**INFORME DE PROYECTO DE PROGRAMACIÓN ORIENTADA A OBJETOS**

**UNIVERSIDAD DE CARTAGENA FACULTAD DE INGENIERÍAS**

**PROGRAMA DE INGENIERÍA DE SOFTWARE**

**SEMESTRE II**

**EJERCICIO 25**

**El Servicio Militar**

**ESTUDIANTES**

JOSE DAVID GOMEZ CHAMORRO

SIXTA VIVANCO PALACIO

**INSTRUCTOR**

JHON CARLOS ARRIETA ARRIETA

**FECHA DE ENTREGA**

22/01/2023

Tabla de contenido

[INTRODUCCION|UNO 2](#_Toc22225970)

[ENUNCIADO DEL RETO |DOS 2](#_Toc22225970)

[ANALISIS DE REQUERIMIENTO Y REQUISITOS|TRES 2](#_Toc22225972)

[MODELO CONCEPTUAL|CUATRO 2](#_Toc22225972)

[DIAGRAMA DE CLASES|CINCO 2](#_Toc22225972)

[DESCRIPCION DE LAS CLASES|SEIS 2](#_Toc22225972)

[bibliografía 2](#_Toc22225973)

EL SERVICIO MILITAR

El Ministerio de Defensa desea diseñar una Base de Datos para llevar un cierto control de los soldados que realizan el servicio militar. Los datos significativos a tener en cuenta son: Un soldado se define por su código de soldado (único), su nombre y apellidos, y su graduación. Existen varios cuarteles, cada uno se define por su código de cuartel, nombre y ubicación. Hay que tener en cuenta que existen diferentes Cuerpos del Ejército (Infantería, Artillería, Armada, ....), y cada uno se define por un código de Cuerpo y denominación. Los soldados están agrupados en compañías, siendo significativa para cada una de éstas, el número de compañía y la actividad principal que realiza. Se desea controlar los servicios que realizan los soldados (guardias, imaginarias, cuarteleros, ...), y se definen por el código de servicio y descripción.

Consideraciones de diseño:

● Un soldado pertenece a un único cuerpo y a una única compañía, durante todo el servicio militar. A una compañía pueden pertenecer soldados de diferentes cuerpos, no habiendo relación directa entre compañías y cuerpos.

● Los soldados de una misma compañía pueden estar destinados en diferentes cuarteles, es decir, una compañía puede estar ubicada en varios cuarteles, y en un cuartel puede haber varias compañías. Eso si, un soldado sólo esta en un cuartel.

● Un soldado realiza varios servicios a lo largo de la mili. Un mismo servicio puede ser realizado por más de un soldado (con independencia de la compañía), siendo significativa la fecha de realización.

**INTRODUCCIÓN**

En el desarrollo del curso se estudia el paradigma de la programación orientado a objetos donde se abarcará temas como lo son abstracción, encapsulamiento, polimorfismo, herencia, además se adquieren habilidades de identificación de clases, atributos, métodos y diagramación de estas para planteamiento y resolución de casos de estudio propio a la asignatura de Programación Orientada a Objetos. Puntualmente también se abarcan estos temas en el semestre junto a la aplicabilidad del proyecto:

1. Diagramación de clases.
2. Relación entre clases.
3. Lenguajes Java.
4. Métodos setter & getter.
5. Métodos constructores.
6. Sobrecarga de métodos.
7. Herencia y Polimorfismo.
8. CRUD.
9. Escuchador de eventos.
10. Conexión de base de datos a Java.
11. Creación de interfaz por Java Swing.

En el desarrollo del proyecto se plantea una problemática relacionada con la necesidad de implementar una base de datos para el ministerio de defensa y así tener un excelente control de la información de prestación de servicio militar, distribución del personal y roles de la camaradería en los batallones. Por este caso descrito se propone un software de escritorio creado a partir del lenguaje de programación Java, junto a los conceptos de POO, CRUD, SQL e interfaces para poder instituir un sistema que permita realizar consultas a la base de datos a través de botones y campos de textos (JButton & JtextField) en la interfaz gráfica de usuarios (GUI).

En la documentación junto a la metodología de desarrollo se plasman las necesidades de la elaboración del software, análisis de requisitos y requerimientos de diseño del mismo, casos de uso de la propuesta para una óptima interacción del usuario con el software, identificación de las partes para implementar el paradigma de la programación orientada a objetos, es decir identificar las clases, atributos de cada clase, los métodos que necesitaran ser establecido para cada clase y los respectivos diagramas que se abarcan en el avance del proyecto.

**OBJETIVO GENERAL**

Dar solución encaminada a la aplicabilidad del paradigma de la programación orientada a objeto en los sistemas de información, a través de la identificación de clases y atributos manifestados en el contexto del caso e implementación de los métodos, además del manejo de los conceptos bases del paradigma como lo son herencia, polimorfismo y encapsulamiento.

**OBJETIVOS ESPECIFICOS**

* Identificar las clases, atributos y tipos de datos plasmados en la descripcion del caso o proyecto.
* Plasmar en los diagramas la estructura del programa e implementar el modelo conceptual.
* Saber los métodos en función de las clases descritas, junto su aplicabilidad en el software
* Conectar el programa con base de dato SQL e implementar CRUD
* Realizar consultas, registrar, actualizar y eliminar información de datos almacenados en BD.
* Implementar GUI con Java swing para una interactividad amigable entre usuario y software.
* Dar resultados del CRUD en la interfaz al usuario final.

**Análisis De Requerimiento y Requisitos**

|  |  |  |  |
| --- | --- | --- | --- |
| Número de requisito | 1 | | |
| Nombre de requisito | Registro | | |
| Tipo | Requisito | Restricción | |
| Fuente del requisito | Se podrá consignar información descrita en el caso y registrar en la respectiva base de datos | | |
| Prioridad del requisito | Alta/Esencial | Media/Deseado | Baja/ Opcional |

|  |  |  |  |
| --- | --- | --- | --- |
| Número de requisito | 2 | | |
| Nombre de requisito | UpLoad Registro (Crear) | | |
| Tipo | Requisito | Restricción | |
| Fuente del requisito | Se carga la data en la base de datos a través de Java | | |
| Prioridad del requisito | Alta/Esencial | Media/Deseado | Baja/ Opcional |

|  |  |  |  |
| --- | --- | --- | --- |
| Número de requisito | 3 | | |
| Nombre de requisito | Solicitud De Información (Leer) | | |
| Tipo | Requisito | Restricción | |
| Fuente del requisito | El usuario de la BD podrá realizar consultas de la información de la BD por medio de la interfaz creada en Java Swing | | |
| Prioridad del requisito | Alta/Esencial | Media/Deseado | Baja/ Opcional |

|  |  |  |  |
| --- | --- | --- | --- |
| Número de requisito | 4 | | |
| Nombre de requisito | Eliminación de información | | |
| Tipo | Requisito | Restricción | |
| Fuente del requisito | Se podrá eliminar información de la BD, además de información visual en la interfaz de Java Swing | | |
| Prioridad del requisito | Alta/Esencial | Media/Deseado | Baja/ Opcional |

|  |  |  |  |
| --- | --- | --- | --- |
| Número de requisito | 5 | | |
| Nombre de requisito | Botones CRUD | | |
| Tipo | Requisito | Restricción | |
| Fuente del requisito | La interfaz deberá de contar con campos, botones y addEventListener para poder ejecutar el llamado de los métodos e implementar el CRUD | | |
| Prioridad del requisito | Alta/Esencial | Media/Deseado | Baja/ Opcional |

|  |  |  |  |
| --- | --- | --- | --- |
| Número de requisito | 6 | | |
| Nombre de requisito | Campos text | | |
| Tipo | Requisito | Restricción | |
| Fuente del requisito | La interfaz tendrá campos de textos para poder depositar la información y al iniciar el CRUD enviar lo teclado a la base de datos | | |
| Prioridad del requisito | Alta/Esencial | Media/Deseado | Baja/ Opcional |

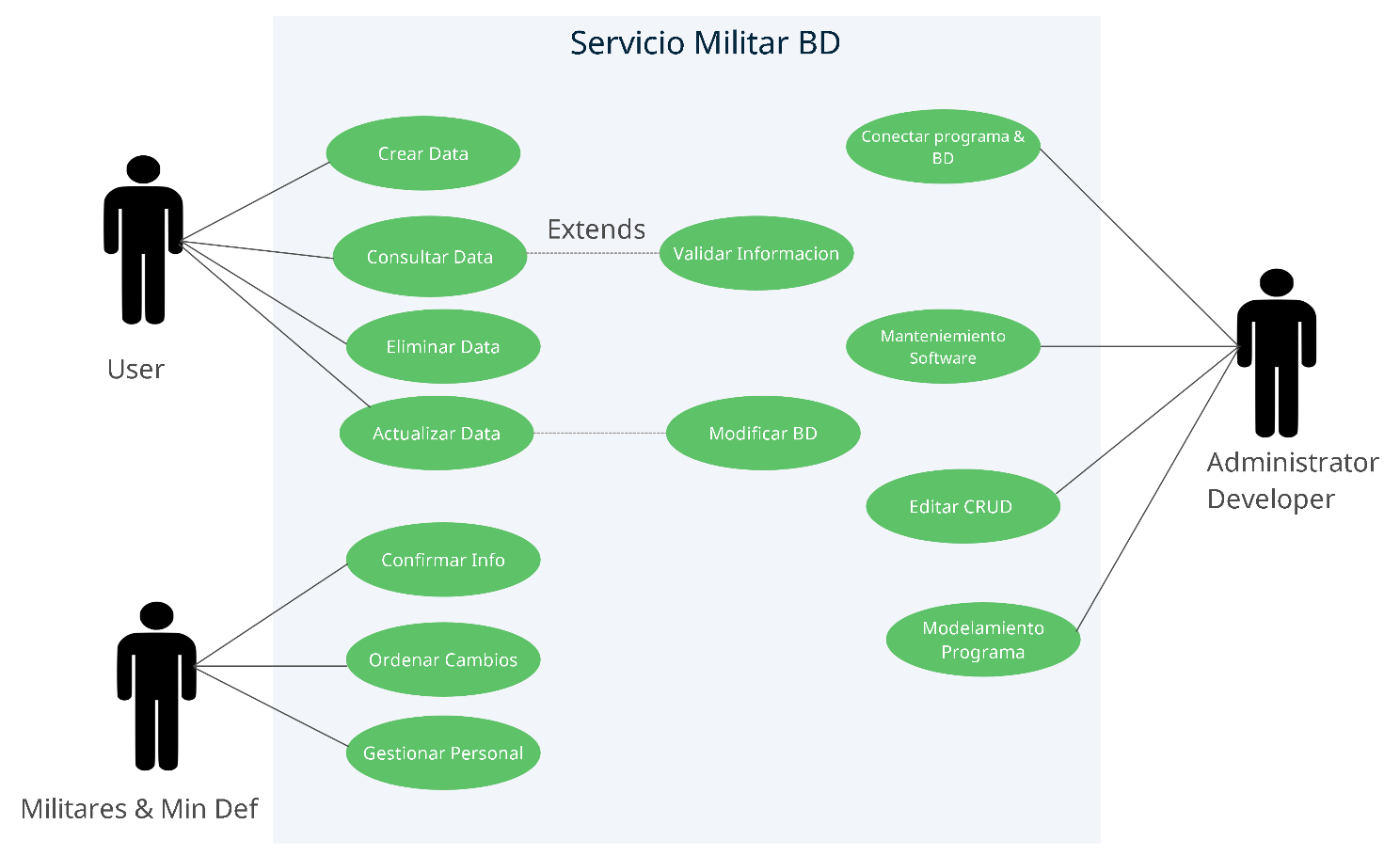
|  |  |  |  |
| --- | --- | --- | --- |
| Número de requisito | 7 | | |
| Nombre de requisito | Proyección Información Registrada | | |
| Tipo | Requisito | Restricción | |
| Fuente del requisito | Información que el usuario registre o consigne a la BD será proyectada inmediatamente en la interfaz validando la consignación de la información | | |
| Prioridad del requisito | Alta/Esencial | Media/Deseado | Baja/ Opcional |

|  |  |  |  |
| --- | --- | --- | --- |
| Número de requisito | 8 | | |
| Nombre de requisito | Validación De Datos Por Campo | | |
| Tipo | Requisito | Restricción | |
| Fuente del requisito | Al momento de implementar le CRUD se debe de validar que el tipo de dato sea consistente con la información solicitada en el campo. | | |
| Prioridad del requisito | Alta/Esencial | Media/Deseado | Baja/ Opcional |

**CASOS DE USO**

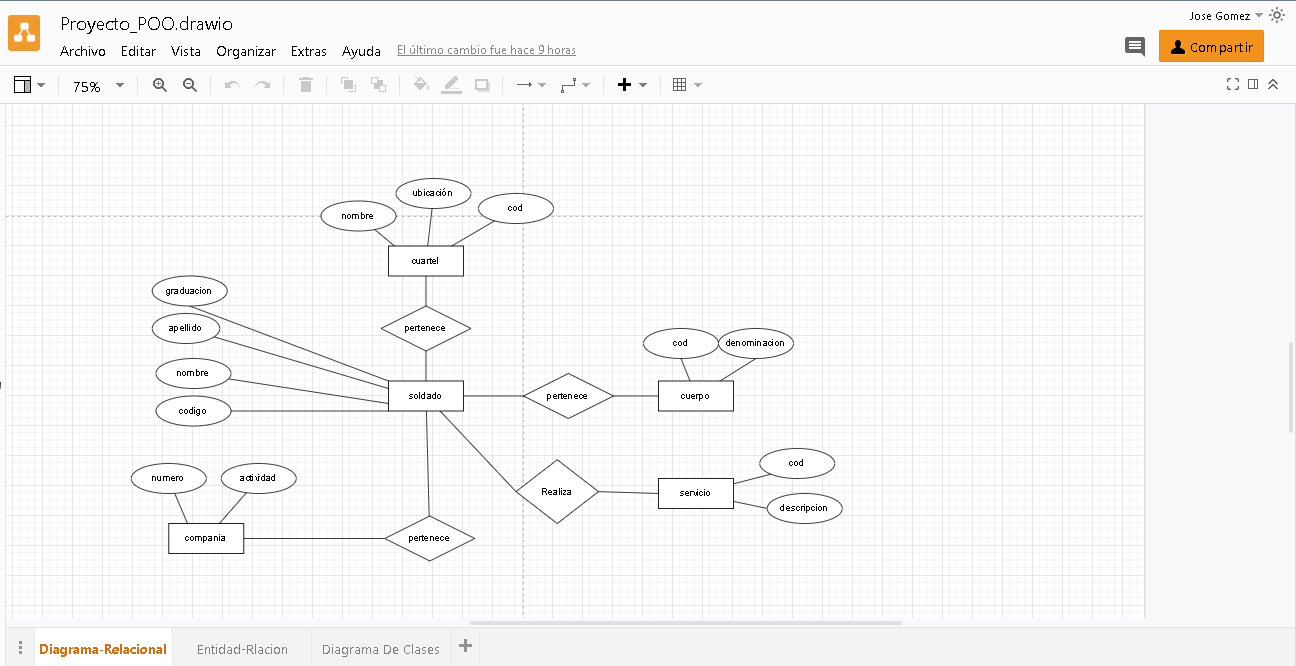
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Actor** | Usuario | | | **#1** | |
| Descripcion | Rol administrativa que será el encargado de manejar el software | | | | |
| Características | Interactuara constantemente con personal militar para alimentar o modificar BD | | | | |
| Relaciones | Informará y notificara constantemente el ingeniero de software las necesidades del sistema | | | | |
| Referencias | Realizara consultas, cambio de información, eliminación o lecturas en la base de datos | | | | |
| Autor | Jose Gomez & Sixta Vivanco | **Fecha** | 26/11/2022 | **Versión** | #1.0 |

**DIAGRAMA DE CASOS DE USO**

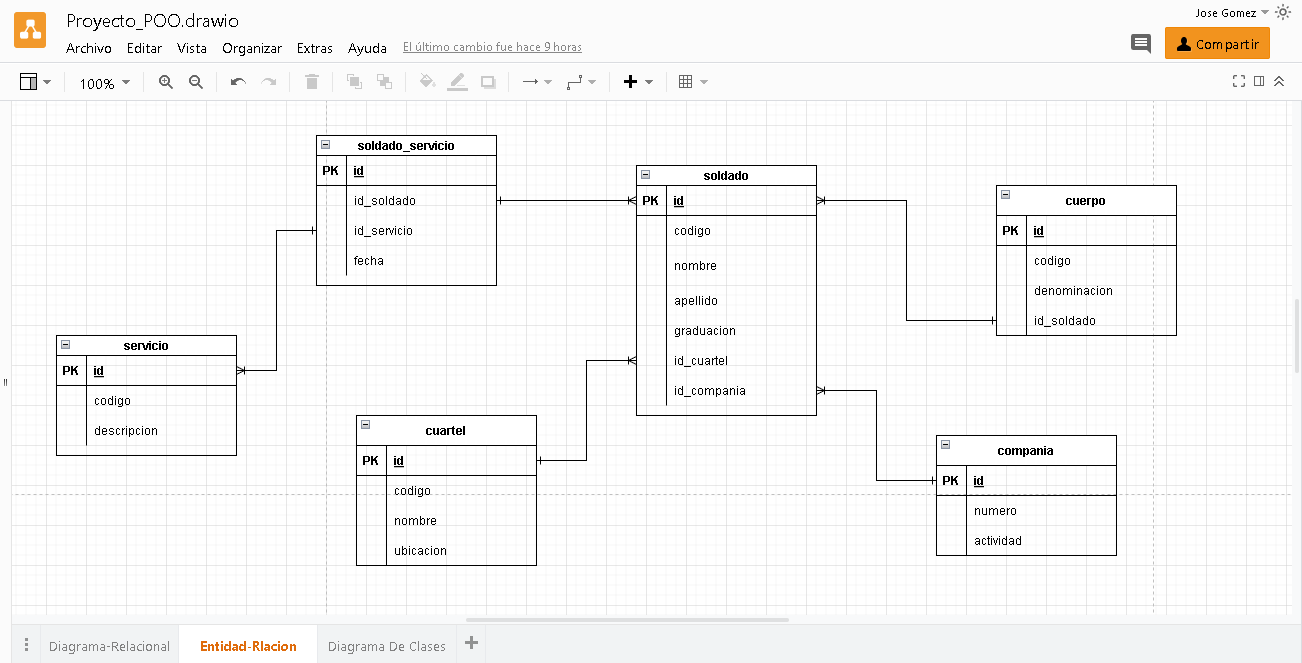


**Construcción De Diagramas (Drawio)**

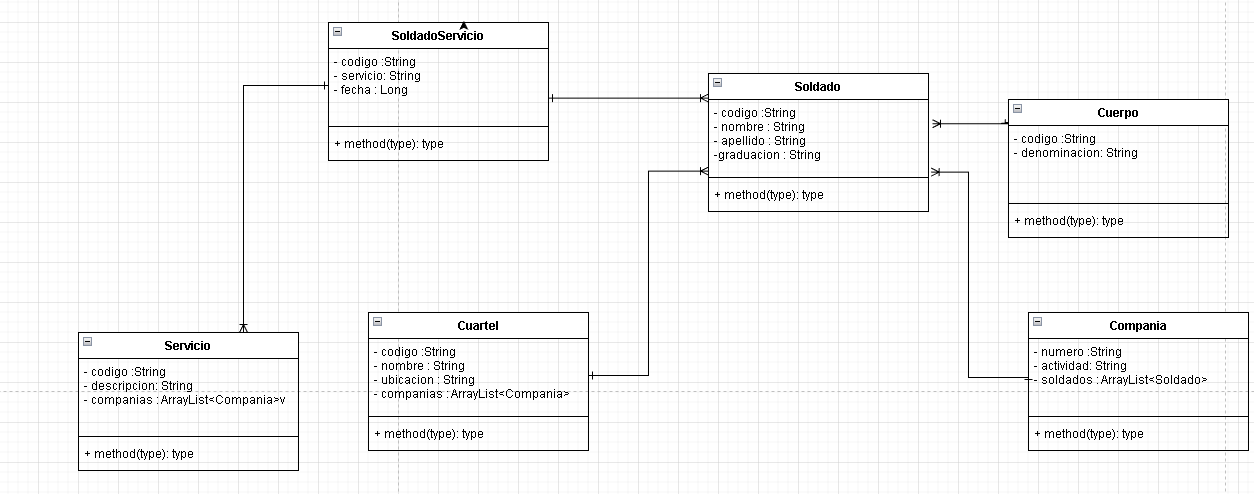
* **Diagrama - Relacional**

****

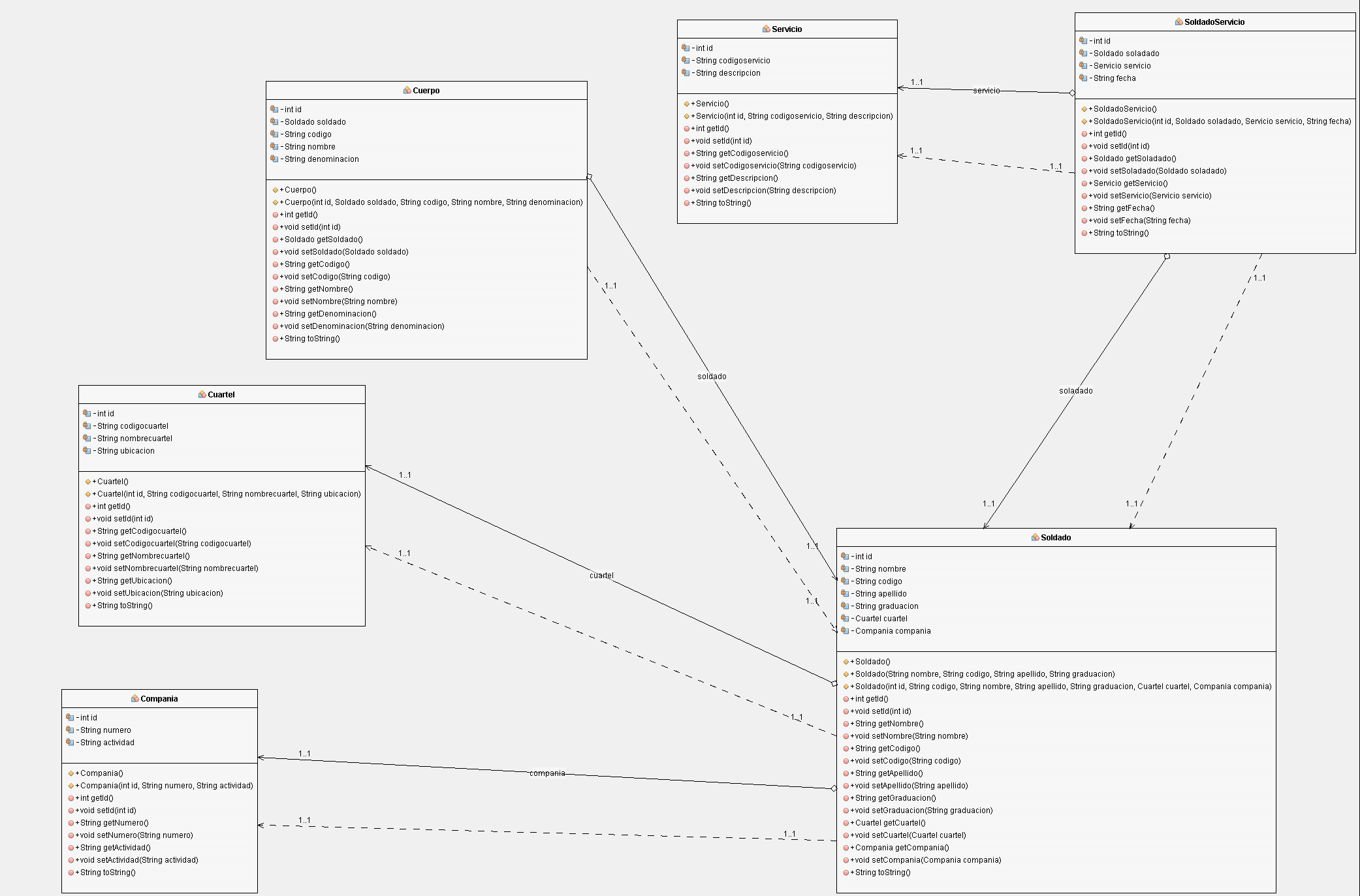
* **Diagrama Entidad – Relación**

****

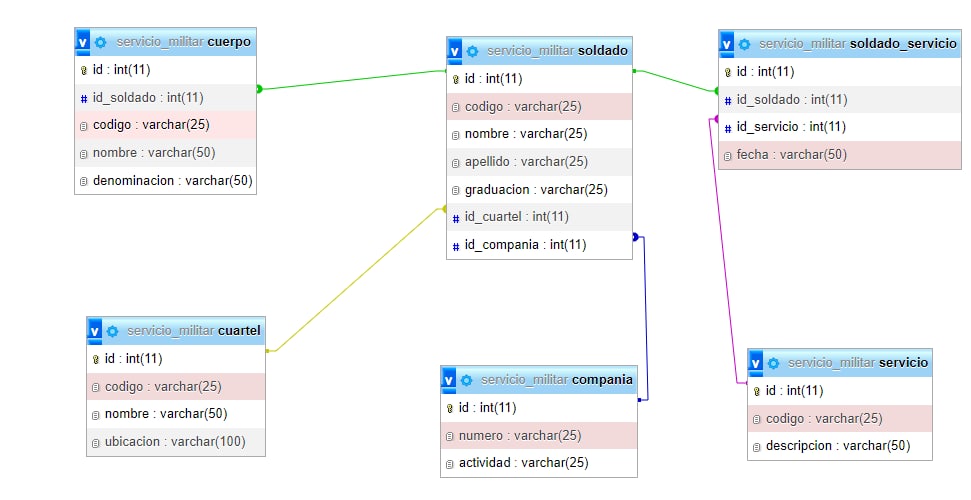
* **Diagrama De Clases (Sin Métodos)**

****

**Diagrama De Clases From Netbeans (Con Métodos)**



**Diagrama Entidad – Relación From Netbeans**

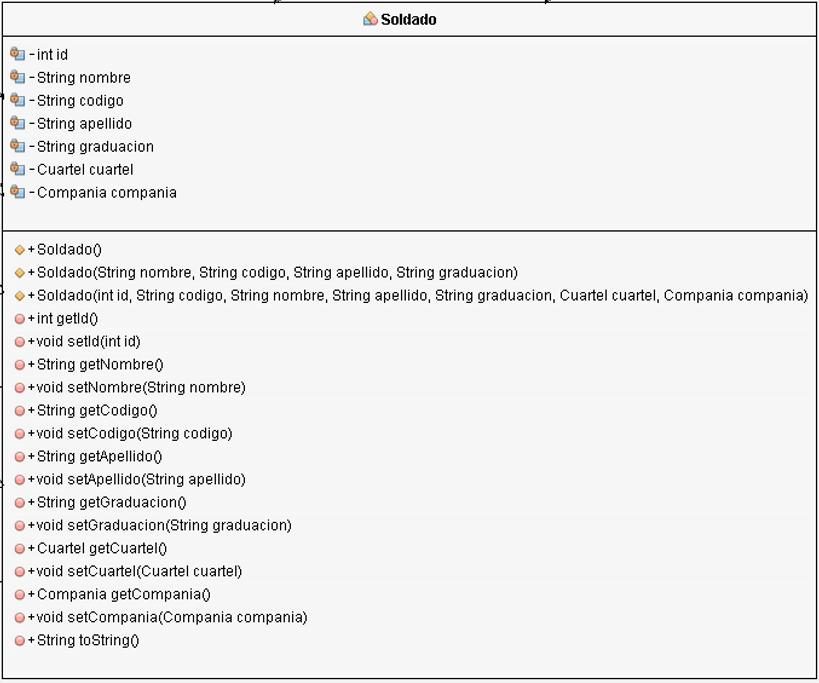
****

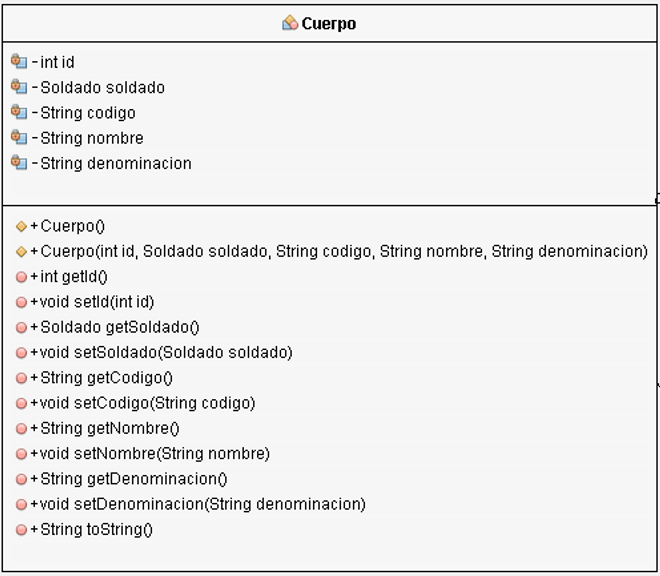
**DESARROLLO**

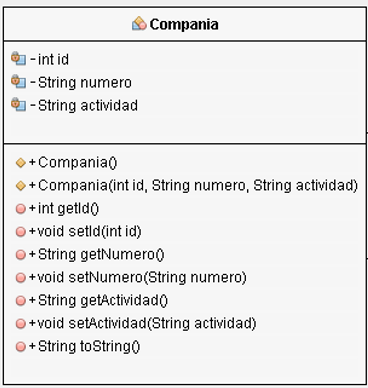
En primera instancia en una propuesta de desarrollo de software se deben de analizar las necesidades del cliente, requerimientos y entorno del caso o problema. Después de haber contextualizado el planteamiento como desarrolladores restringidos a elaborar la solución aplicando el paradigma de Programación Orientado a Objetos se deben identificar las clases en nuestro caso de estudio se encuentran seis clases donde la principal o que conecta con todas las clases es la clase soldado, a continuación, se describirán las clases encontradas recordando que las clases son vistas como un molde que permitirá instanciar o plasmar los objetos, nuestras clases identificadas son la siguientes:

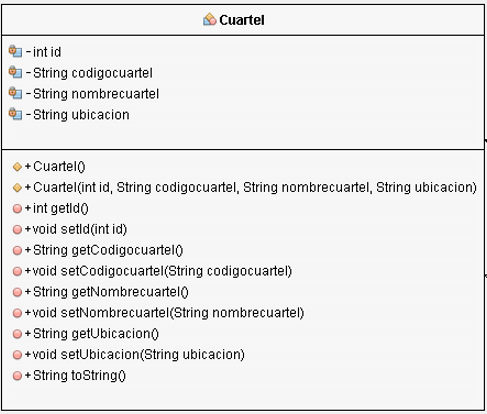
* Soldado
* Cuerpo
* Compania
* Cuartel
* Servicio
* SoldadoServicio

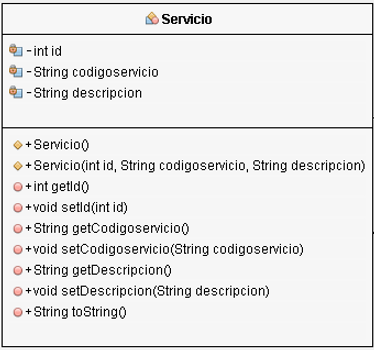
Ahora se proyectará puntualmente las clases anteriormente mencionadas, junto a los atributos que se lograron percibir en estas, además de los métodos implementados.

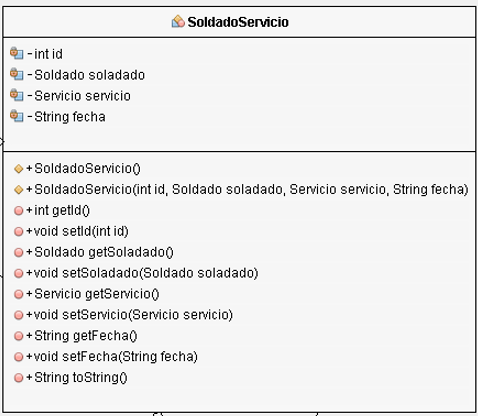




****

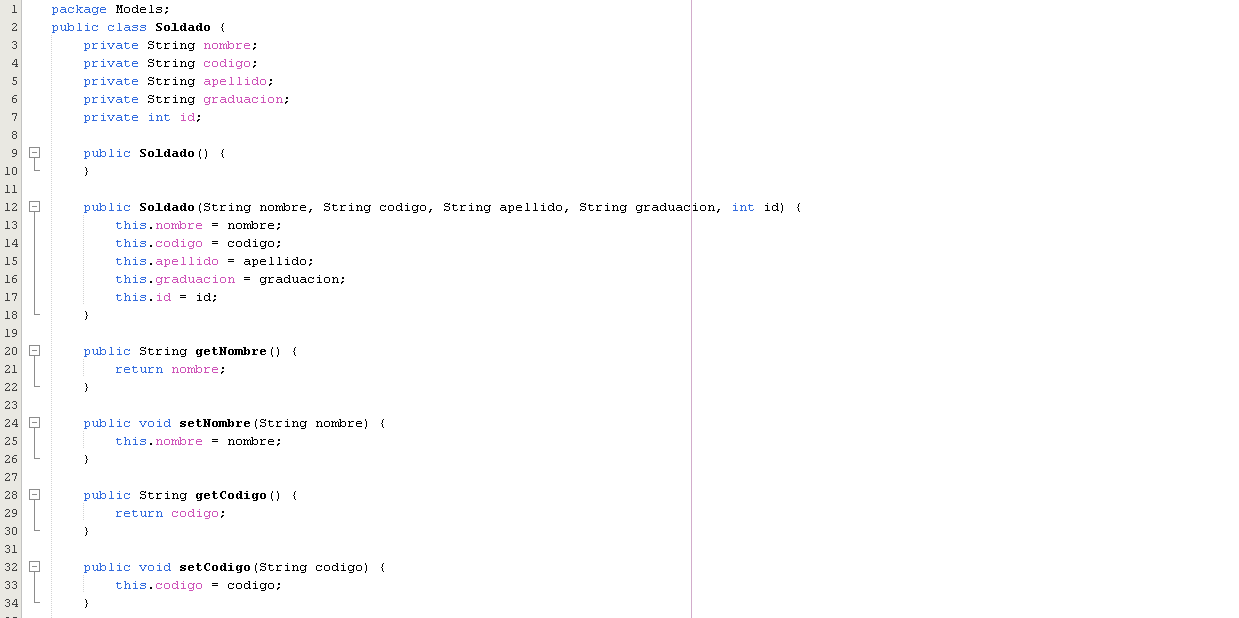
****

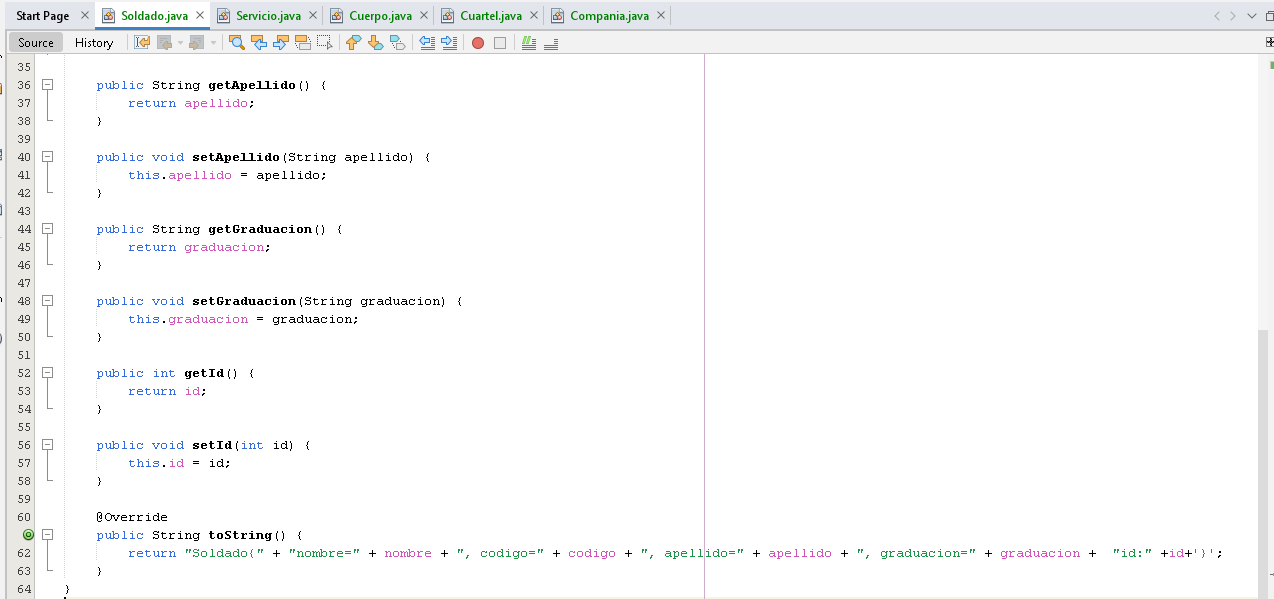
****

****

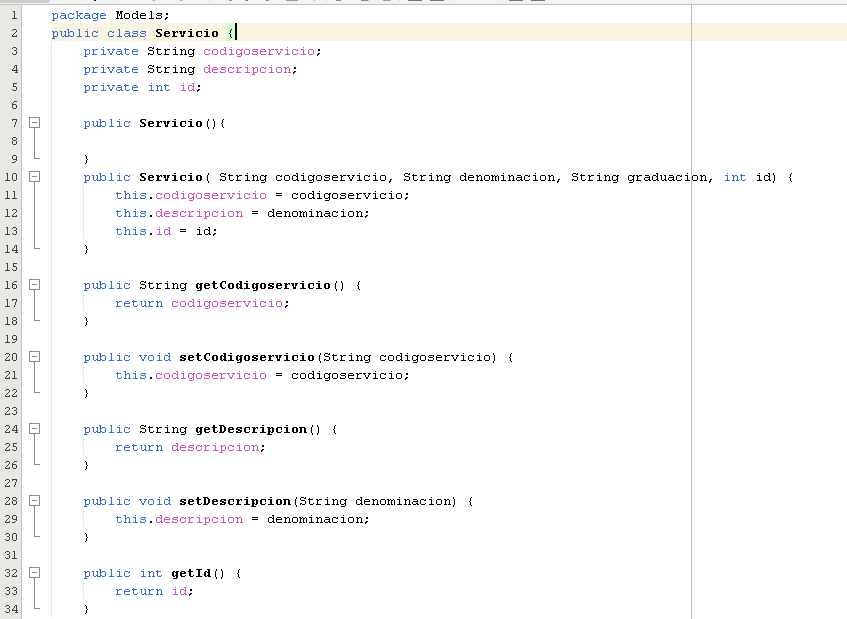
**CODIGO JAVA POO En NetBeans**

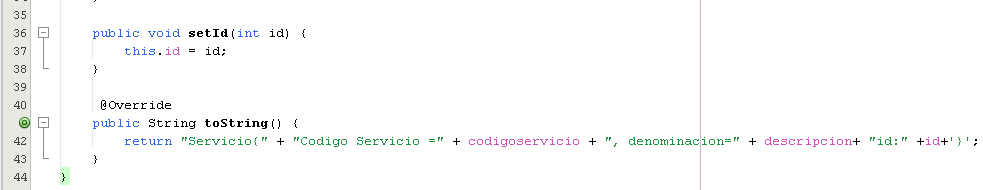
**Clase Soldado (Análisis)**

****

****

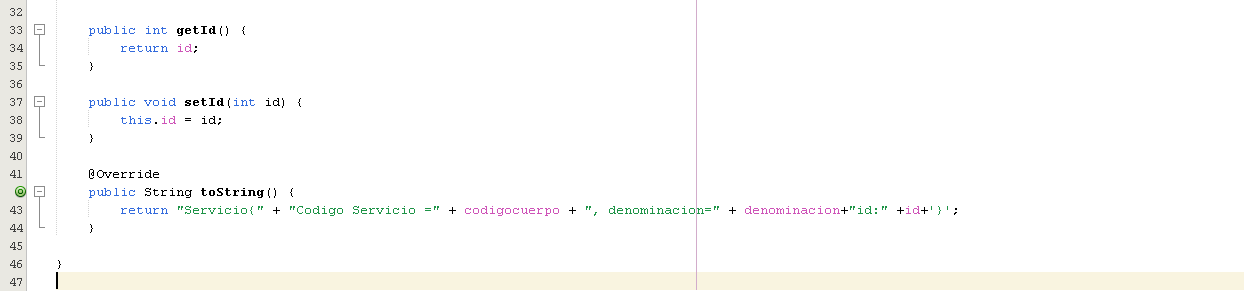
**Clase Servicio (Análisis)**

****

****

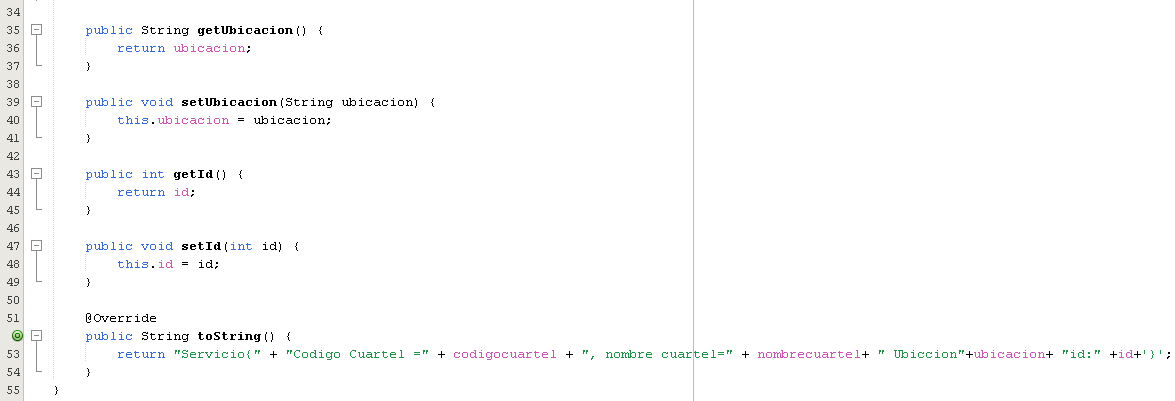
**Clase Cuerpo (Análisis)**

****

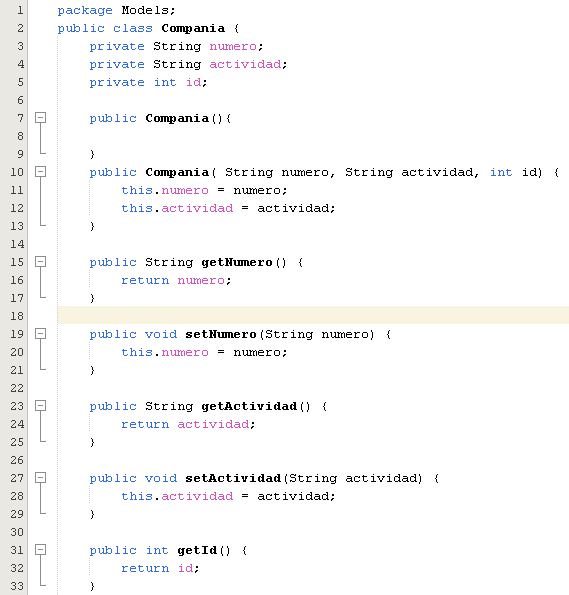
****

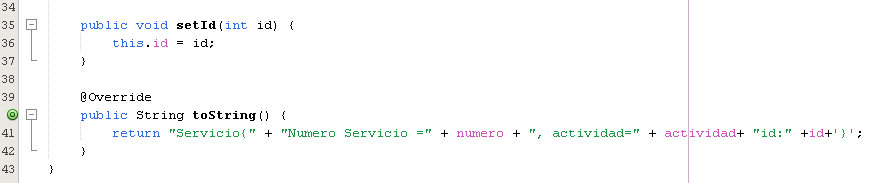
**Clase Cuartel (Análisis)**

****

****

**Clase Compania (Análisis)**

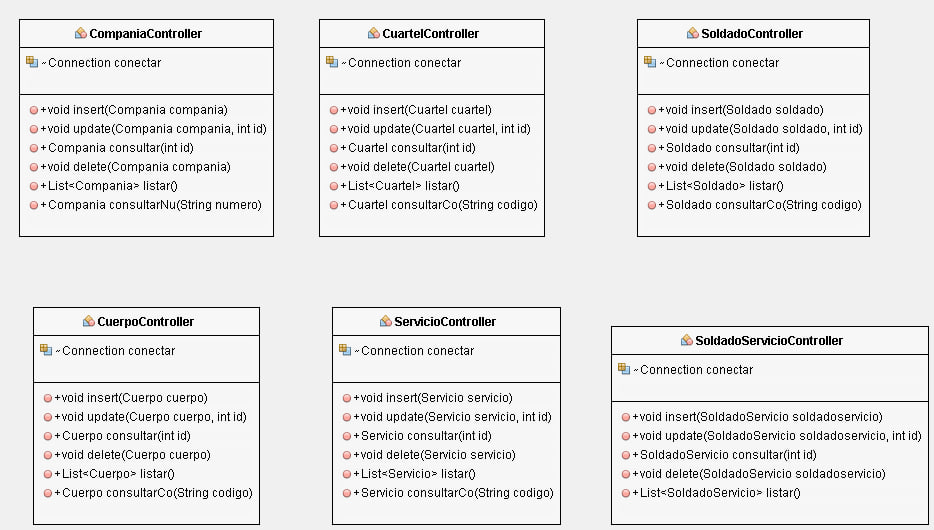
****

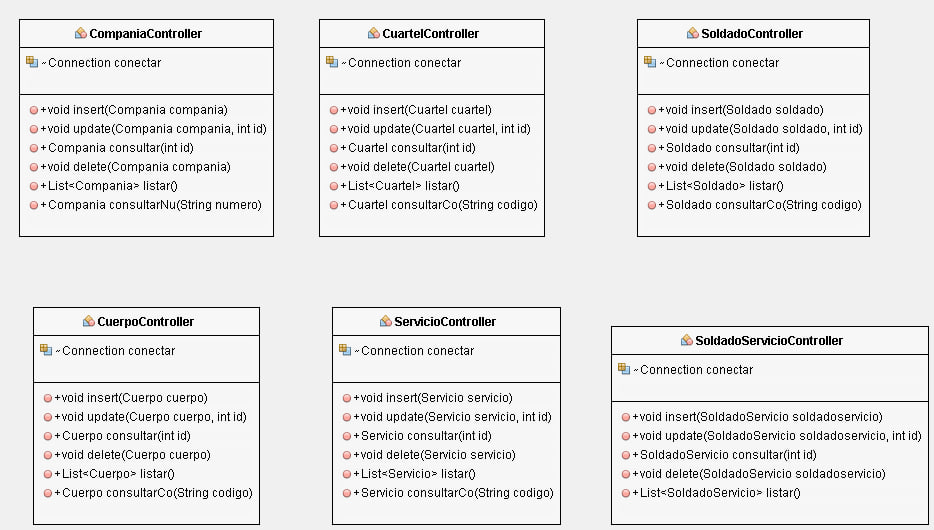
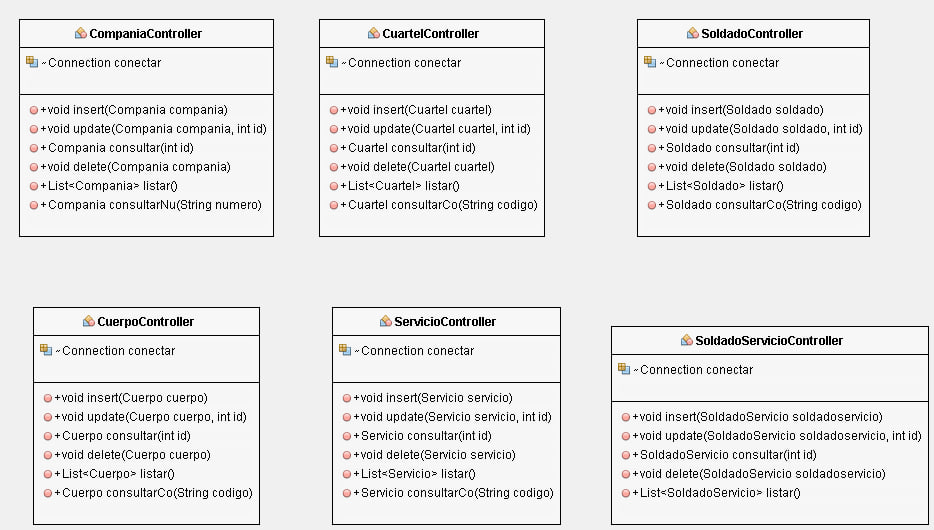
****

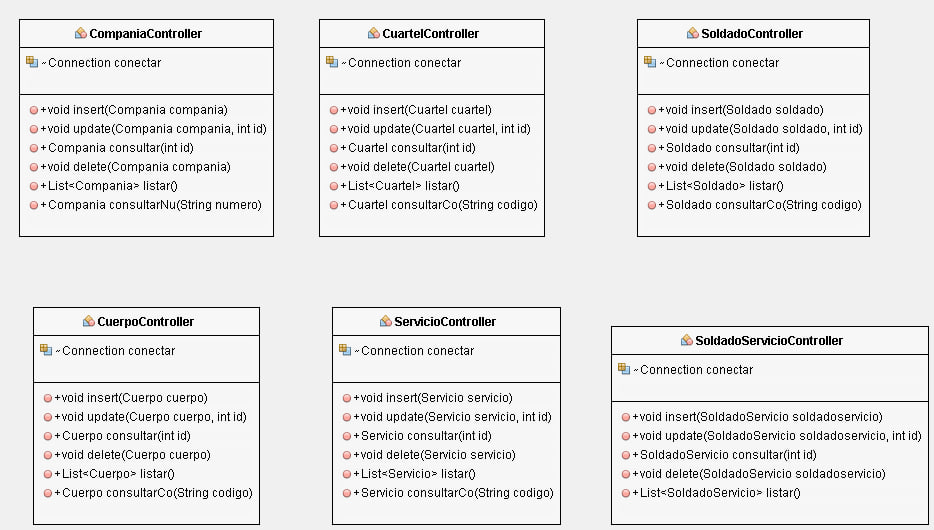
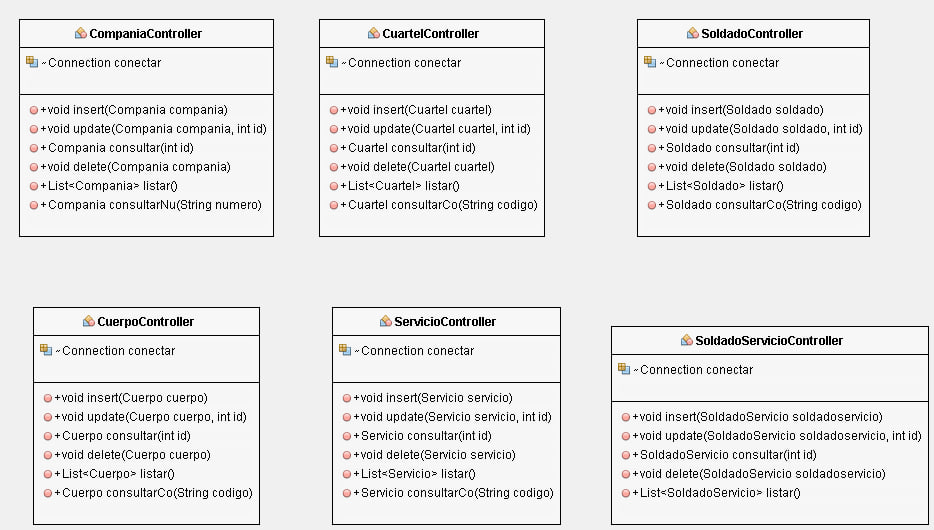
**Paquete Controlador**

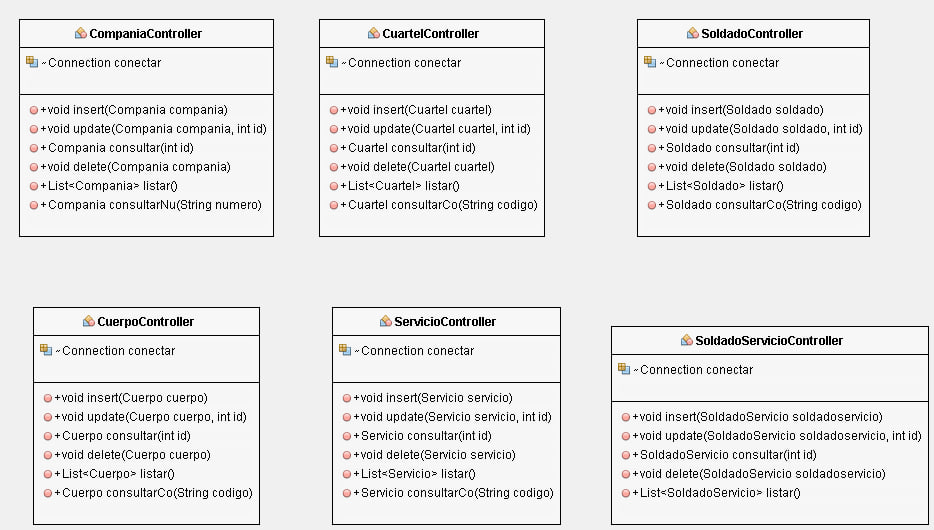
Entre este paquete localizamos las clases que cumplen las consultas a la base de datos “servicio\_militar”. Desempeñando el principio de base de datos de crear, leer, actualizar y eliminar datos, los respectivos paquetes controladores son :

****



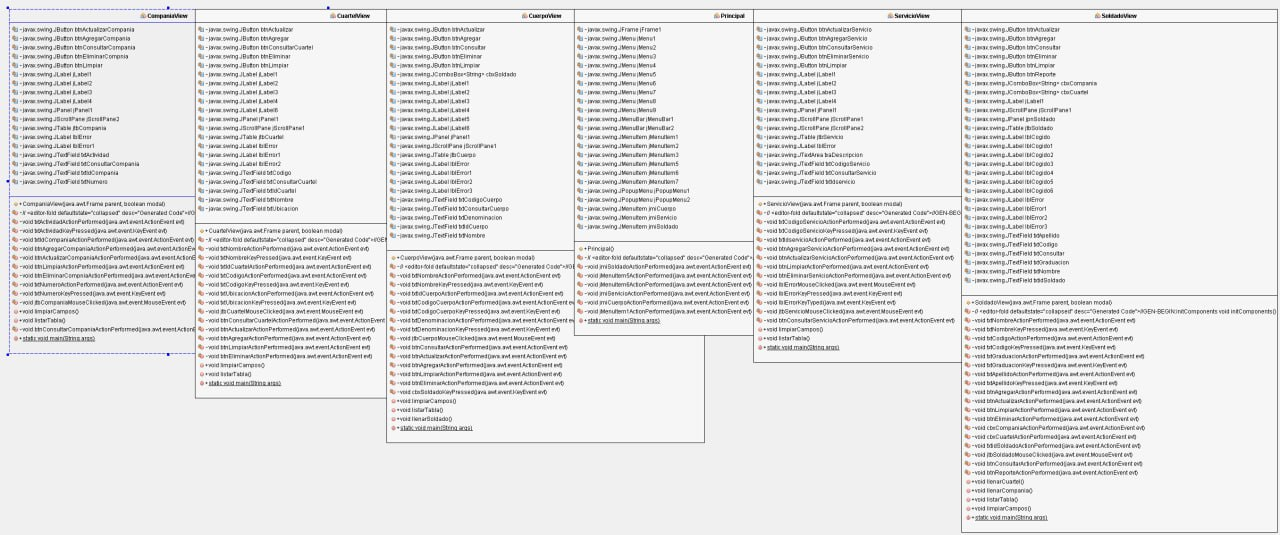
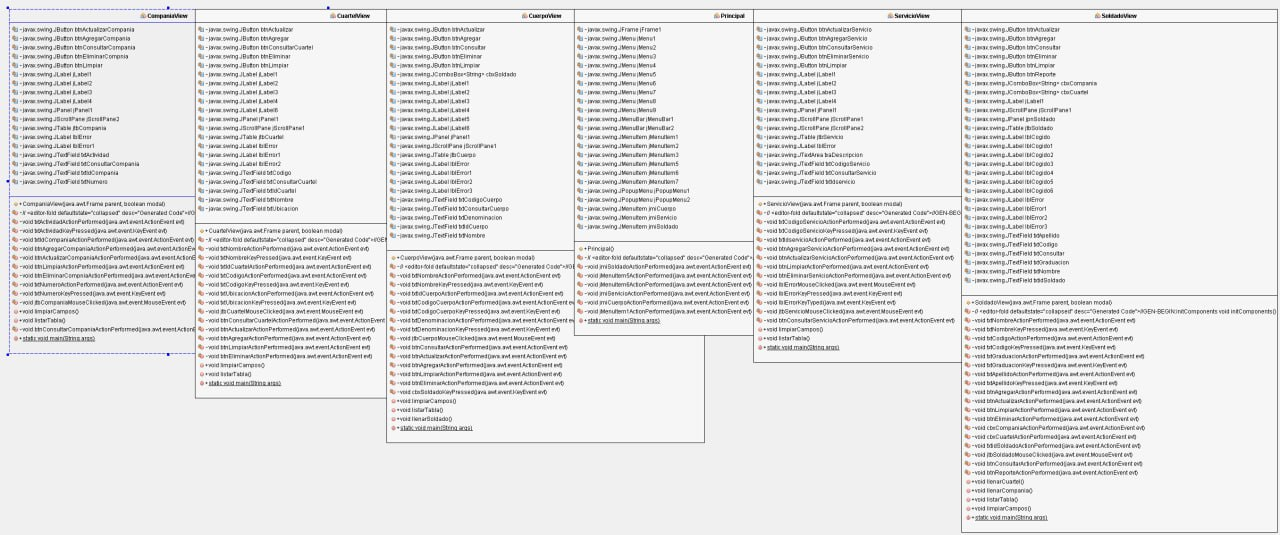


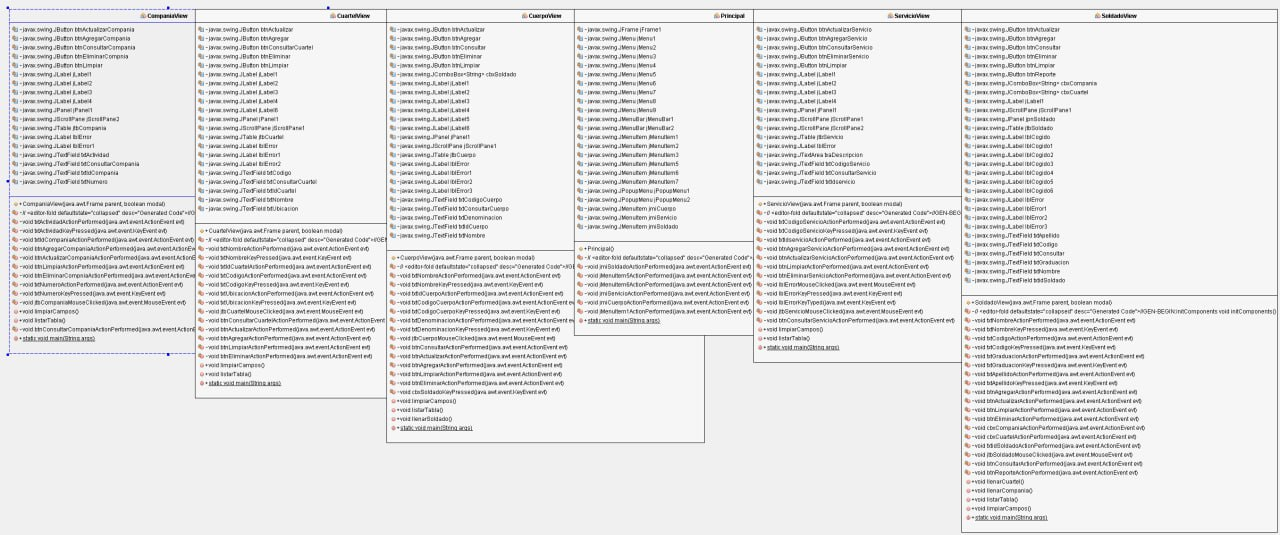
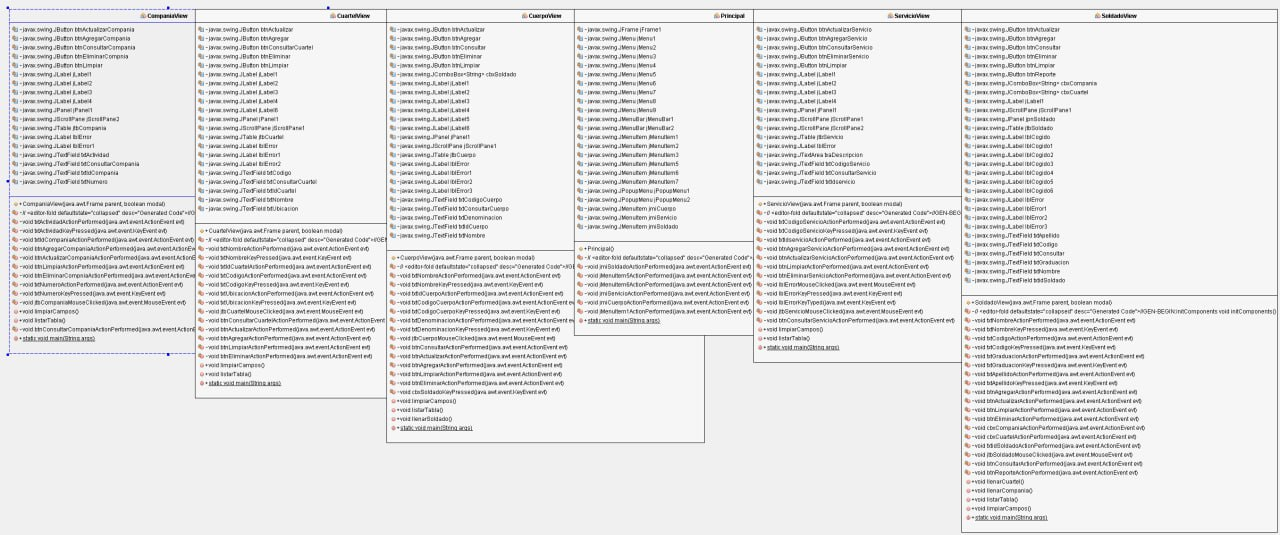


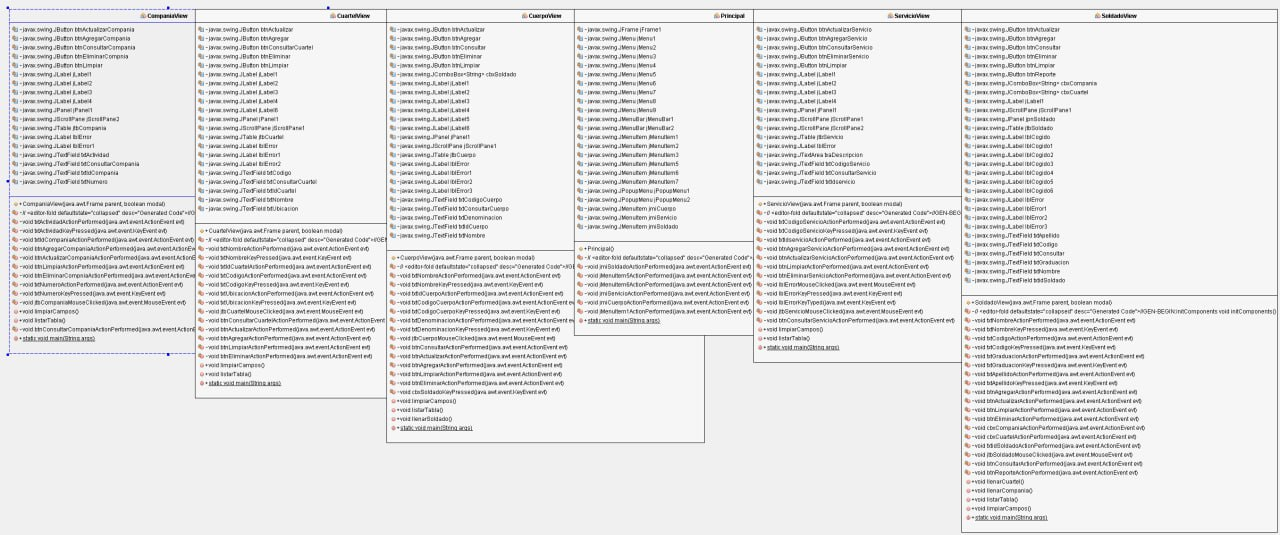
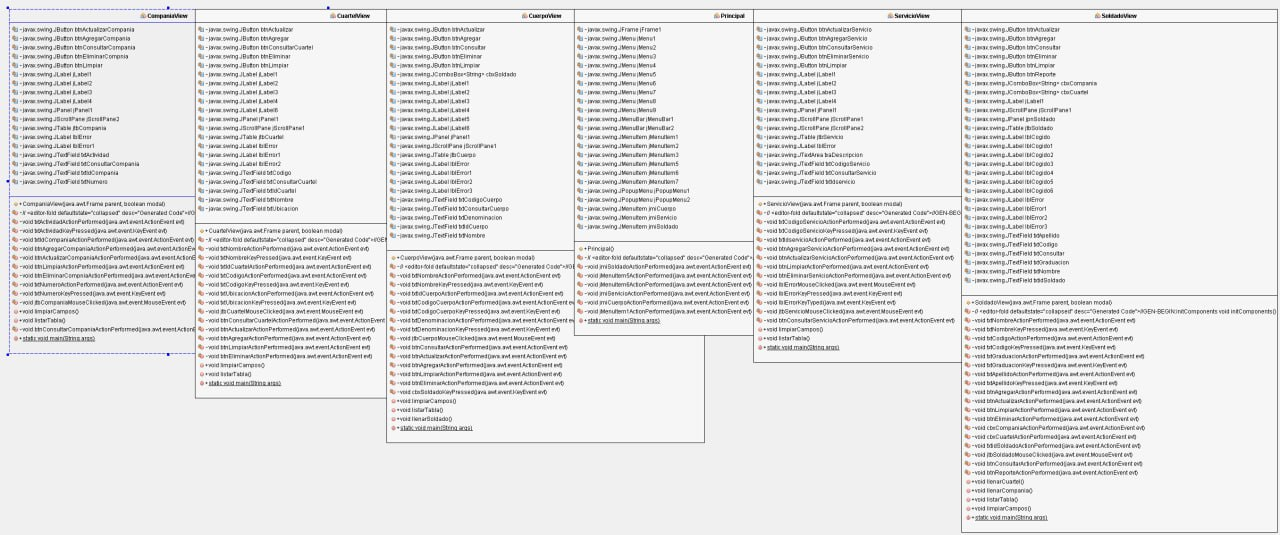
**Paquete Vistas**

Con este paquete tenemos los formularios y la GUI (Graphic User Interface) que accederán para hacer las respectivas capturas de los datos en un entorno amigable para el usuario del software.

****





**JUSTIFICACION**

En las clases planteadas se tiene el paquete modelos, en la clase soldado, creamos los atributos que corresponden a la descripción del problema dado, en esta clase tenemos los siguientes atributos:

● Id tipo entero

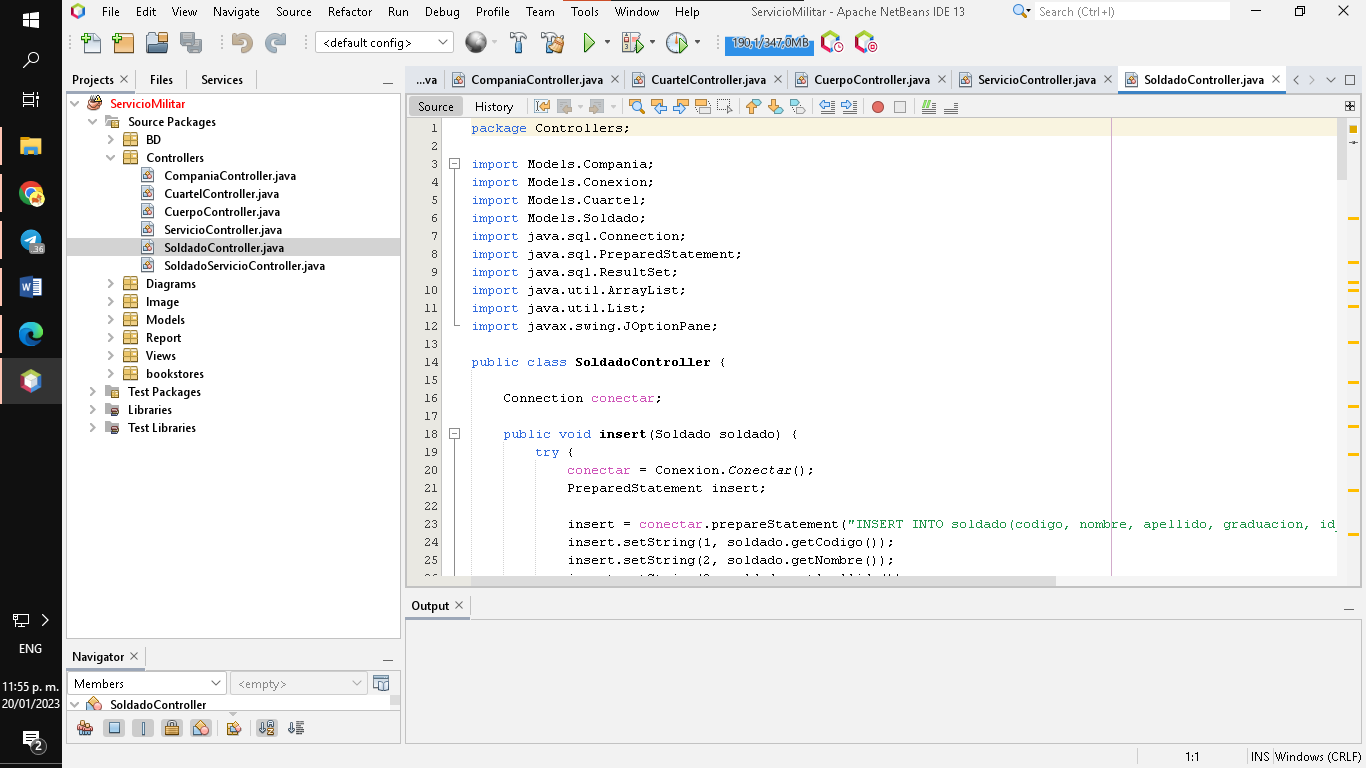
● Nombre tipo String

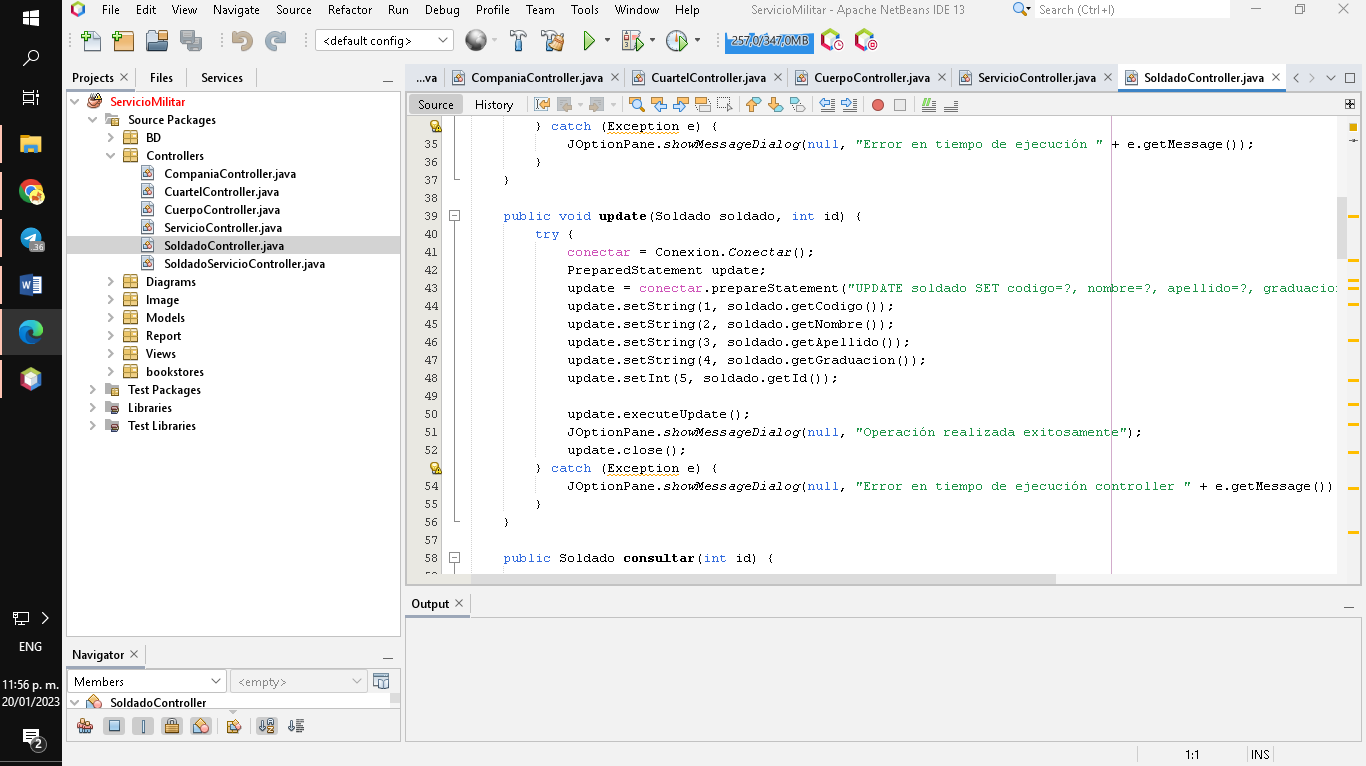
● Código tipo String

● Apellido tipo String

● Graduación destino tipo String

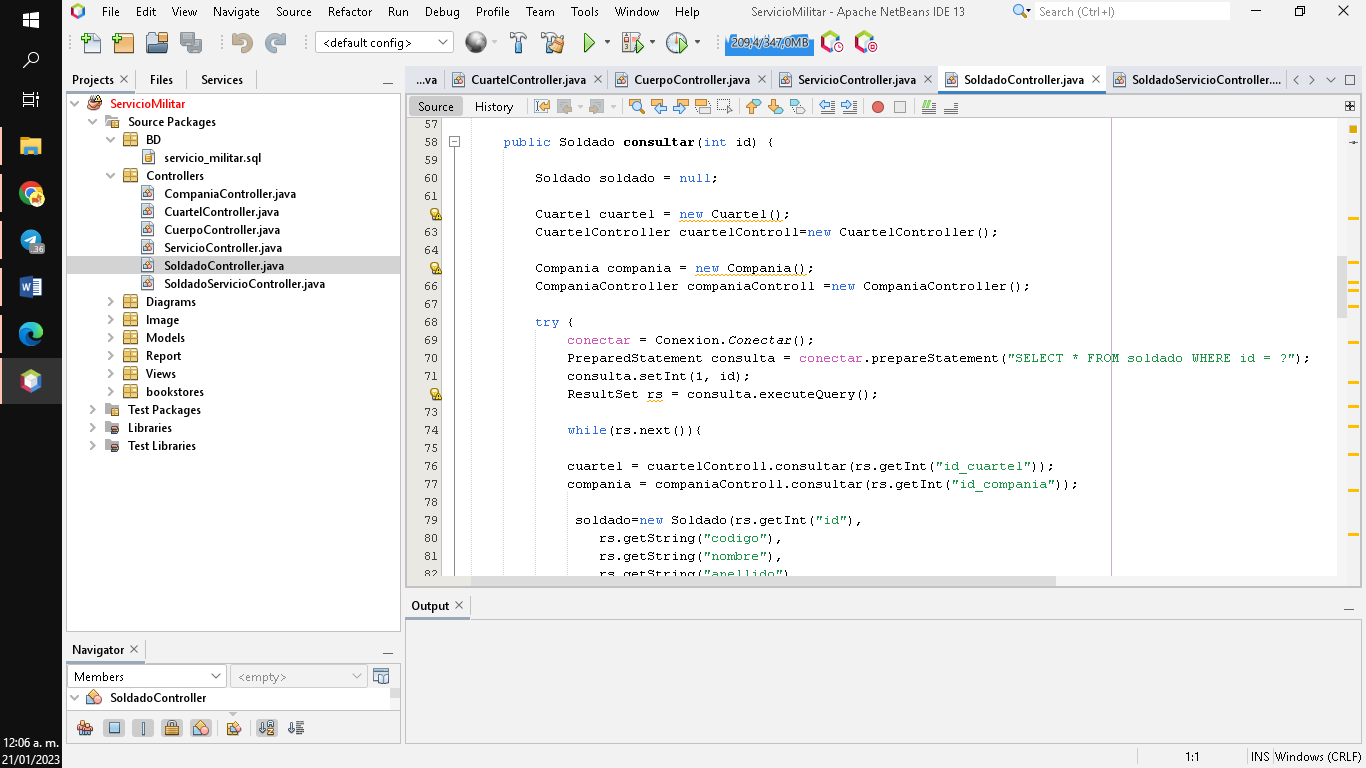
Los atributos inicialmente citados fueron necesarios para poder conseguir los datos a partir de una instancia, en las clases se crean distintos métodos constructores, empleados para settear la información que suministramos a través de ellos, mientras que todos los métodos getter y setter equivalentes a los atributos, cuando se intercambia los datos entre clases, finalmente el método ToString permite mostrar todos los atributos de la clase. En el paquete controllers logrará encontrar todas las clases relacionadas con el CRUD, respectivamente las clases tiene del paquete modells, además de un CURD independiente. A continuación en los siguientes screenshot se visualizaran todos los métodos relacionados con el crud y descripción.



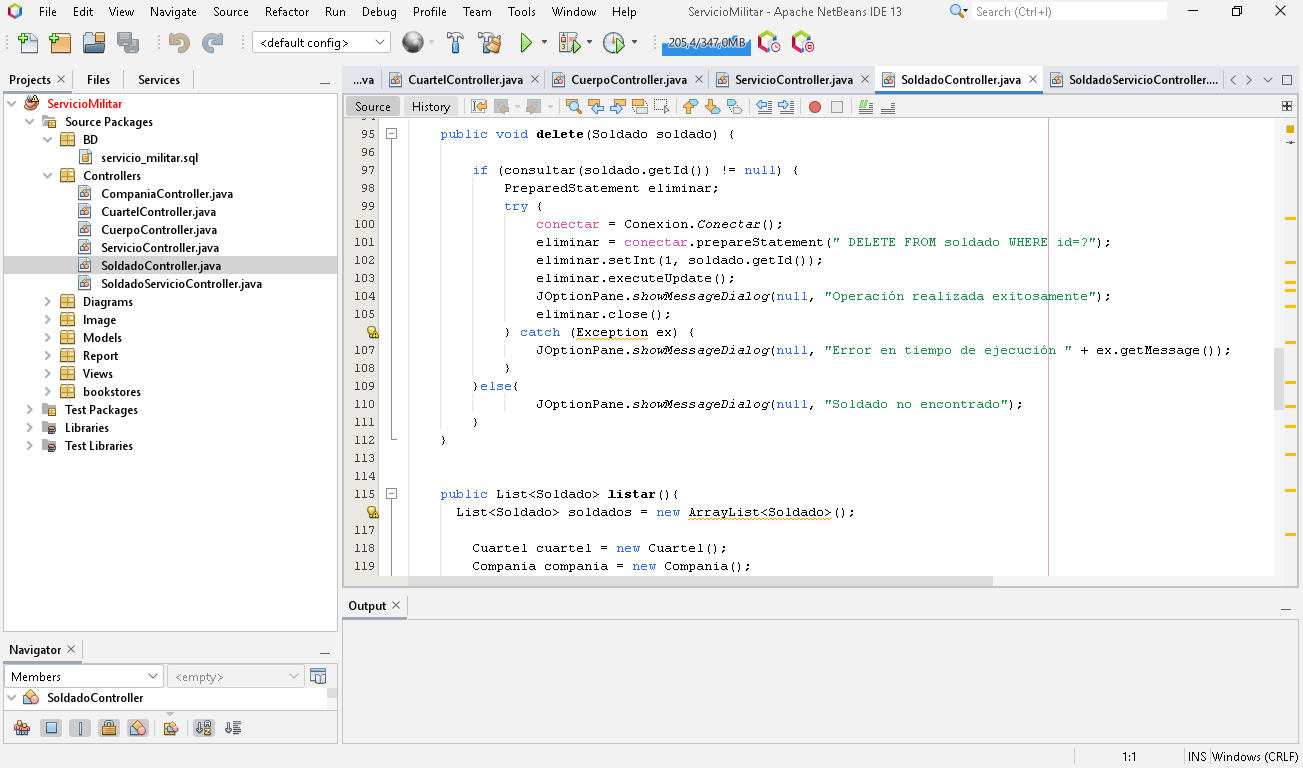


En el método actualizar recibe a **Soldado**, siendo los datos que se van a actualizar o insertar, se importan los paquetes y librerías relacionadas escrutando un óptimo funcionamiento de todos los métodos dados.

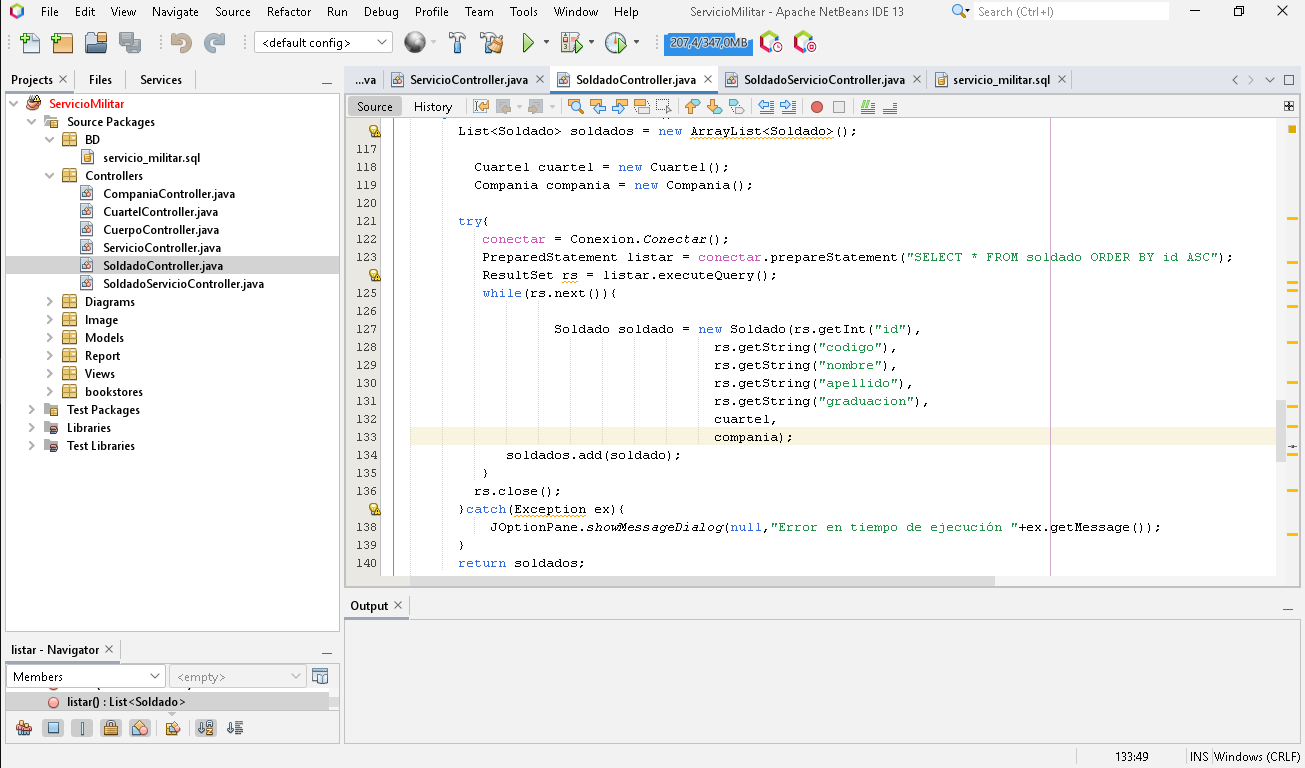
Dentro del método declarado como “actualizar”, se tienen dos acciones a desarrollar donde una es “insert” y la otra “actualizar”, mediante la sentencia sql "INSERT INTO " + Nombre de la tabla + "(Codigo, Nombre, Apellido) VALUES, nos permite registrar a los soldados en la base datos, utilizamos los getter para adquirir la información antes enviada mediante los métodos setter. Para renovar los campos en la base de datos, le hacemos con la sentencia UPDATE, actualizando los campos dependiendo del id, que le suministramos en la sentencia sql, como podemos ver en la imagen.

****

En el método llamado consultar se recibe como parámetro un valor de tipo entero, por el cual nos aprobará hacer la consulta insertando como dato el dato ID para poder realizar la respectiva consulta.

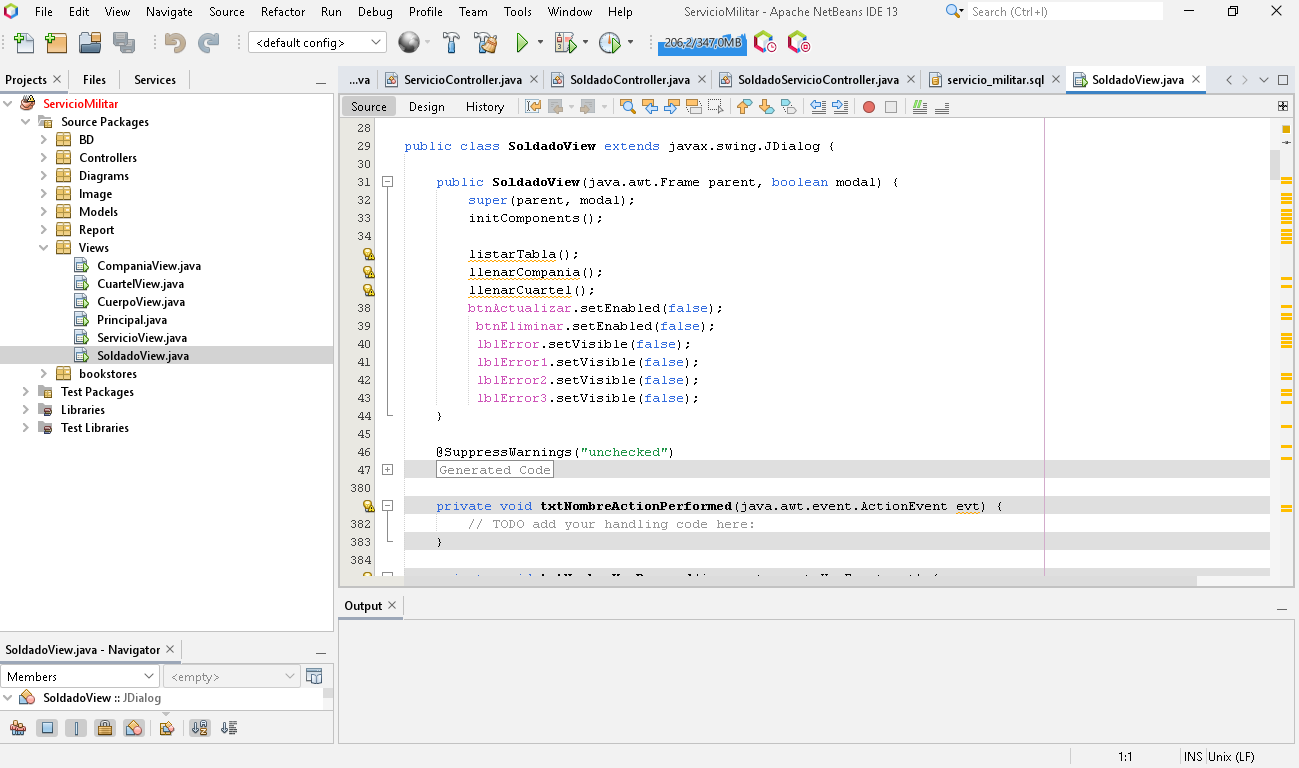
****

“UPDATE” + THIS.TABLA + “SET estado =? WHERE id =?”. Hace que la información cambie de estado dando un concepto de eliminación en el CRUD

****

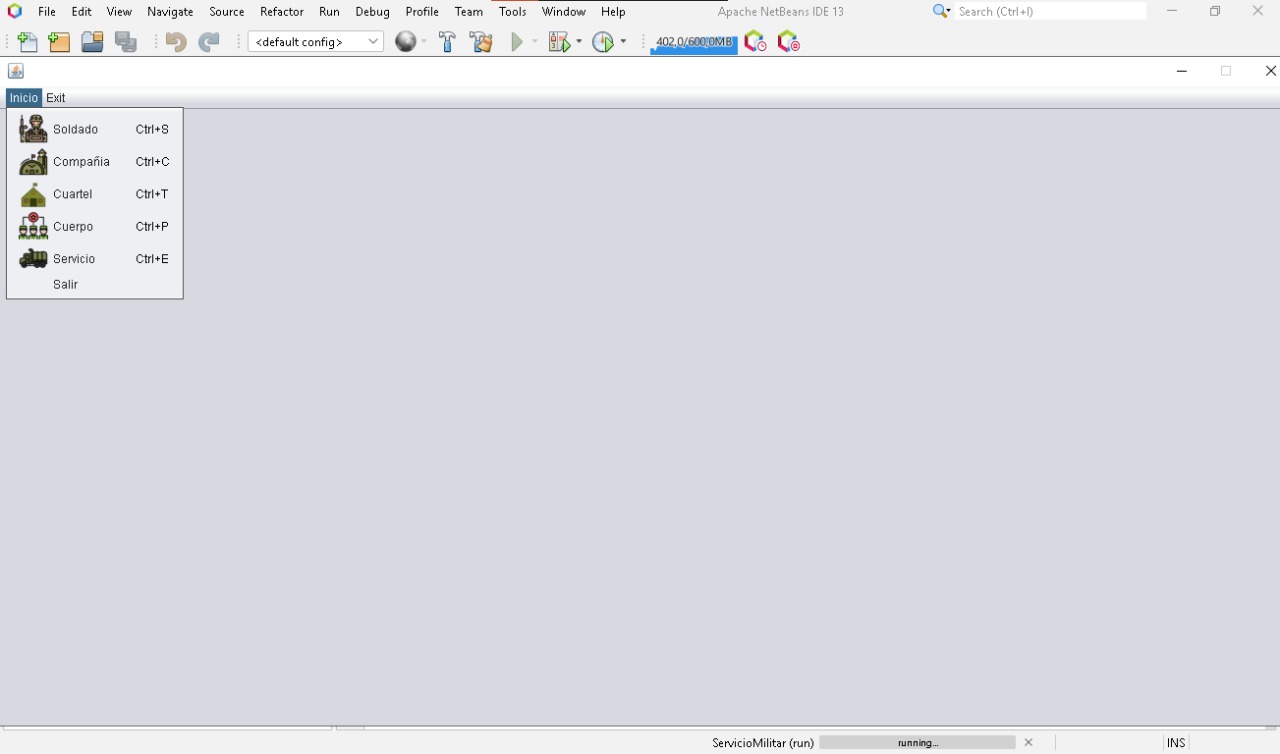
**List** recibe los datos de soldado

En el paquete el package “view” se encuentra relacionado con las clases de tipo javaDialog, con varios componentes que nos ayudan a interactuar con el usuario del software donde conectaremos los modelos y los controladores.

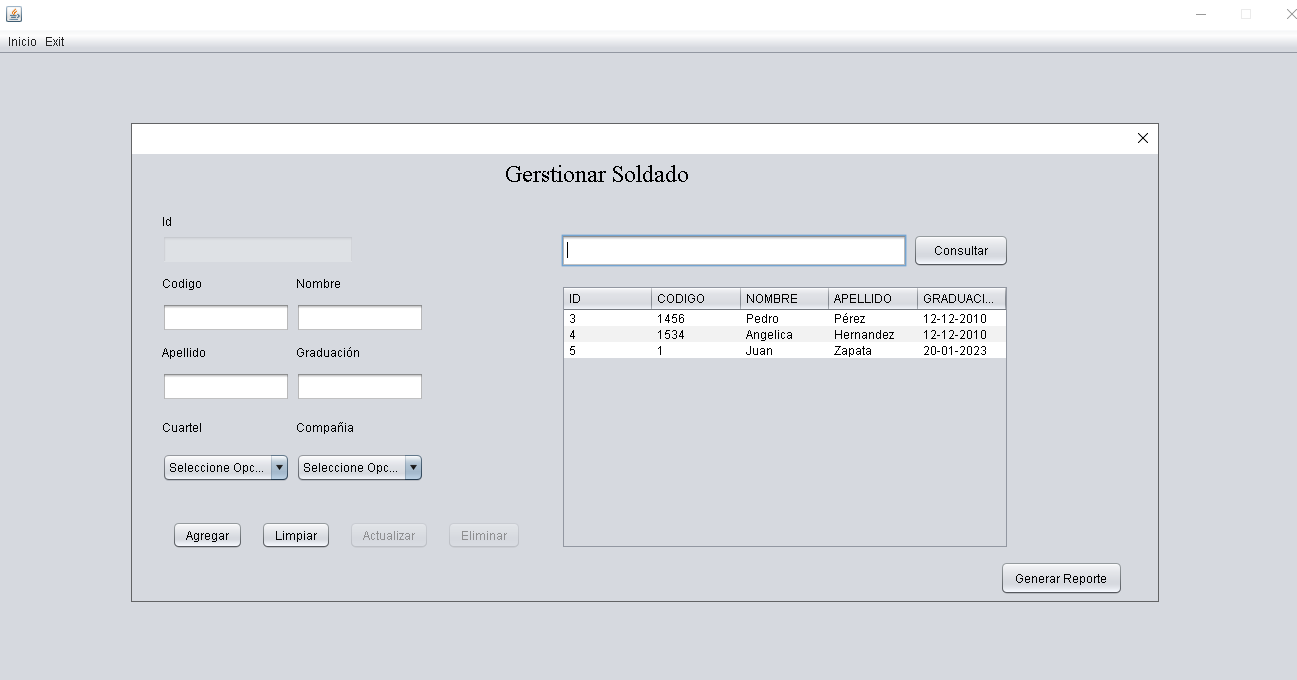
****

**DISEÑO GUI Y MAQUETA**

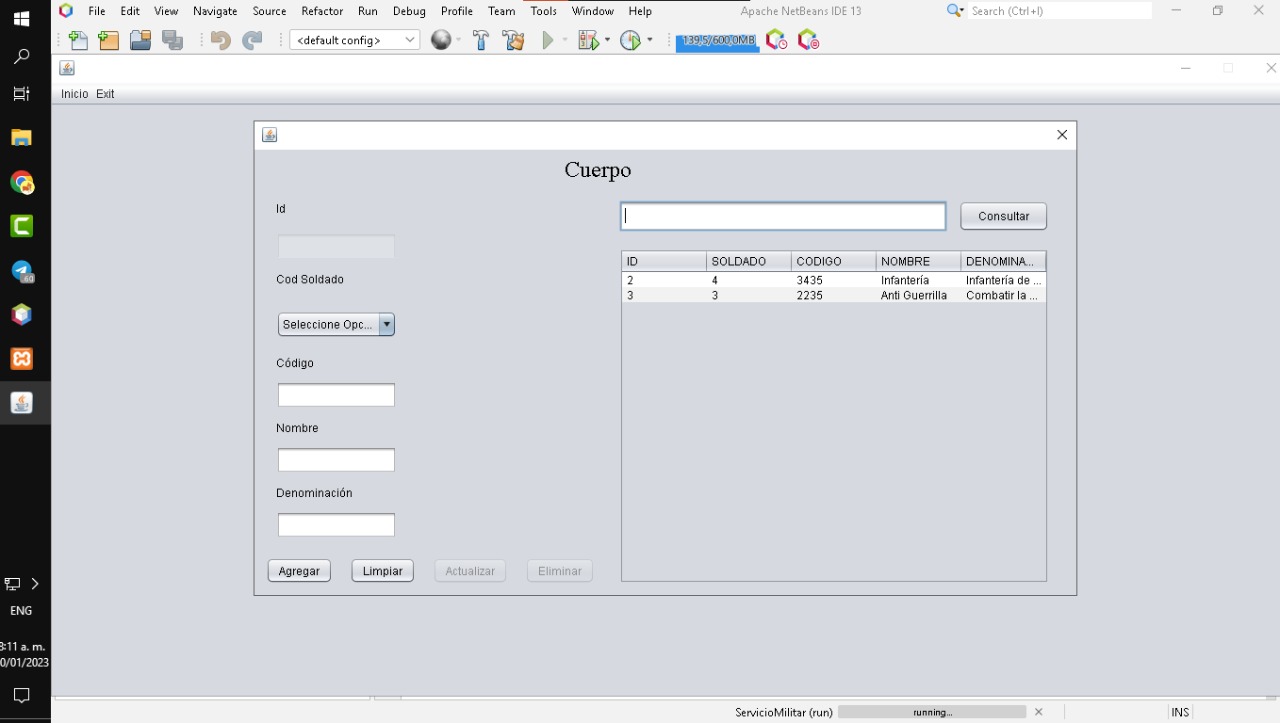
La primera ventana, a lo que se conoce como el Home, está hecho de un JFrame el cual tiene un botón de inicio y salida, dicho botón tiene 6 JMenuItem, los cuales son: Soldado, Compañía, Cuartel, Cuerpo, Servicio salir. Como se puede observar en la siguiente imagen:

****

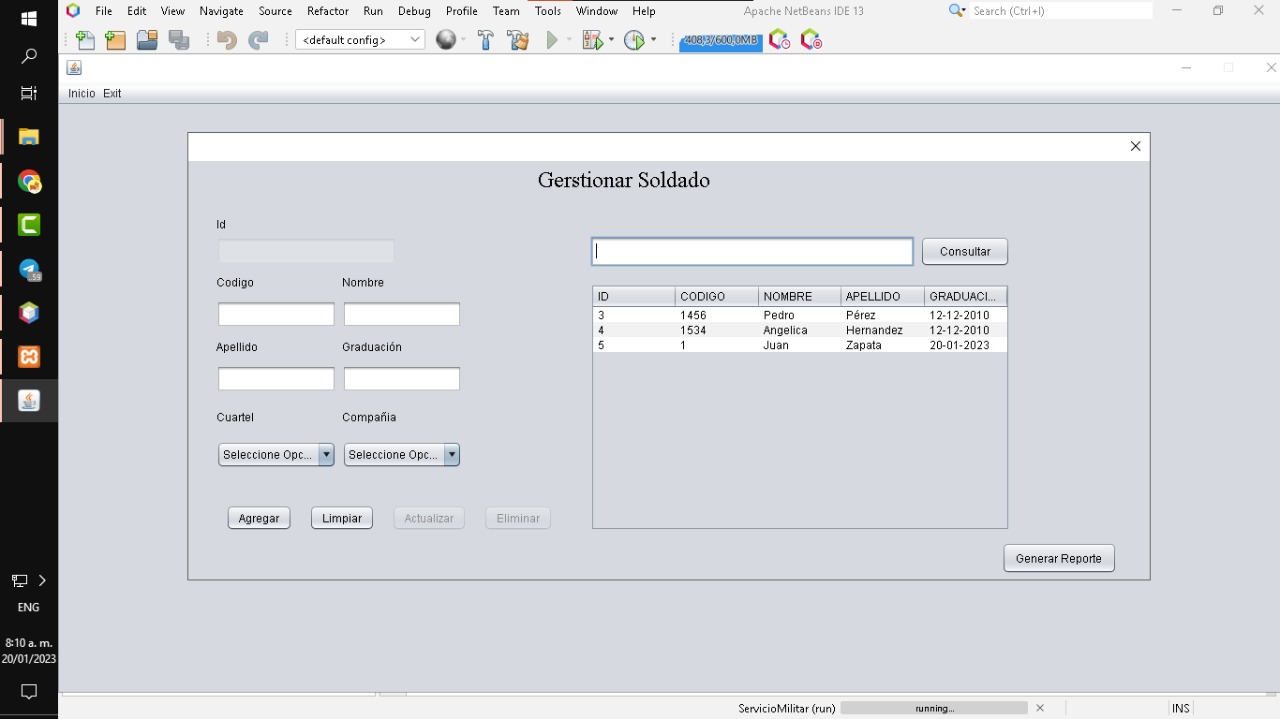
Esta vista se llama Soldados, tiene como función tomar los datos tales como ID, Nombre, Apellidos, Código, Graduación, Escoger Cuartel y Compañía en función de los datos ya consignados en el CRUD el botón AGREGAR, guarda todos los datos de dicho soldado. Una vez realizado el proceso de guardado se puede limpiar todos los campos presionando el botón LIMPIAR. En la parte derecha de la vista se encuentra el botón CONSULTAR, en el cual se debe escribir el ID del militar que se quiere encontrar.

****

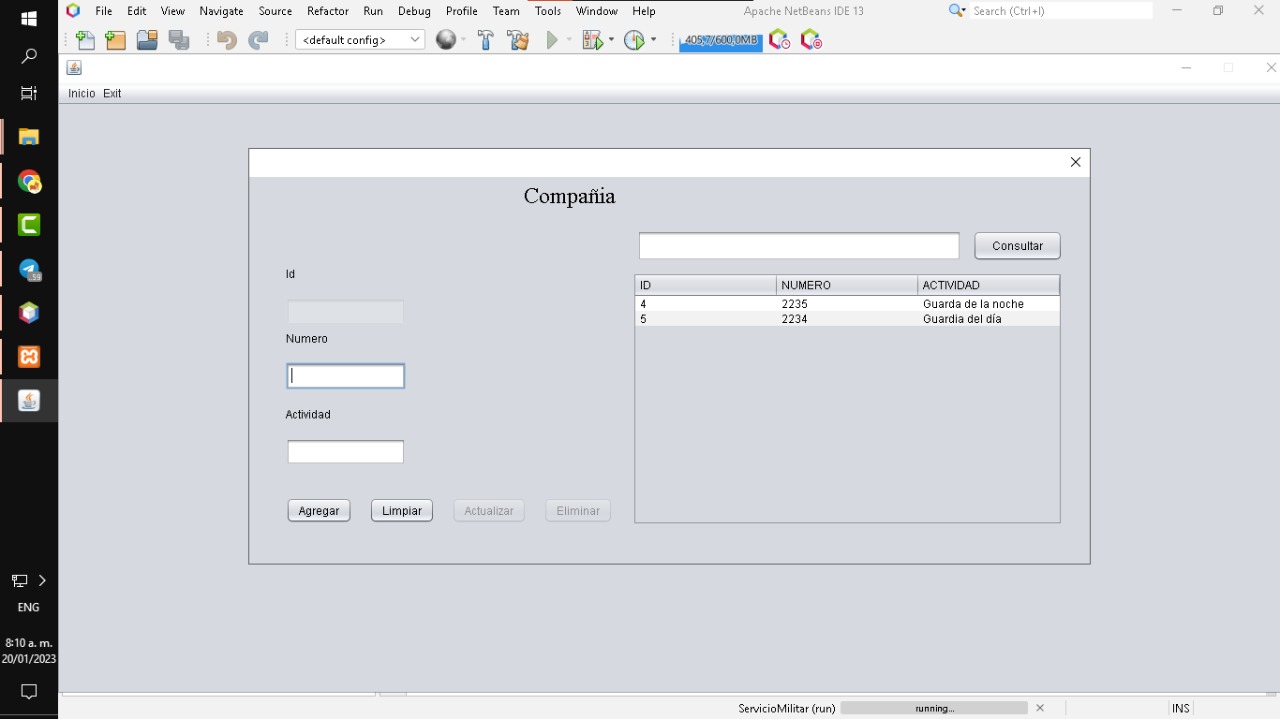
Esta vista se llama Cuerpo, tiene como función tomar los datos tales como ID, Código, Nombres, Denominación, en el CRUD el botón AGREGAR, guarda todos los datos de dicho soldado. Una vez realizado el proceso de guardado se puede limpiar todos los campos presionando el botón LIMPIAR. En la parte derecha de la vista se encuentra el botón CONSULTAR, en el cual se debe escribir el ID del militar que se quiere encontrar.



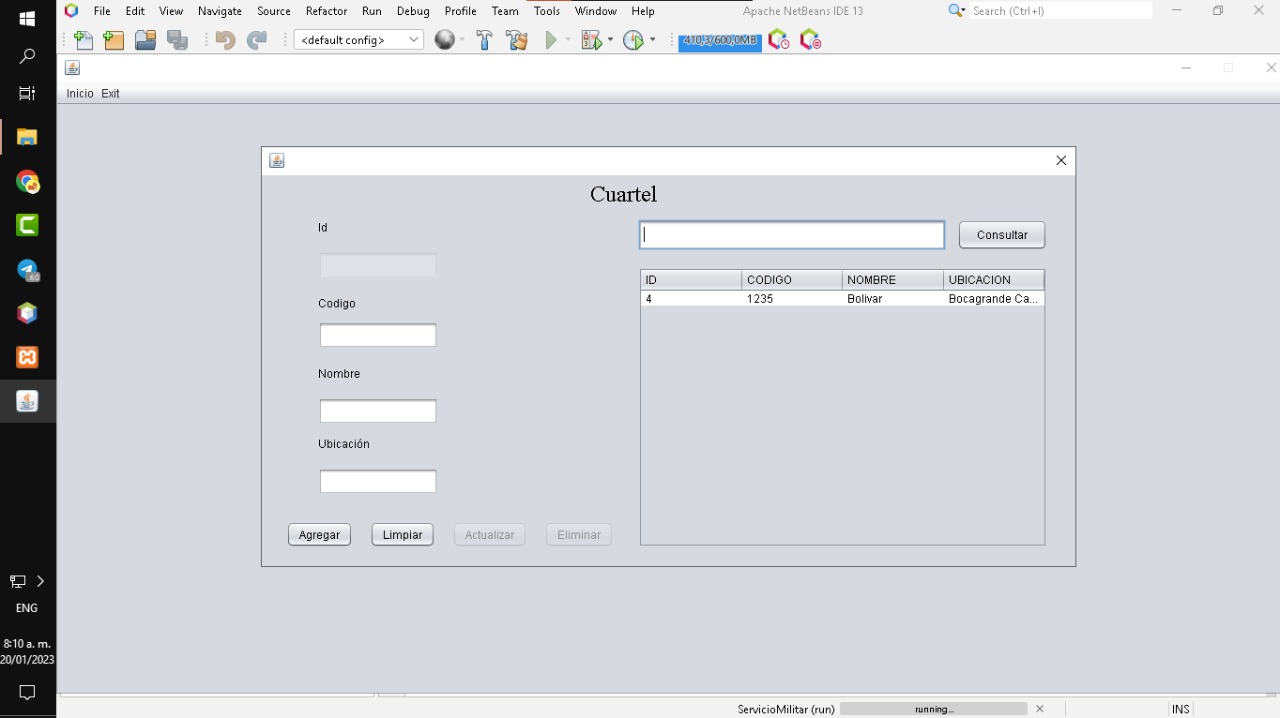
Esta vista se llama Gestionar Soldado, tiene como función tomar los datos tales como ID, Código, Nombres, Graduación, Escoger entre los campos cuartel y compañías ya grabados. en el CRUD el botón AGREGAR, guarda todos los datos de dicho soldado. Una vez realizado el proceso de guardado se puede limpiar todos los campos presionando el botón LIMPIAR. En la parte derecha de la vista se encuentra el botón CONSULTAR, en el cual se debe escribir el ID del militar que se quiere encontrar.



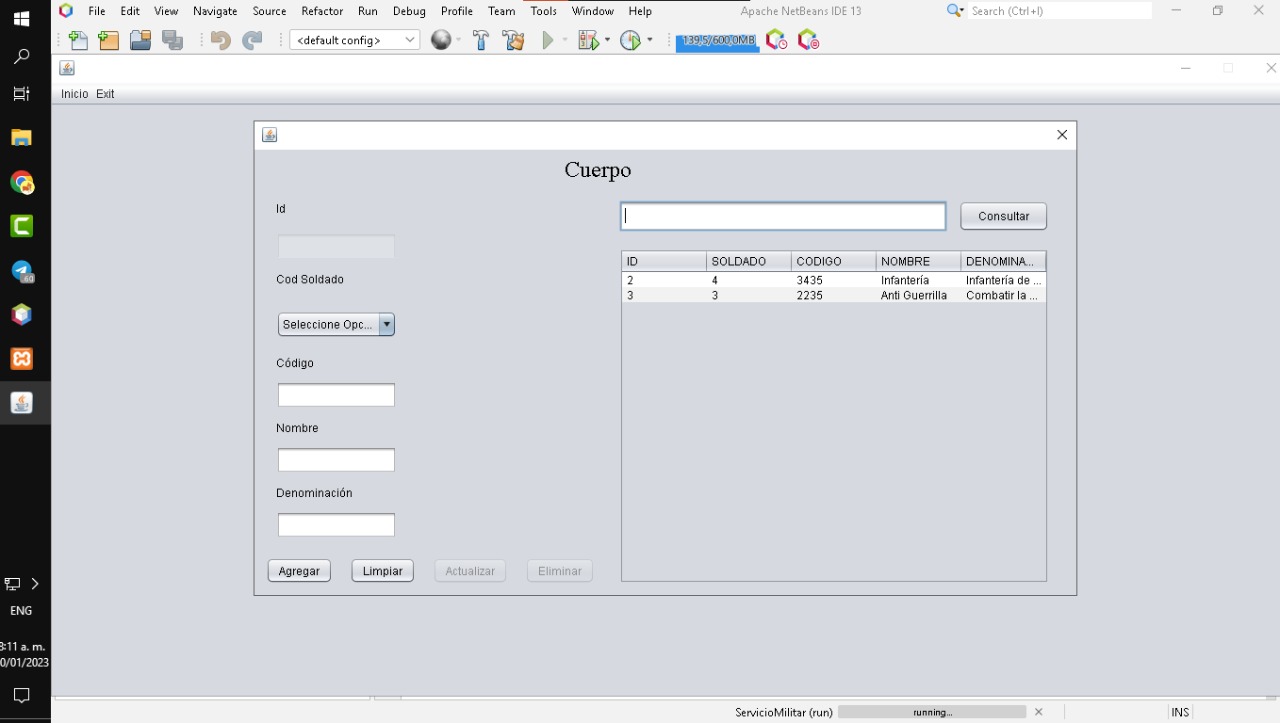
Esta vista se llama Compañía, tiene como función tomar los datos tales como número de la compañía y actividad Graduación botón AGREGAR, guarda todos los datos de dicho soldado. Una vez realizado el proceso de guardado se puede limpiar todos los campos presionando el botón LIMPIAR.



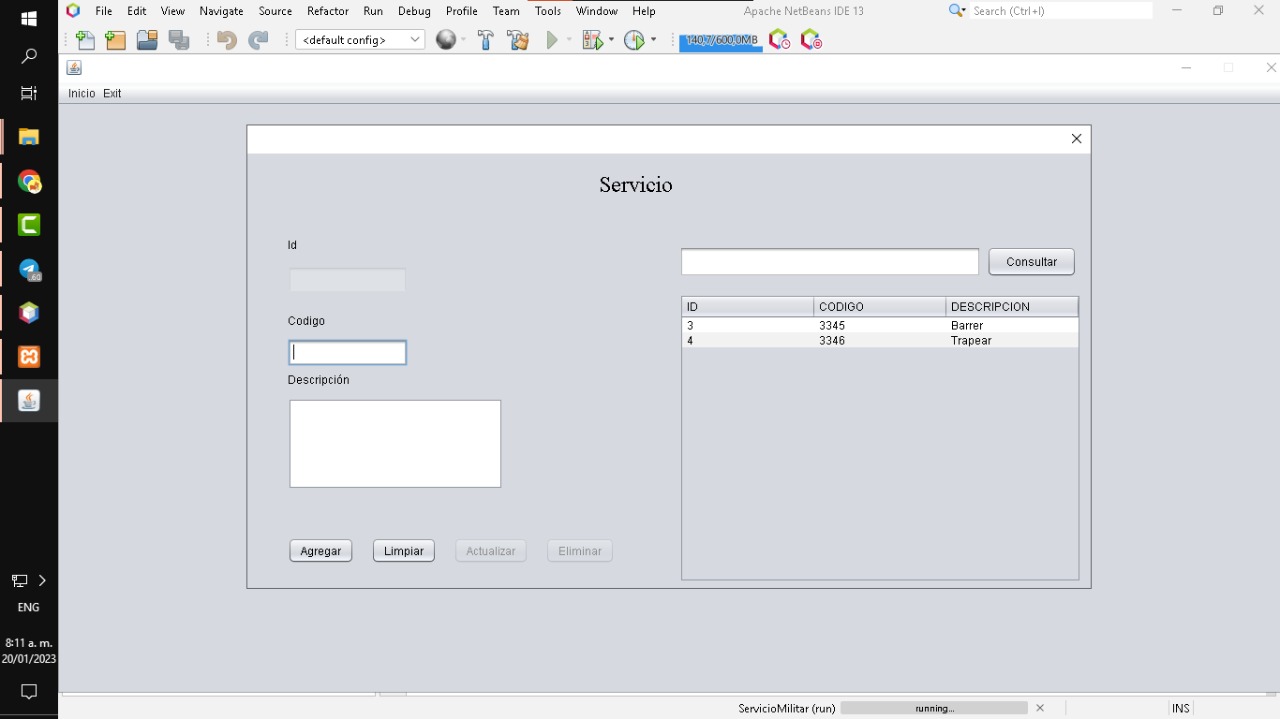
Esta vista se llama Cuartel, tiene como función tomar los datos tales como nombre, código y ubicación el botón AGREGAR, guarda todos los datos de dicho soldado. Una vez realizado el proceso de guardado se puede limpiar todos los campos presionando el botón LIMPIAR.

****

Esta vista se llama Cuerpo, tiene como función tomar los datos tales como nombre, código y denominación el botón AGREGAR, guardar datos. Una vez realizado el proceso de guardado se puede limpiar todos los campos presionando el botón LIMPIAR.



Esta vista se llama Servicio, tiene como función tomar los datos tales como código del servicio, código y descripcion del servicio, el botón AGREGAR, guarda los datos. Una vez realizado el proceso de guardado se puede limpiar todos los campos presionando el botón LIMPIAR.



**Conclusión**

El paradigma orientado a objetos emerge debido a las grandes extensiones de código y código espagueti cuando no aplicamos dicho paradigma porque aplicando sistemas para reducir líneas de código, reutilizar variables y clases por medio de la instancia contrae la dimensión del código y el esfuerzo en programación y solución de problemas. Buscando simular a través de clases, junto atributos respectivos sustantivos de la realidad y a través de los métodos las acciones que ejecutaría en el contexto del problema a solucionar. Siendo un objeto un ente de cualidad y funciones, que combina estos elementos para reaccionar a eventos. Un objeto como se ha mencionado es la instancia de una clase. Entendiéndose clases como la definición de cada una de las propiedades y funcionalidades de un objeto.

Para comprender como se desempeña el paradigma de la programación orientada a objetos es preciso visualizar un programa como una recopilación de objetos que tiene una interacción y cambiando de estados e información a través de la ejecución, el haber trazado el problema y solucionado implementándolo a través de la programación orientado a objetos implica determinar y caracteriza los diferentes objetos que intervienen en el problema, desde definir sus propiedades y métodos y poder interactuar.

En el transcurso del proyecto también se implementaron tecnologías y conceptos como es base de datos relacionales, bases de datos sql y el concepto de CRUD llamado de esta forma por las siglas de CREATE, READ, UP DATE y DELETE, los procesos que se plasmaron en el proyecto para almacenar información, leer los datos, actualizar la base de datos y eliminar la información . Si ninguna de estas tecnologías se implementaran en el proyecto se desplegaría el fenómeno conocido como código espagueti cuando un algoritmo o proyecto de programación se hace tan extenso que no es entendible y complejo de analizar. Por otro lado, se tendrían que crear N variables en función de los soldados a almacenar en nuestro caso, siendo muy extenso, además de no dinámico para el proyecto.

# **Bibliografía**

Alvarez, M. A. (18 de 09 de 2021). *Desarrollo Web*. Obtenido de Desarrollo Web: https://desarrolloweb.com/articulos/499.php

*https://app.diagrams.net/*. (26 de 11 de 2022). Obtenido de https://app.diagrams.net/: https://drive.google.com/file/d/1JEPWSsEfHbiqbwYmxcQPLMQigLj5SM9O/view?usp=sharing

Juarez, I. (22 de 01 de 2021). *Platzi*. Obtenido de Platzi: https://platzi.com/tutoriales/1545-c-plus-plus/9141-que-es-la-programacion-orientada-a-objetos-poo/

Luisina. (30 de 09 de 2021). *TodoCode*. Obtenido de Youtube: https://www.youtube.com/watch?v=tcza2FEz4u4&list=PLQxX2eiEaqbwNP20GMMCjRslRq2lOLWlg&ab\_channel=TodoCode

OpenBootcamp. (26 de 11 de 2022). *OpenBootcamp*. Obtenido de OpenBootcamp: https://open-bootcamp.com/cursos/java/crear-clases

vega, A. S. (26 de 11 de 2022). *Platzi*. Obtenido de Platzi: https://platzi.com/tutoriales/1474-oop/9009-apuntes-del-curso-de-programacion-orientada-a-objetos/