UNIVERSIDAD SIMÓN BOLÍVAR SEDE SARTENEJAS
REPÚBLICA BOLIVARIANA DE VENEZUELA
CI-3115 LABORATORIO DE INGENIERÍA DE SOFTWARE
MODALIDAD SEMI-INTENSIVA: SEPTIEMBRE-OCTUBRE 2013

INFORME: Tarea 3

Profesor:
Carina Ferreira

Elaborado por:

Andrea Salcedo	10-10666
Jhon Delgado	10-10196
Marcos Campos	10-10108
Fernando D'Agostino	10-10812
Gabriela Limonta	10-10385
José Montenegro	10-10469
Luis Fernández	10-10239
Oswaldo Jiménez	10-10368

Caracas, Sartenejas. 16 de Septiembre de 2013

• Índice

• Introducción

El presente documento tiene como objetivo presentar el diseño y la funcionalidad del software orientado por objetos desarrollado por el equipo fundador de INNOVA. Dicho software consta fundamentalmente de una base de datos objeto-relacional y un diagrama de clases como soporte de diseño e implementación. En adición, el equipo de desarrollo acogió la propuesta original de modularización sugerida por el cuerpo docente.

El siguiente informe se divide en secciones o apartados. En estos se expondrá de forma clara y precisa lo referente al proceso de desarrollo de cada fase de elaboración y codificación del software. En particular, se hará mención de tres aspectos claves: conformación de la base de datos, diagrama de clases e implementación.

Con respecto a la base de datos, se hará exhibición de las 3 fases básicas inherentes al bosquejo de una base de datos: El modelo conceptual, lógico y físico. Para el conceptual, se mostrará el modelo Entidad Interrelación de la base de datos, generado a partir de la rigurosa inspección de los elementos afines observados en los diversos procesos de negocio empresarial. Del mismo modo, se concretará un diccionario de datos que permita la comprensión de las entidades, atributos e interrelaciones existentes. Con respecto al modelo lógico, se pondrán de manifiesto los esquemas de relación resultantes de haber procedido a la traducción del ERE.

Consiguientemente, se presentará el artefacto 'diagrama de clases', incluyendo la descripción del proceso de elaboración correspondiente y una reflexión sobre la importancia que este instrumento representa para el diseño arquitectónico del software. En conclusión, se describirá la estructura de codificación empleada y el uso de las pruebas unitarias como método de verificación y correctitud de las cualidades funcionales del software.

• Modelo Entidad Interrelación

A continuación se muestran las dos versiones de modelos entidad interrelación ERE escogidas para el desarrollo del software. Ambos diseños fueron evaluados en conjunto, a fin de concebir un único modelo que permitiera integrar eficazmente las ventajas de diseño proporcionadas por cada esquema.

Este nuevo modelo tiene como propósito principal cubrir de una forma más acertada los requerimientos solicitados, en particular está acondicionado para escrutar el reporte eficiente de las facturas por cliente.

El ERE final representa no sólo la manera en la que el equipo de desarrollo consideró más apropiada la disposición de los datos concernientes a las redes de negocio INNOVA; además de ello, constituye una importante base arquitectónica del software, pues en torno a ella giran artefactos de diseño e implementación decisivos como el diagrama de clases y el patrón fachada.

* **VER ARCHIVO ADJUNTO EN EL CORREO ELECTRÓNICO**:

Primer Modelo ERE: PDF MODELO_ERE_A
Segundo Modelo ERE: PDF MODELO_ERE_B
Modelo ERE Final: PDF MODELO_ERE_FINAL

• Modificaciones al modelo

Como ya fue mencionado anteriormente, el modelo entidad interrelación final fue construido a partir de la consideración de dos modelos base. Se procederá a indicar las modificaciones realizadas a los modelos (éstas modificaciones tendrán como referencia al MODELO_ERE_A). Se acometieron los siguientes cambios:

1) Integración de la entidad 'Tipo_Servicio', cuyo único atributo asociado es "nombre". La presencia de esta entidad permite registrar de forma genérica diversos tipos de servicios en el sistema.

2) Integración de la entidad 'Modelo', cuyo único atributo asociado es "nombre". La presencia de esta entidad permite registrar de forma genérica diversos tipos de modelos en el sistema.

3) Agregación de la interrelación "deriva" entre las entidades 'Servicio' y 'Tipo_Servicio', de cardinalidades (1,1), (0,N) de forma respectiva.

4) Agregación de la interrelación "Esta_Vinculado" entre las entidades 'Producto' y 'Modelo, de cardinalidades (1,1), (0,N) de forma respectiva.

5) Se añadió el atributo "tipo_plan" a la entidad 'Adicional'. Fue necesaria esta modificación en vista de que los servicios de tipo adicional pueden atender a un conjunto restringido de planes, esto es, pueden existir planes que no sean cubiertos por este servicio adicional, haciendo que esta indicación sea esencial a la hora de ser ofertado un servicio de este tipo.

6) Se suprimió el atributo "fecha_fin" de la interrelación 'posee'. Fue eliminado como medida de simplificación al modelo, en vista de que el valor de este atributo no es de especial interés. Por otra parte, se evita la aparición de nulos sistemáticos en el sistema.

• Diccionario de Datos

Se presenta aquí el diccionario de datos correspondiente al modelo ERE propuesto. En él se detallan las características y significados de todos los atributos de cada entidad, así como las especializaciones y categorizaciones. En la siguiente sección, serán listadas las interrelaciones que relacionan dichas entidades.

* **VER ARCHIVO ADJUNTO EN EL CORREO ELECTRÓNICO**:

PDF MODELO_ERE

• Interrelaciones

Con el fin de clarificar las convenciones empleadas en la construcción del modelo entidad interrelación, se introducirá a continuación un catálogo de interrelaciones; recordando que éstas corresponden a estructuras lógicas utilizadas como mecanismo de interconexión entre las entidades que componen el ERE. Se explicará entonces el significado semántico de dichas interrelaciones, además de hacer mención de los atributos que las conforman.

* **VER ARCHIVO ADJUNTO EN EL CORREO ELECTRÓNICO**:
PDF INTERRELACIONES_ERE

• Modelo Lógico

En este apartado se presenta el modelo lógico o relacional, el cual es resultado de traducir el modelo entidad interrelación presentado anteriormente, utilizando las diversas técnicas de traducción del modelo relacional y realizando los refinamientos pertinentes. Las claves primarias se representan con un subrayado simple, mientras que las claves foráneas se indican de forma explícita al final de cada relación.

Modelo Relacional

Adicional(nombre_servicio, tarifa, cantidad_adicional, tipo_plan)
nombre_servicio referencia a nombre_servicio en Servicio.
Cliente(CI, nombre, direccion)
Comentario(ID, mes, año, valor)
ID, numero, mes, año hacen referencia a dichos atributos en Factura.
Consume(ID, nombre_servicio, fecha, cantidad)
ID referencia a ID en Producto.
nombre_servicio referencia a nombre_servicio en Servicio.
Contiene(nombre_paquete, nombre_servicio, cantidad, costo_adicional)
nombre_paquete referencia a nombre_paquete en Paquete.
nombre_servicio referencia a nombre_servicio en Servicio.
Efectivo(nro_pago, postiza_pago)
postiza_pago referencia a postiza_pago en Forma_Pago.
Es_Dueño(ID, CI)
ID referencia a ID en Producto.
CI referencia a CI en Cliente.
Esta_Afiliado(ID, nombre_plan, tipo_plan, fecha_inic, fecha_fin)
ID referencia a ID en Producto.
nombre_plan, tipo_plan referencia a nombre_plan, tipo_plan en Plan.
fecha_fin puede ser NULL.
Factura(ID, mes, año, monto_total)
ID referencia a ID en Producto.
Forma_Pago(postiza_pago)
Modelo(nombre_modelo)
Paga(ID, mes, año, postiza_pago)
ID, mes, año hacen referencia a ID, mes, año en Factura.
postiza_pago referencia a postiza_pago en Forma_Pago.
Paquete(nombre_paquete, descripcion)
Plan(nombre_plan, tipo_plan, descripcion)
Posee(ID, nombre_servicio, fecha_inic)
ID referencia a ID en Producto.
nombre_servicio referencia a nombre_servicio en Servicio.
Producto(ID, nombre_modelo)
nombre_modelo referencia a nombre_modelo en Modelo.

Servicio(nombre_servicio, descripcion, nombre_tipo_servicio)
    nombre_tipo_servicio referencia a nombre_tipo_servicio en Tipo_Servicio.
Tarjeta(numero, marca, banco, cod_seguridad, fecha_venc, tipo_tarjeta, postiza_pago, ci_titular)
    postiza_pago referencia a postiza_pago en Forma_Pago.
Telefono(CI, valor)
    postiza_cliente referencia a postiza_cliente en Cliente.
Tiene(nombre_plan,tipo_plan, nombre_paquete, fecha_inic, fecha_fin, costo)
    nombre_plan, tipo_plan referencia a nombre_plan, tipo_plan en Plan.
    nombre_servicio referencia a nombre_paquete en Paquete.
Tipo_Servicio(nombre_tipo_servicio)

- # Diagrama de Clases

En esta sección se presenta el artefacto "diagrama de clases", instrumento de diseño empleado por el equipo de desarrollo para conducir la fase de implementación.

La elaboración de esta herramienta tuvo su principal base en el uso de la técnica de interpretación y traducción del modelo entidad interrelación. A través de esta abstracción se consolidaron las estructuras orientas al manejo de objetos observables en el presente diagrama. En consecuencia, es observable una analogía entre las clases de este esquema y las entidades el modelo ERE.

No obstante, además de la mencionada abstracción, el diagrama de clases incluye clases suplementarias, pensadas para integrar de manera eficaz al conjunto de estructuras que conforman la red de recuperación de datos del sistema, cuyo fin consiste en aprovechar los beneficios de débil acomplamiento (acoplamiento) y mantenibilidad ofrecidos por el patrón fachada.

La utilidad del diagrama de clases para el software es inminente. En primer lugar, por su capacidad comunicativa: éste permite comprender y modelar el problema planteado, logrando fijar panoramas afines entre los miembros del equipo. En segundo lugar, porque es en función de éste que se articula gran parte del proceso de codificación.

* **VER ARCHIVO ADJUNTO EN EL CORREO ELECTRÓNICO** :
PDF DIAGRAMA_CLASES

• Diseño, Implementación y Pruebas unitarias

La estrategia empleada para dirigir el proceso de codificación y diseño asienta sus bases en el diagrama de clases y en la propuesta de modularización sugerida en el planteamiento del problema.

A nivel de diseño, como ya se ha mencionado, el software desarrollado por el equipo aprovecha las ventajas de débil acoplamiento que ofrece el patrón fachada. En concreto, las clases especificadas en el modelo de clases fueron integradas en 4 módulos principales: Afiliaciones, Cliente, Consumo y Producto, permitiendo consolidar la interfaz característica de este patrón.

Desde una perspectiva arquitectónica, podemos identificar en el prototipo desarrollado por el equipo una composición por capas, dados los siguientes niveles de abstracción:

Disposición física de los datos en el manejador -> tipos abstractos de datos -> gestión del negocio a través de una interfaz con el usuario

En cuanto a la codificación respecta, se implementaron las clases debidamente representadas en el diagrama de clases, con sus respectivos métodos y atributos. Se respetó la jerarquía de herencia y el manejo de referencias plasmados en el esquema. Finalmente, a un mayor nivel de abstracción, se configuraron las clases asociadas a la gestión de consumos, clientes, productos y afiliaciones.

Una vez avanzado el prototipo, fue necesario el uso de las pruebas unitarias facilitadas por el paquete Junit. Se construyeron "tests" en todas las clases desarrolladas, a fin de validar los métodos y el comportamiento del prototipo en general. Cada prueba respeta el principio de independencia, garantizando la consistencia de las mismas. Parar ello, fue preciso el uso de datos ajenos a los ya encontrados en el manejador; a fin de no alterar o comprometer la BD al acometer las pruebas.

• Conclusiones

El software confeccionado por el equipo de trabajo fue desarrollado rigurosamente bajo una serie de etapas de diseño, sujetas tanto a los requerimientos y expectativas perseguidas como a controles de calidad característicos de la ética profesional del equipo.

El seguimiento de las estrategias proporcionadas por los conocimientos en ingeniería de software jugó un papel protagónico en el proceso de estructuración de la aplicación. Éste, en conjunto con el método iterativo de planificación y elaboración permitieron optimizar la agilidad y los resultados concernientes a las diversas etapas de desarrollo.

La correcta implantación de la base de datos, entendiéndose por ésta al debido establecimiento de entidades, restricciones de integridad y disparadores (triggers), resultó de vital importancia pues actuó como soporte de inicio a las fases posteriores de codificación. Más aún, ha de destacarse que la correcta articulación del diagrama de clases figuró una pieza clave en el acabado del producto final, al ser éste el esqueleto sobre el cual se materializaron los diversos módulos.

La implementación o codificación atendió a las convenciones más apropiadas posibles, así como a las buenas prácticas de programación y a la manipulación segura de los datos alojados en el DBDM. Fueron aprovechadas las herramientas proporcionadas por java y Junit. En este sentido, es importante resaltar la notoria utilidad que tuvieron las pruebas unitarias en materia de comprobación y correctitud del software. Gracias a éstas, fallos significativos pudieron ser detectados y, en consecuencia, corregidos.

• Referencias bibliográficas

• SHAMKANT NAVATHE, Ramez Elmasri, 2007. Fundamentos de Sistemas de Bases de Datos. Madrid :PEARSON

• Referencia Virtual : https://sites.google.com/site/ingsoftware1teruel/home/tareas-del-taller. 2013. Patrón Fachada

• Referencia Virtual : https://sites.google.com/site/ingsoftware1teruel/home/tareas-del-taller. 2013. Diseño(1)

• Referencia Virtual : https://sites.google.com/site/ingsoftware1teruel/home/tareas-del-taller. 2013. Diseño(2)

• Referencia Virtual : www.postgresql.org/files/documentation/pdf. 2012. Uso del manejador POSTGRESQL 9.0

• Referencia Virtual : http://www.postgresqlya.com.ar/temarios. 2013. Uso del manejador POSTGRESQL 8.3

• ABAD MOTA, Soraya. 2010 Lineamientos sobre cómo escribir informes técnicos. Caracas, Universidad Simón Bolívar