Universidad Privada Boliviana
Computación para Ingeniería II
Programación II

Base de datos para un servicio de renta de automóviles

Rodrigo Solares A.
Alejandro Vila

Versión 2.0

Cochabamba, 18 de septiembre de 2018

Registro Histórico de Cambios y Revisiones

Fecha	Versión	Descripción	Autor Responsable	Aprobado
03/09/2018	-	Presentación del tema	R. Solares, A. Vila	04/09/2018
03/09/2018	-	Presentación de los Requerimientos	R. Solares, A. Vila	04/09/2018
11/09/2018	-	Presentación del Modelo de Datos	R. Solares, A. Vila	11/09/2018
12/09/2018	<0.1>	Inicio de la codificación	R. Solares	
13/09/2018	<0.2>	Adición de clases correspondientes a Agencia, Cliente	A. Vila	
13/09/2018	<0.3>	Adición de clases correspondientes a Automóviles, Talleres	R. Solares	
15/09/2018	<0.4>	Adición de clases correspondientes a Rentas	R. Solares	
15/09/2018	<0.5>	Adición del registro de Servicios al registrar una Renta	R. Solares	
15/09/2018	<0.6>	Adición de clases correspondientes a Administrativos	A. Vila	
15/09/2018	<0.7>	Adición del cálculo del monto de una Renta	R. Solares	
15/09/2018	<0.8>	Actualización de Cliente, Admin., Agencia.	A. Vila	
15/09/2018	<0.9>	Actualización del Menú Principal, Cliente, Admin., Agencia	A. Vila	
16/09/2018	<1.0>	Adición de clases correspondientes a Mecánicos	A. Vila	
17/09/2018	<2.0>	Comprobación y actualización de las consultas SQL.	R. Solares, A. Vila	

Tabla de Contenidos

Informe Final del Proyecto

1. Introducción

El presente documento tiene como fin registrar la realización del proyecto. El proyecto tuvo como fin la aplicación de los conocimientos y técnicas adquiridos en el curso de Programación II; para éste, se decidió realizar un sistema que sea capaz de procesar, almacenar y manejar datos pertinentes a un servicio de rentas de automóviles. Un servicio de rentas necesita del registro de sus bienes y servicios y una contabilización de sus precios. Se necesitó de un programa que interactúe con una base de datos creada, además de interactuar con el usuario final.

2. Análisis de Requerimientos

Para lograr crear un sistema que registre la actividad de un servicio de rentas primero debimos establecer las funciones y límites de nuestro servicio ficticio. Éste contará de una flota de automóviles de diverso tipo, agencias y talleres en diferentes ciudades, empleados administrativos y mecánicos, y servicios complementarios a la renta de un automóvil.

2.1 Requerimiento *«Registrar automóviles»*

- Debe registrarse a la base de datos los siguientes datos de un automóvil:
  - Código de identificación.
  - Placa vehicular.
  - Tipo de automóvil.
  - Capacidad del automóvil.
  - Modelo del automóvil (año)
  - Estado del automóvil (Rentado o no rentado)
  - Precio de renta del automóvil por día.
- Debe verificarse que el código de identificación y la placa vehicular sean únicas para cada automóvil, además de que el precio no sea nulo.
- Duplicación del código de identificación y de la placa vehicular.

2.2 Requerimiento *«Registrar agencias»*

- Registrar a la base de datos:
  - Código de identificación.
  - Dirección.
  - Ciudad.
- Unicidad del código de identificación.
- Duplicidad del código de identificación.

2.3 Requerimiento *«Registrar talleres»*

- Los talleres tienen una relación uno a uno con las agencias; es decir, a cada agencia le corresponde un taller. Debe registrarse:
  - Código de identificación.
  - Dirección.
  - Capacidad.
- Unicidad del código de identificación.
- Duplicidad del código de identificación.

2.4 Requerimiento *«Registrar administrativo»*

- Registrar la información de un empleado de administración el cual tendrá una relación con las agencias (donde trabaja) de uno a muchos; muchos administrativos trabajan en una agencia. Se toma en cuenta:
  - Código de identificación.
  - Código de la agencia en donde trabaja.
  - Carnet de identificación.
  - Número de contacto.
- Unicidad del código de identificación. Existencia del código de la agencia donde trabaja el administrativo. Debe considerarse que en el caso de que se elimine de la base de datos a una agencia, todos los administrativos correspondientes también deberán ser eliminados; eliminación en cascada.
- Duplicidad del código de identificación. Inexistencia del código de agencia donde trabaja el administrativo.

2.5 Requerimiento *«Registrar mecánicos»*

- Registrar a la base de datos los siguientes datos:
  - Código de identificación.
  - Código del taller donde trabaja el mecánico.
  - Carnet de identificación.
  - Número de contacto.
- Unicidad del código de identificación. Debe verificarse la existencia del registro del taller en el que trabaja el mecánico, caso contrario no se procede con el registro del mismo. Eliminación en cascada de los mecánicos pertenecientes a un taller eliminado.
- Duplicidad de código de identificación. Inexistencia del código de taller relacionado al mecánico.

2.6 Requerimiento *«Registrar clientes»*

- Registro de los datos relacionados a un cliente:
  - Código de identificación.
  - Número de contacto.
  - Carnet de identificación.
  - Dirección domiciliaria.
- Unicidad del código de identificación y del carnet.
- Duplicidad de código.

2.7 Requerimiento *«Registrar servicios»*

- Registrar los servicios complementarios que el servicio de rentas ofrezca. Debe registrarse:
  - Código de identificación.
  - Precio del servicio.
  - Tipo del servicio.
- Unicidad del código de identificación y la no-nulidad del precio.
- Duplicidad del código y la nulidad del precio.

2.8 Requerimiento *«Registrar rentas»*

- Registrar los datos correspondientes a una renta. Debe registrarse:
  - Código de identificación.
  - Código del cliente que adquiere el servicio.
  - Código de la agencia que realiza la renta.
  - Código del automóvil rentado.
  - Fecha de inicio de la renta.
  - Fecha de fin de la renta.
- Unicidad del código de identificación. No-nulidad de las fechas. Correspondencia de los códigos ingresados con los de la base de datos.
- Duplicidad del código, inexistencia de las fechas o de los códigos ingresados.

2.9 Requerimiento *«Deducir monto final»*

- Debe calcularse el monto final de la renta de un automóvil en base al precio de renta del automóvil por día, la duración de la renta y los servicios complementarios otorgados. Para acceder a los datos necesarios se considera al código de la renta y los códigos de los servicios adheridos a la renta.
- Validez de las fechas de la renta. Existencia del código de renta del cual se desea deducir el monto.
- Fechas inválidas. Inexistencia de una renta consultada.

2.10 Requerimiento *«Modificar»*

- La capacidad de modificar cualquiera de los datos, a excepción de códigos, de cualquier entidad.
- Debe comprobarse la existencia del elemento a modificar.
- Inexistencia del elemento.

2.11 Requerimiento *«Eliminar»*

- La capacidad de eliminar cualquier elemento registrado.
- Al eliminar un elemento que esté relacionado a otros, deben eliminarse estos elementos relacionados también.
- Inexistencia del elemento.

2.12 Requerimiento *«Listar»*

- La capacidad de listar entidades a partir de un código de identificación o a partir de una entidad.
- Deben existir elementos registrados para ser listados.
- Inexistencia de elementos.

2.13 Requerimiento *«Añadir servicios a una renta»*

- Al momento de registrarse una renta se deben añadir, si es el caso, los servicios que complementarán a la renta.
- Deben existir servicios que añadir.
- Inexistencia servicios.

3. Objetivo General

Obtener un sistema de información para un servicio de renta de automóviles que sea eficiente y funcional para los usuarios finales. A la vez se busca obtener mayor conocimiento y experiencia para la materia de programación. Se busca implementar el conocimiento adquirido a lo largo de la materia para que el sistema logre recopilar, almacenar y procesar datos que el usuario final ingrese, cuidando que la base de datos sea de fácil uso para los usuarios.

4. Objetivos Específicos

- Identifica las entidades que tendría una concesionaria.
- Relacionar de forma adecuada todas las entidades que tiene la concesionaria.
- Registrar los atributos que tenga cada entidad.
- Implementar una base de datos para la concesionaria.
- Crear métodos para poder ingresar, actualizar, eliminar, listar y procesar datos.
- Crear un menú para cada entidad que tengamos.
- Realizar un menú principal que te envié al menú de la entidad que desees.
- Almacenar datos a la base de datos.

5. Límites y Alcances

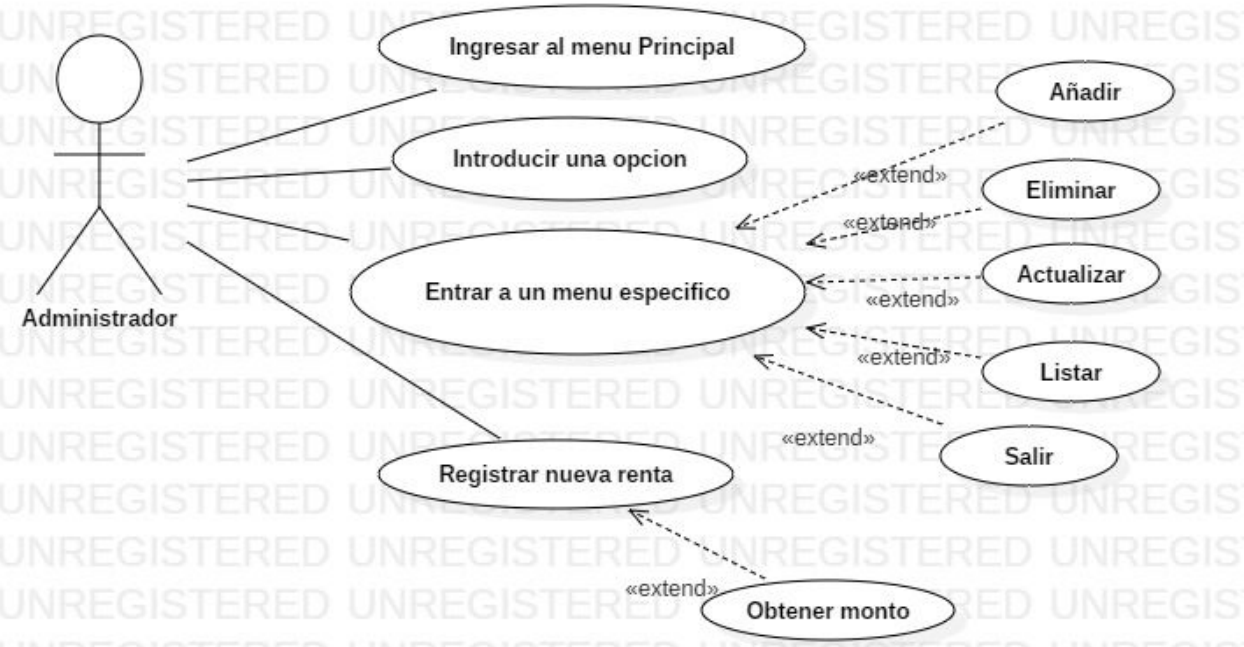
El proyecto pretende realizar todos sus objetivos propuestos; sin embargo, vale notar que como limitación tendremos a nuestro conocimiento adquirido en clases y la expansión más inmediata de este. Esto quiere decir que por motivos de tiempo y de técnica, no se incursionará en la utilización de técnicas que estén lejos de nuestro actual conocimiento; como por ejemplo, el uso de interfaces que permitan la centralización y/o generalización de las consultas SQL, visto que muchos de los métodos del programa varían tan sólo en estas: las consultas SQL.

6. Marco Teórico

Más allá del conocimiento adquirido en el curso se tuvieron que implementar nuevas técnicas:

- Uso de la restricción *Cascade*.- permite que al momento de eliminarse un elemento, se eliminen los elementos relaciones, mediante llaves foráneas, a éste. Su implementación se dio lugar al momento de crear la base de datos.
- Uso de la librería *java.util.Calendar*.- fue necesaria su implementación para hacer uso de los tipos de datos *Date* de la base de datos.
- Uso de la librería java.sql.Date.- se usó el método *getTimeInMillis()* lograr convertir el tipo de dato *Calendar* (usado por java para manejar fechas) al tipo de dato *Date* (usado por sql).

7. Diagramas de Casos de Uso



Administrador

7.1 Caso de Uso «<Ingresar al Menú Principal>»

7.2 Caso de Uso «<Elegir una opción>»

7.3 Caso de Uso «<Entrar a un menú específico>»

7.3.1 Caso de Uso «<Ingresar>»

7.3.2 Caso de Uso «<Eliminar>»

7.3.3 Caso de Uso «<Registrar>»

7.3.4 Caso de Uso «<Actualizar>»

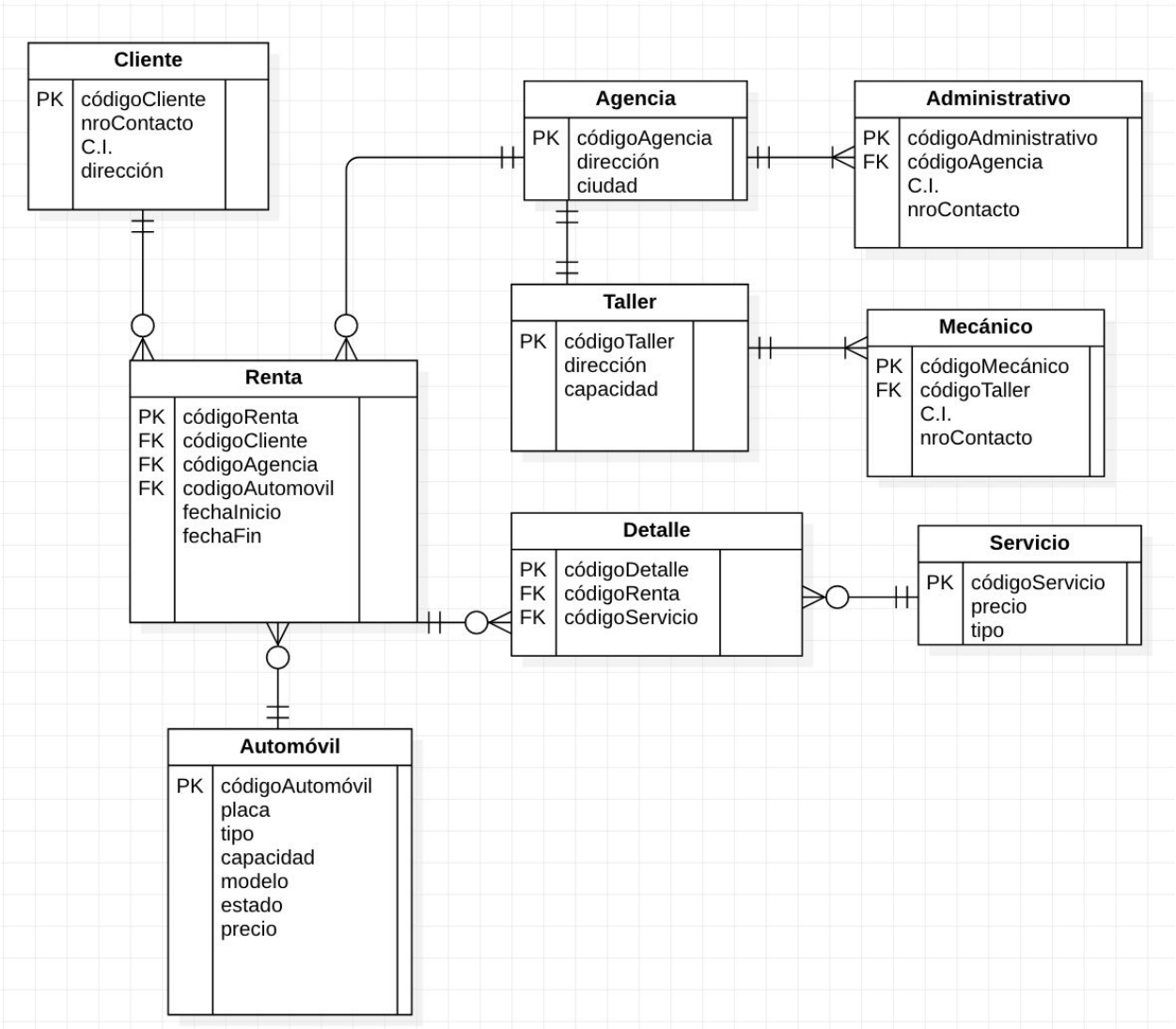
7.3.5 Caso de Uso «<Listar>»

7.3.6 Caso de Uso «<Salir>»

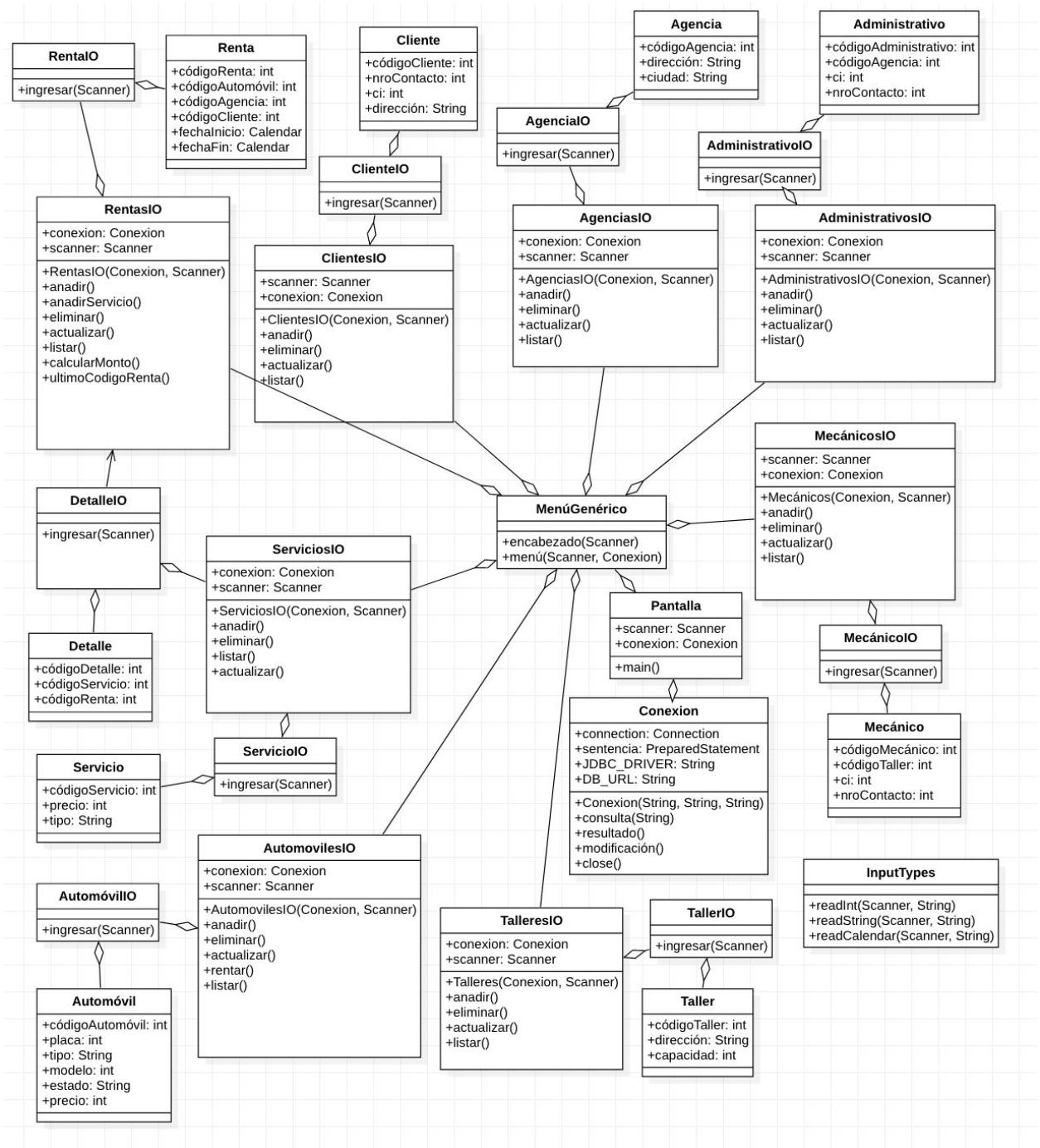
7.4 Caso de Uso «<Registrar una nueva Renta>»

7.4.1 Caso de Uso «<Obtener el monto de la Renta>»

8. Modelo de Datos



9. Diagrama de Clases



10. Pruebas y Validación

Las pruebas realizadas para validar el funcionamiento del programa fueron básicas: se usó la prueba y error para identificar problemas. Una vez concretada la codificación del programa, y la comprobación de que no existiesen problemas con la sintaxis de la misma, se procedió con las pruebas de conexión con la base de datos. Se comprobaron cada una de las funcionalidades, revisando si tuviesen el efecto esperado en la base de datos. Con la herramienta dbForge, se recuperaron los datos de la base de datos luego de ser modificada para cerciorarnos de que el efecto haya sido positivo. Funcionalidad por funcionalidad se fue comprobando el correcto funcionamiento del programa.

11. Conclusiones

Los objetivos planteados a un inicio del documento fueron logrados con éxito. Se notó que a lo largo del proceso muchos métodos y consultas se repitieron, lo cual suscitó ideas de generalizar el sistema, pero que no fueron concretadas. Sin contar con pequeños detalles, el conocimiento adquiridos en la materia fueron suficientes para desarrollar el programa. El mayor problema que se encontró al desarrollar el programa fue de implementar la funcionalidad de añadir servicios al momento de registrar una renta. Esto se debe a que para el registro de un servicio a una renta se debe conocer el código de la renta a la cual se suscribe el servicio; pero, no hallamos una forma directa de adquirir el código de renta que apenas fue creado, entonces se decidió aprovechar la naturaleza de incremento de las llaves principales para ordenar los códigos de renta y seleccionar el mayor. Siendo este el mayor ‘problema’, concluimos el proyecto satisfactoriamente.

12. Recomendaciones

Con miras a un futuro desarrollo del sistema creado, se recomienda trabajar en formas de generalizar los método creado, tal vez con la ayuda de interfaces. Muchas de las consultas sql utilizadas en el proyecto difieren en tan sólo un par de palabras. Nos parece que existe el potencial para centralizar los métodos y consultas utilizados y volver al sistema más eficiente. Además, se podría idear una manera alternativa de registrar los servicios correspondientes a una renta, puesto que actualmente existe la necesidad de que dentro de la Tabla de Servicios exista un elemento de tipo «Sin servicio» y de precio «0». Creemos que esto se debe a que el código SQL utilizado para el cálculo del monto toma en cuenta un valor nulo al no haberse añadido ningún servicio, que convierte la suma del monto a un valor nulo.

13. Bibliografía

1. https://docs.oracle.com/javase/7/docs/api/java/util/Calendar.html (Manejo de tipo de dato *Calendar*)
2. https://stackoverflow.com/questions/9112770/how-to-convert-calendar-to-java-sql-date-in-java (Conversión de *Calendar* a *Date*)

14. Anexos

- Manual de Usuario.- puede encontrarse en la rama de «Documentación» del repositorio del proyecto.