiento entre clases y ha servido de interfaz a un nivel mucho más accesible en el subsistema. El patrón decorador ha logrado resolver el problema de asociar servicios adicionales a los productos de los clientes de buena forma, afectando el precio de los productos de forma acorde, el único problema que presenta este patrón es el uso de clases definidas para hacer referencia a los decoradores, esto hace que los servicios adicionales se manejen de forma casi estática y por eso se ha propuesto en el diseño tres ejemplares para estos. Por último el patrón estrategia ha logrado resolver la diatriba de generar formas de facturación

para los clientes, dependiendo del plan al que sus productos estén afiliados, esto modulariza mejor las estrategias de facturación facilitando la mantenibilidad del software.

Para cada patrón se usaron las pruebas unitarias apropiadas para verificar la correctitud del software en general, dichas pruebas fueron satisfactorias y se pudo cerciorar el buen funcionamiento del subsistema, comprobando una vez más la importancia de la verificación para la detección de errores y defectos a través de las fallas que pudieron ser convenientemente observadas por las pruebas, con lo que se puede concluir en que el producto brindado es un software de calidad a nivel de funcionalidad.

7 Bibliografía

- ABAD MOTA, Soraya. 2010 Lineamientos sobre cómo escribir informes técnicos. Caracas, Universidad Simón Bolívar.
- SOMMERVILLE, Ian. 2007. Software Engineering. 8ª Edición.
- ZHANG, Kevin. 1997. Design Patterns, Elements of Reusable Object-Oriented Software.
- LARMAN, Craig. UML y Patrones. 2ª Edición.