MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

UNIT CONTROL MILITARY

Autor: Manuel Domínguez Piña

Fecha de entrega: 02/05/2023 (Actualización 2025)

Convocatoria: 2S – 22/23

Unit Control Military Desarrollado por Manuel Domínguez Piña. Este proyecto fue desarrollado inicialmente en 2023 como parte de mi formación, y ha sido actualizado y revisado en 2025.

Contenido

1. Introducción

Unit Control Military, es un proyecto destinado a ser utilizado dentro del ámbito de las fuerzas armadas, donde su principal función será el almacenar, actualizar, borrar y consultar la información perteneciente a los medios humanos y materiales, dentro de una compañía militar.

Para realizar este proyecto, hemos optado por desarrollar una aplicación de escritorio, con la idea de que pueda ser usada en los equipos de las oficinas de las diferentes compañías.

Además, para su elaboración utilizaré diferentes tecnologías, las cuales están desarrollas en los siguientes puntos de esta memoria, como puede ser Java, un lenguaje de programación de alto nivel, que es de los mas usados a nivel hoy en día a nivel mundial.

Por otro lado, haré uso de **Swing** que es una biblioteca que nos proporciona Java con la que construiremos la interfaz, y **NetBeans**, que es el IDE (Entorno de Desarrollo Integrado), que utilizaremos para desarrollar nuestro código.

Y para finalizar, crearé la base de datos utilizando el motor **MariaDB** integrado en el paquete **XAMPP**, compatible con MySQL. La conexión la realizaremos mediante el conector **JDBC**, para así poder almacenar los datos mediante nuestra interfaz.

1.1. Motivación

Hoy en día, el continuo avance tecnológico ha hecho que esta forme parte de nuestra vida, ya no solo a nivel cotidiano, si no a nivel laboral. Unit Control Military, esta pensada para llevar estos avances tecnológicos a la institución de la cual forme parte durante seis años de mi vida, el ejército.

La elección de este proyecto surge en primer lugar por el gran interés que sigo manteniendo por esta institución. Antes de abandonarla, ya hablaba con compañeros de empleo, y con los diferentes mandos que tenía en aquel entonces, sobre la elaboración de una interfaz que permitiera agilizar el trabajo en las oficinas de la compañía.

Durante los últimos años, muchas personas han tomado la decisión de acceder a esta institución, ya sea por vocación o por las salidas laborales que ofrece. Todas estas personas, finalmente llegarán a una unidad, donde serán encuadrados dentro de una compañía. Dentro de esta compañía cada persona deberá disponer de un número identificativo, al que se

conoce como **TIM** (**Tarjeta de identificación militar**), así como de un tipo de armamento, vehículo etc.…

Lo que se pretende con Unit Control Military, es agilizar procesos como pueden ser, realizar consultas, inserción de datos, actualización de estos, y su borrado, en el caso que sea necesario. Será mucho más fácil y eficiente realizar este tipo de tareas, permitiendo agilizar las diferentes tareas administrativas dentro de la una compañía, siendo estas tan importantes, como la instrucción del combatiente.

1.2. Objetivos propuestos

En este punto, vamos a hablar de los objetivos tanto generales, como específicos de nuestra aplicación.

Objetivos generales:

- Desarrollar una aplicación de escritorio para el manejo de la información dentro de una compañía militar.
- Agilizar los diferentes procesos como pueden ser la inserción, actualización, y borrados de datos, proporcionando una mayor eficiencia a la hora de realizar estos procesos.
- Realizar consultas que permitirá buscar datos de forma eficiente.
- Generar archivos Pdf con la información seleccionada.

Objetivos específicos:

- Usar el IDE **NetBeans** para el desarrollo del código
- Uso del lenguaje de programación **Java**, para el desarrollo de la aplicación.
- Integrar la biblioteca **Swing** para la creación de una interfaz intuitiva y amigable
- Crearé la base de datos utilizando el motor **MariaDB** integrado en el paquete XAMPP, compatible con MySQL.
- Realizar la conexión a la base de datos mediante el conector **JDBC**.
- Proporcionar una experiencia de usuario satisfactoria y atractiva.

1.3. Contexto laboral

Durante los 6 años que he pertenecido a esta institución, el ejército, me he visto en múltiples ocasiones en la tesitura de dar información referente a mi DNI, TIM (Tarjeta de identificación militar), numero de fusil etc.…

Esto es algo que se hace cuando el personal militar va a realizar maniobras, o vamos a desplegar en zona de operaciones. Por otro lado, me he visto en la obligación de revisar una y otra vez el número de fusil o la matricula del vehículo que tenía asignado. Esto es algo que hace perder mucho tiempo en el día a día del combatiente entorpeciendo de esta manera la instrucción.

Como ya he mencionado antes, durante los años que estuve en el ejército, comentaba con compañeros de empleo, y con los propios jefes, sobre la forma de conservar esta información, ya que normalmente se perdía. Esto lo note en mayor media a la hora de desplegar en zona de operaciones en el año 2020, en Letonia. El papeleo se hizo muy tedioso debido a que la información referente a los medios humanos y materiales se perdía, provocando que se recopilase de nuevo la información.. Esto de debido a que solo se usaba el formato papel. Esto son los principales motivos a nivel laboral que condicionarían la elección de este proyecto.

Por otro lado, se puede destacar las posibilidades a nivel laboral que puede ofrecer, algo. Por norma general, dentro del ejército, cada persona tiene una labor. Hay personas que dedican a tareas administrativas, al avituallamiento, al combate etc.…

Esta aplicación da la posibilidad de crear un equipo formando por integrantes de las diferentes compañías de las que dispone un batallón, destinado a la gestión de esta información. Esto permitirá que una compañía se coordine con otra para ofrecer los diferentes medios humanos y materiales, a la hora de realizar maniobras, ejercicios tácticos o despliegues en zonas de operaciones.

2. Metodología usada

De forma inicial, y antes de comenzar cualquier proyecto, debemos considerar la metodología que vamos a emplear para el desarrollo de este. Esto es importante, ya que tenemos a nuestra disposición diferentes tipos de metodologías, y la elección de una u otra, condicionará tanto para bien como para mal, el desarrollo de nuestro proyecto.

Estas metodologías podemos agruparlas en dos grandes tipos, las **metodologías tradicionales y las ágiles**. Antes de decidir definitivamente que metodología vamos a usar, hay que tener en cuenta las ventajas y desventajas que cada una nos puede ofrecerme para el desarrollo de este proyecto.

Por un lado, consideré el uso el uso de una **metodología tradicional**, como puede ser en cascada, espiral, incremental etc.…Este tipo de metodologías se caracterizan por tener una proyección lineal y sin marcha atrás. Con esto no esperamos cambios en el proyecto, ya que los requisitos se especifican una sola vez. (Demera, 2021)

Por otro lado, tenemos las **metodologías ágiles**, que surgen como una alternativa a las metodologías tradicionales. Este tipo de metodologías se caracterizan por ser adaptativas y flexibles. De esta forma, solventaríamos el problema que teníamos con las tradicionales, ya que no podíamos volver a una etapa anterior. Entre las mas usadas tenemos Kanban, Scrum y Lean. (Demera, 2021)

Finalmente, para el desarrollo de este proyecto me he decantado por utilizar una metodología tradicional en cascada con retroalimentación. Este modelo es una variante del modelo en cascada convencional, e incluye una realimentación entre las diferentes etapas del proyecto. De esta manera, será posible volver a una etapa anterior, e incluso varias etapas en cualquier momento para corregir errores, o realizar modificaciones.

La elección de esta metodología ha venido condicionada por el tipo de proyecto, y sobre todo por la entrega de este, ya que el cliente, que en este caso he considerado que es el tutor que tengo asignado para el proyecto, no ve el producto hasta que este esta terminado.

El ciclo de vida de este modelo pasa por diferentes etapas, tal y como podemos ver en la imagen insertada a continuación.

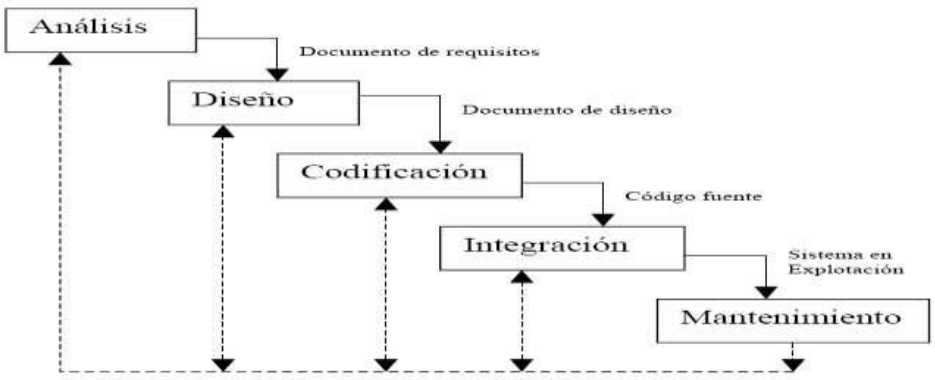

Figura 1 - Modelo en cascada con realimentación (Spuzi, s.f)

En cuanto a estas etapas, cabe destacar que existe un paso previo al análisis, que es la **planificación**. En la mayoría de los diagramas, no aparece la planificación como tal, ya que se considera un paso previo antes de iniciar el proyecto, pero en mi opinión es de las fases más importantes.

En la planificación determinaremos los objetivos, estimación de recursos y se definirá el plan de trabajo. En nuestro caso estableceremos los tiempos empleados, tiempos que representaremos mediante un diagrama de Gantt. En este diagrama se verá reflejado tanto la estimación inicial para el desarrollo del proyecto, como el tiempo real empleado. Esto lo desarrollaremos con mas detalladamente.

Una vez acabemos con la fase de planificación, pasaremos a la de **análisis**. Esta es la fase del proyecto donde determinaremos las necesidades y la finalidad de este. Aunque ya hemos hablado de ello en un punto anterior, en esta fase desarrollaremos de forma mas extensa y detallada los requisitos funcionales y no funcionales.

Además, realizaremos el modelado de datos mediante un diagrama Entidad-Relación, lo que nos permitirá representar de manera visual la estructura de estos, identificar las entidades y las relaciones.

También, dentro de esta fase, introduciremos el diagrama de casos de uso, que nos servirá para visualizar de manera clara las diferentes interacciones que los usuarios pueden tener con nuestra aplicación, algo que nos va a facilitar la compresión y el diseño del mismo.

Una vez terminada la fase de análisis, pasaríamos a la fase de **diseño**. En esta fase, nos centraremos en realizar un diseño previo a la implementación de nuestra aplicación. Definiremos como vamos a construir nuestra aplicación, para cumplir con los requisitos establecidos en la fase de análisis.

Otra de las particularidades de esta fase, es que podemos hacernos una idea de cómo será la interfaz de usuario, y que tipo de interacciones nos va a permitir realizar con la aplicación, así como el orden en que estas se producen. Todo esto lo realizaremos apoyándonos en ilustraciones, lo que nos proporcionará una visión mas clara de nuestra aplicación en esta fase del proyecto.

Llegados a este punto, realizaremos la **codificación.** En esta fase del proceso, realizaremos la escritura del código, basándonos en los diseños que hemos realizado en la fase anterior. Es decir, traduciremos las especificaciones y diseños en nuestro lenguaje de programación,

que como bien hemos dicho anteriormente, será Java, para que pueda ser interpretados y ejecutados por nuestro ordenador.

En esta fase, ya podemos aprovechar las ventajas que nos ofrece la metodología que hemos empleado, ya que, al utilizar un modelo en cascada con retroalimentación, nos va a permitir volver a una fase anterior para corregir los errores, en caso de que sea necesario.

Posteriormente, llegaremos a la fase de **integración**. En esta fase integraremos cada una de las partes de nuestra aplicación, y comprobaremos su funcionamiento, verificando así que cumple con todos los requisitos.

Por otro lado, también nos centraremos en corregir los errores que no hayamos localizado durante la fase anterior, la codificación, y le daremos solución, volviendo si es necesario, a una fase anterior. También obtendremos el ejecutable, que será entregado al cliente, que, en este caso será al tutor asignado al proyecto.

Para finalizar, llegamos a la última fase del ciclo de vida de nuestro proyecto, el **mantenimiento**. Durante esta fase, realizaremos mejoras destinadas al mantenimiento y mejora de nuestra aplicación, con el fin de garantizar su correcto funcionamiento. Esta fase la realizaremos desde el momento en que se acabe la aplicación, hasta la entrega final del proyecto.

Esto lo haremos durante ese periodo de tiempo, porque según la planificación que he llevado a cabo, se prevé acabar la aplicación una semana antes de la entrega, aunque esto se podrá ver mas detalladamente en los siguientes puntos, mediante el diagrama de Gantt.

3. Tecnologías y herramientas utilizadas en el proyecto

Git: Git es un sistema de control de versiones que utilizaremos para guardar un registro de las versiones de nuestro proyecto conforme vayamos avanzando y realizando cambios en el. Con esta herramienta podremos llevar a cabo un seguimiento de errores, y a su vez corregirlos. (B., 2023)

GitHub: GitHub es una plataforma, que sirve para dar alojamiento a nuestros proyectos, utilizando el sistema de control de versiones. Lo utilizaremos para guardar los cambios que vayamos realizando del código fuente de nuestra aplicación. (Solé, 2021)

GitHub Desktop: GitHub Desktop es un software que nos va a permitir actualizar nuestro proyecto desde su propia interfaz, subiendo a GitHub los cambios que vayamos realizando en él. (Calderon, 2020)

NetBeans: NetBeans es un entorno de desarrollo, que nos permitirá desarrollar aplicaciones nuestra aplicación, en nuestro vaso, usando el lenguaje de programación Java. (Aguilera, 2021)

Java: Java es un lenguaje de programación de alto nivel publicado en 1995. Su principal característica es que está orientado a objetos. (Coppola, 2023)

Hoy en día, Java es uno de los lenguajes de programación mas utilizados en el mercado. Actualmente se encuentra entre los 5 primeros lenguajes de programación, por debajo de C++, C y Python. (Tiobe, 2023)

Swing: Swing es una biblioteca perteneciente a Java, que nos permitirá crear interfaces gráficas de usuario. Esto lo haremos mediante el paquete javax.swing, donde encontraremos todos los componentes necesarios para construir nuestra interfaz. Antes de Swing, contábamos con la biblioteca AWT, la cual utilizaría los componentes nativos de cada sistema operativos para crear las interfaces. (Puig, 2022)

XAMPP: XAMPP es un paquete de software libre, el cual vamos a destacar por tener integrado el motor **MariaDB**, compatible con MySQL.". (Garcia, 2020)

JDBC: JDBC, también conocido como Java Database Connectivity, es una API que nos va a permitir conectarnos a nuestra base de datos. Usando esta tecnología, realizaremos la conexión a la base de datos de nuestro proyecto final, así como la ejecución de las diferentes operaciones. Esto lo haremos mediante el lenguaje de programación Java. (keepcoding, 2022)

OBS Studio: OBS Studio, son las siglas de Open Broadcaster Software, y es una aplicación que nos permitirá entre otras muchas funcionalidades realizar la grabación de nuestro video de defensa de nuestro proyecto final. (Pereira, 2021)

Word: Word es un software creado por Microsoft, y que se encuentra integrado el paquete Microsoft Office, y es utilizado hoy día para tratar textos. En nuestro caso, lo utilizaremos para la creación de la memoria de nuestro proyecto. (Alvarado, 2023)

Canva: Es un software, y un sitio web, que nos va a ofrecer, entre otras herramientas, una serie de plantillas para elaborar diagrama, imágenes, textos etc.… Además, nos va a permitir guardar nuestros proyectos para seguir editándolos en el futuro. En nuestro caso, el uso que le daremos será para realizar el diagrama de Gantt. (Allue, 2022)

Draw.io: Es un software multiplataforma utilizado para realizar diagramas de flujo, diagramas de red, etc.… En nuestro caso, lo vamos a utilizar para realizar el diagrama entidad relación. (KeepCoding Team, 2023)

Lucidchart: Podemos definirla como una plataforma online que nos permitirá crear diferentes tipos de diagramas. En nuestro caso, lo hemos utilizado para crear los diagramas de casos de uso. (Oyarzún, 2023)

4. Estimación de recursos y planificación

En este apartado haremos uso de un diagrama de Gantt, diseñado inicialmente y previo al desarrollo del proyecto, donde estará representado el tiempo inicial que se tendría en cuenta para llevar a cabo cada tarea de este proyecto. Al finalizar el proyecto, realizaremos otro diagrama de Gantt, con el tiempo real que nos han llevado dichas tareas, y veremos en que ha variado esta planificación

Diagrama de Gantt inicial

En este diagrama de Gantt hemos realizado una estimación del tiempo que nos llevará cumplir cada etapa del proyecto. Tal y como podemos ver en la imagen (Anexo 1) se tuvo en cuenta el tiempo que podría llevar la aprobación de la propuesta, y si esta era rechazada de manera inicial, tener margen para realizar modificaciones en caso de que sea necesario.

Por ello, decidí lanzar la propuesta entre el día 22 y 23 de Febrero, y tener ese margen de modificación hasta el día 28. En caso de que se aprobase antes, emplearía ese tiempo ganado en etapas posteriores del proyecto.

Una vez obtenido el apto, se ha tenido en cuenta el horario de clases, y las diferentes actividades pertenecientes a otras asignaturas, para elaborar una buena planificación a la hora de realizar el proyecto. De esta forma, definí una serie de objetivos, mediante fechas, las cuales podemos ver en este primer diagrama de Gantt, para tener margen de error y de mejora, en caso de necesidad.

La idea inicial es dedicar 5 horas diarias al desarrollo del proyecto, desde el momento en que obtuvimos al apto, hasta, en primer lugar, la fecha de entrega de feedback. En función de la respuesta obtenida en esa entrega, se subirán o se mantendrán las horas de trabajo, aunque esto es algo estimado y sujeto a posibles cambios.

Considero que esta es una de las entregas más importantes, ya que me permitirá realizar los ajustes necesarios, en función de la respuesta obtenida. Una vez obtenga la respuesta, la planificación continuará hasta la fecha de entrega ordinaria del proyecto, según lo establecido en el diagrama de Gantt que he realizado.

La parte mas importante del desarrollo de la aplicación, en mi caso, considero que es el código, ya que errores posteriores a la implementación de este, podría provocar que no se cumplan con las fechas de entrega, pero como hemos dicho antes, lo he planificado de tal forma, que tenga margen de error.

Diagrama de Gantt final

Como podemos ver en la imagen de nuestro diagrama final (Anexo 2), hemos tenido ligeros cambios, los cuales no han sido notorios, ya que la diferencia del diagrama estimado, al diagrama final, es de días.

Los cambios se han visto reflejados sobre todo en la implementación del código, ya que es algo que no podemos predecir, y encontrar soluciones a algunos errores, puede llevarnos más tiempo del esperado.

Es por eso por lo que todos estos errores, se han ido solucionando a medida que iban apareciendo. Al final, solo quedaban pequeños matices en la implementación del código, que se solucionaron después de realizar las pruebas finales para verificar que todo funcionaba correctamente.

Por otro lado, la memoria es lo que más tiempo me ha llevado, ya que se estaba trabajando en ella desde la aprobación de la propuesta, hasta el último día, completando este apartado. Además, me encontré con la problemática de las grandes dimensiones del diagrama de clases, las cual, me hizo perder bastante tiempo.

En algunos momentos, he aumentado las horas dedicadas a la elaboración del proyecto. En un primer momento estimé unas 5 horas diarias de trabajo, pero en algunos momentos las llegué a aumentar hasta las 10 horas. Esto es debido a motivos académicos, ya que había entregas de otras asignaturas, que coincidían, por ejemplo, con la fecha de entrega de feedback.

Por último, la grabación y elaboración del powerpoint, me ha llevado más tiempo del esperado, ya que quería hacer una buena exposición. La creación de diapositivas se ha realizado cuidadosamente, ya que era el soporte en el cual iba apoyar mi defensa, al menos hasta mostrar la aplicación.

5. Análisis del proyecto

Para comenzar este apartado, hablaremos de una de las partes más importantes en el proceso de desarrollo de nuestro proyecto, y de cualquier software. Hablamos de los requisitos funcionales y no funcionales.

Requisitos funcionales

Los requisitos funcionales nos darán información sobre cómo se va a comportar nuestra aplicación en determinadas situaciones. (Ilerna Online S.L, 1.ª Edición: Enero de 2021) Estos requisitos los tenemos a continuación.

- **RF1**: **Iniciar sesión** en la aplicación. La aplicación nos pedirá un usuario y una contraseña creados previamente, con los que accederemos a las funcionalidades que esta nos va a ofrecer.
- **RF2**: **Gestión de personal**. Podremos insertar, actualizar y eliminar información perteneciente a los medios humanos, como puede ser la TIM (Tarjeta de identificación militar), DNI, armamento asignado etc.….
- **RF4: Gestión armamentística.** La aplicación nos va a permitir insertar, actualizar y eliminar información perteneciente al armamento, como puede ser el número del arma, persona a la que esta asignada, el estado etc.…
- **RF5**: **Gestión vehicular**. La aplicación permitirá la inserción, actualización y eliminación de la información referentes a los medios automovilísticos. De esta manera podremos manipular la información vehicular, como puede ser las matriculas de los vehículos, el estado, el personal al que esta asignado ese vehículos entre otro tipo de información.
- **RF6**: **Gestión de transmisiones**. Vamos a poder manipular información referente las transmisiones de las que se dispongan, como puede ser el número, el estado, vehículo al que pertenece etc... Realizando la inserción, la actualización y la eliminación de esta estos.
- **RF7**: **Realizar consultas**. Hacer este tipo de acciones es una de las principales funcionalidades que nos va a permitir esta aplicación.
- **RF8**: **Generar PDF**. Podremos generar archivos PDF que contenta la información relacionada de una tabla.

Requisitos no funcionales

Los requisitos no funcionales nos ayudarán a comprender las limitaciones en cuanto a la funcionalidad se refiere de nuestra aplicación. (Ilerna Online S.L, 1.ª Edición: Enero de 2021) A continuación, veremos cuales tendríamos.

- **RNF1**: La aplicación será segura y protegerá la información mediante técnicas de autenticación.
- **RNF2**: La aplicación debe ser capaz de manejar una gran cantidad de datos.
- **RNF3**: La aplicación será fácil de usar e intuitiva, con una interfaz de usuario clara y amigable.
- **RNF4**: La aplicación debe ser compatible con diferentes sistemas operativos.
- **RNF5**: La aplicación será podrá ser usada en cualquier sistema, sin necesidad de disponer de un equipo potente.
-

Al terminar de definir los requisitos funcionales y no funcionales, pasaremos a realizar el diagrama entidad relación, el cual analizaremos para explicar cada una de las relaciones y las tablas creadas.

Diagrama entidad-relación.

Tal y como podemos ver en la imagen de nuestro diagrama (Anexo 3) he identificado 4 entidades. Estas entidades serían personal, vehículos, armamento y transmisiones.

Una vez tenemos claras cuales son las entidades, debemos pasar a definir los atributos de cada una, algo muy importante, porque podremos identificar cual será la clave primaria de cada entidad, pero esto es algo que analizaremos mas adelante y de forma más detallada. Por otro lado, cabe destacar, que estas entidades, son entidades fuertes. De momento vamos a identificar los atributos de estas cuatro entidades:

- **Personal**: TIM, nombre, apellido1, apellido2, DNI, rango y teléfonos. De estas entidades, hay que tener especial cuidado con teléfonos, ya que es un atributo multivaluado, y estos atributos, generan tablas. Esto lo veremos más adelante cuando especifiquemos que tablas salen de cada relación.
- **Vehículos**: Matricula, nombre, tipo, kilómetros y estado.
- **Armamento**: Número, nombre, tipo, calibre y estado.
- **Transmisiones**: Numero, nombre, tipo y estado.

Cuando ya tenemos definidas nuestras entidades con sus atributos, hay que dar un paso el cual considero muy importante y hay que tener muy claro, y es el de definir cuál será la clave primaria, o la **"primary key".**

Comenzando por la entidad personal, me ha parecido correcto utilizar el atributo TIM (Tarjeta de identificación militar) como clave primaria. En un principio barajé introducir un id, o incluso usar DNI, como clave primaria, pero dada la naturaleza de esta aplicación, decidí utilizar el atributo TIM.

Como bien sabemos, la clave primaria, o primary key, debe identificar de manera única cada fila de una tabla, y debe ser un valor único, es decir, que no se repita. Es por este motivo por el cual he decidido usar el atributo TIM como la clave primaria de esta entidad, ya que todo personal militar, dispone de una tarjeta de identificación, la cual es única e irrepetible en toda la institución militar.

Pasando a analizar las claves primarias de las 3 entidades restantes, tenemos algo parecido a la tabla personal. En la tabla vehículos, he utilizado la matrícula de estos como primary key, ya que también es única e irrepetible y solo puede haber un vehículo en todo el ejército español, con esa numeración.

Respecto al armamento y a las transmisiones, son medios los cuales están identificados por un número, y esta es la forma por la cual se sabe que medio pertenece a que persona dentro de esta institución. Por tanto, he utilizado el número identificativo de cada entidad como clave primaria, ya que cumplen con los requisitos para poder serlo.

Ya con las primary key identificadas, vamos a pasar a realizar las relaciones entre las diferentes entidades:

- **Relación personal y armamento**. En esta relación vamos a obtener una cardinalidad **1:N.** Esto se debe a que una persona puede tener asignado una o varias armas, y un arma puede pertenecer a una persona o a ninguna. En el primer caso se explica por que hay personal que puede tener asignado pistola y fusil, o pistola y ametralladora, o simplemente fusil.

  En el siguiente caso nos encontramos con que un arma, solo puede estar asignada a una persona, pero también puede no estar asignada, ya que, por norma general, se dispone de material "de fondillo", en caso de que el armamento de alguien quede inoperativo. Este armamento igualmente esta inventariado y es este el motivo por el cual lo hemos tenido en cuenta.

- **Relación personal vehículos**: En este caso, la cardinalidad que vamos a obtener es **N:M.** Esto se debe a que una persona puede no tener asignado ningún vehículo, como puede ser el caso de una sección de reconocimiento que se dedica a reconocer el terreno a pie, o varios vehículos, ya que por ejemplo en el caso de los carros de combate, una tripulación puede tener asignado dos carros, para tener siempre una solución a la posible inoperatividad de alguno de estos.

  En el caso contrario, un vehículo puede estar asignado a una sola persona, como es el caso de los vehículos ligeros, a los cuales les puede realizar el mantenimiento una sola persona, o varias personas, ya que tenemos casos, como es el de los carros de combate, ya mencionados antes, que están formados por una tripulación de cuatro personas.
  Algo para tener en cuenta, es que con las cardinalidades **N:M**, debemos crear una nueva tabla, pero esto lo detallaremos mas adelante para dejar bien estructuradas las tablas que crearemos.

- **Relación personal transmisiones:** Aquí tenemos un caso como el que habíamos visto en la relación de personal armamento, una cardinalidad **1:N**. Una persona puede tener asignado un equipo de transmisiones, como puede ser un walkie talkie, o varios equipos de transmisiones, ya que además del walkie talkie, puede tener asignado una radio portátil, que se utiliza para establecer contacto con la zona donde esta el mando de operaciones.

  Por otro lado, las transmisiones pueden estar asignadas a una persona, y solo a una, o a ninguna persona, como es el equipo que se tiene inventariado para recambios, en caso de inoperatividad de los equipos a los que se les está dando uso.

Modelo relacional

Para explicar las tablas que se van a generar a partir de estas cardinalidades, haré uso del modelo relacional ya que con él se entenderá mejor, y podremos explicar de una manera mas detallada, porque se han generado esas tablas.

En total vamos a tener 7 tablas, y todas excepto una, se relacionarán entre ellas. La tabla que no se relacionará es la tabla Usuario.

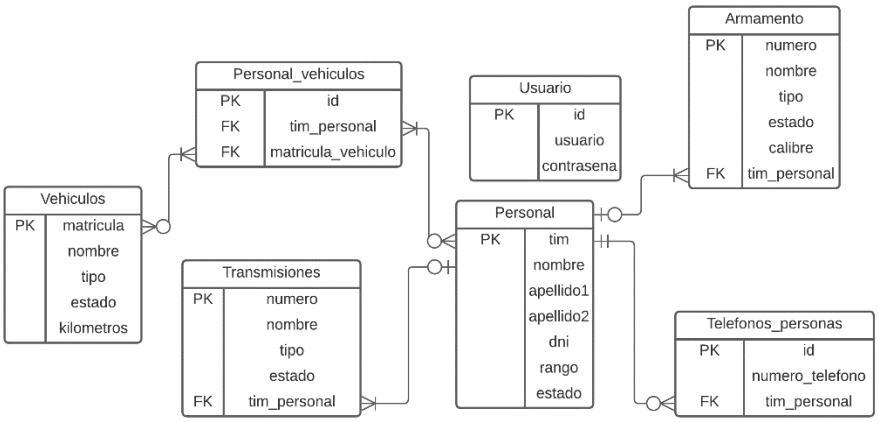

Figura 4 – Modelo relacional. Fuente: Elaboración propia en Lucidchart.com

De todas las relaciones que hemos desarrollado anteriormente, cabe destacar la relación entre vehículos y personal. Esta relación, como ya hemos dicho antes, es **N:M**, y como en todas las relaciones **N:M**, va a generar una tabla intermedia.

Esta tabla intermedia se relaciona con la tabla personal, y vehículo, mediante una **FK**, que está conectada a la **PK**, de cada tabla, creando de esta forma la relación.

En el resto de las tablas, como por ejemplo la tabla armamento, y transmisiones, se relacionan con la tabla personal, mediante un atributo, que tendrá la función de **FK**, y se encargará de relacionarse con dicha tabla mediante su **PK**.

Por otro lado, cabe destacar la tabla teléfono_persona, con una relación **1:N.** Esta tabla surge de la propia tabla personal, ya que el atributo teléfono, es multivaluado, y va a generar una nueva tabla.

En esta nueva tabla, he identificado como primary key el id de la tabla teléfono, y como **FK** tim_personal, que hará referencia a al atributo TIM, de la tabla personal.

Por último, tenemos la tabla usuarios que simplemente la he creado para tener un registro de los usuarios que hay registrados en la aplicación. En nuestro caso al tener un único usuario, dada las características de nuestra aplicación, no realizaremos interacciones con esta tabla.

La única interacción que se realizará, la hará la propia aplicación, para verificar que el usuario y la contraseña introducidos, coincide con el registro que tenemos en la tabla.

Diagrama de casos de uso.

Una vez hemos realizado la explicación de las tablas finales que vamos a tener en nuestra base de datos, vamos a realizar el **diagrama de casos de uso**, con el que representaremos el comportamiento de muestra aplicación.

Estos diagramas los tenemos en la zona de anexos, ya que, debido a sus dimensiones, y por normativa, no podemos verlos de forma de correcta en esta sección. (Anexo 4), (Anexo 5), (Anexo 6), (Anexo 7), (Anexo 8).

En estos diagramas tenemos a un actor, que será el usuario que itera con la aplicación, y que estará representado por un monigote. Por otro lado, también tenemos la representación de lo que es nuestro sistema, y dentro de él, los distintos casos de uso representando por un ovalo.

Estos casos de uso se encuentran unidos por líneas, que representan la asociación. Tenemos asociaciones "**include**", que representarían que por ejemplo un caso de uno "**B**", incorporaría el comportamiento de "**A**".

Por otro lado, tenemos los casos de uso "**extends**", que significaría que el comportamiento del caso de uso "**A**" podría extenderse a "**B**", pero esto si se cumplen unas condiciones.

Una vez hemos realizado nuestros diagramas, explicaremos esto mediante las **tablas de especificación.**

En nuestro caso, al tener varios diagramas que son idénticos, donde lo único que cambia es la tabla con la que estamos realizando interacciones mediante la interfaz, realizaremos nuestras tablas de especificación para una sola ventana de la interfaz.

Se ha tomado esta decisión, porque realizarlos todas estas tablas, sobrepasaría el límite de 16 páginas establecidos para este punto. Igualmente, estas tablas de especificación son totalmente validas para los demás casos.

De esta forma, la interfaz seleccionada es la de Personal, ya que en ella nos encontramos con funcionalidades como inserción, modificación, y eliminación de datos, además, de mostrar registros.

Antes de nada, comenzaremos con el caso de uso Login, ya que será el primer paso para acceder a nuestra aplicación y comenzar a iterar con ella

Caso de uso	Login	Numero	1
Actores	Actor no logueado		
Descripción	El actor debe ser capaz de iniciar sesión		
Precondición			
Secuencia normal		Acción	
	Paso	Actor	Sistema
	1	Accede al Login	
	2	Introduce los datos de inicio de sesión	
	3		El sistema comprueba los datos

Caso de uso	Comprobar Login	Numero	2
Actores	Actor no logueado		
Descripción	El sistema comprueba los datos introducidos		
Precondición	Introducir datos para ser comprobados		
Secuencia normal		Acción	
	Paso	Actor	Sistema
	1		Comprueba los datos
	2		Permite el acceso a las funcionalidades de la aplicación.

Caso de uso	Acceso al menú	Numero	3
Actores	Actor logueado		
Descripción	El actor debe poder acceder al menú principal		
Precondición	El actor debe estar logueado para poder acceder al menú		
Secuencia normal		Acción	
	Paso	Actor	Sistema
	1	Accede al Login	
	2	Introduce los datos de inicio de sesión	
	3		El sistema comprueba los datos
	4		Permite el acceso al menú principal

Caso de uso	Acceso al menú personal	Numero	4
Actores	Actor logueado		
Descripción	El actor debe poder acceder a al ventana personal		
Precondición	Pulsar el botón personal del menú principal		
Secuencia normal	Paso	Acción	
		Actor	Sistema
	1	Pulsa el botón personal	
	2		Permite el acceso a la ventana personal

Como ya hemos dicho antes, llegados a este punto, realizaremos las tablas de especificación de los casos de usos de uno de nuestros diagramas, ya que esto sería igual en las siguientes tablas de especificaciones.

Caso de uso	Elegir tabla y crear personal, personal_vehiculos o teléfonos	Numero	5
Actores	Actor logueado		
Descripción	El actor debe elegir si insertar personal, personal_vehiculos o teléfono		
Precondición			
Secuencia normal	Paso	Acción	
		Actor	Sistema
	1	Pulsa el botón correspondiente a la tabla que se quiere acceder	
	2		Permite el acceso la tabla seleccionada

Caso de uso	Insertar datos	Numero	6
Actores	Actor logueado		
Descripción	El actor debe insertar datos		
Precondición	Elegir donde insertar datos (Personal, personal_vehiculos, teléfonos)		
Secuencia normal	Paso	Acción	
		Actor	Sistema
	1	Rellena los datos correspondientes	
	2	Pulsa el botón insertar	
	3		Comprueba los datos
	4		Inserta los datos

	5		Informa que se han insertado los datos
Casos alternativos		Acción	
	Paso	Actor	Sistema
	1a	Rellena los datos	
	2a	Pulsa el botón insertar	
	4a		Informa que el formato de los datos son incorrectos
	4b	Revisa los datos	
	4c	Reenvía los datos	
	4d		Vuelta al paso 3

Caso de uso	Seleccionar registro	Numero	7
Actores	Actor logueado		
Descripción	El actor debe seleccionar un registro		
Precondición			
Secuencia normal		Acción	
	Paso	Actor	Sistema
	1	Pulsar sobre un registro	
	2		Selecciona el registro sobre el que se ha pulsado

Caso de uso	Eliminar datos	Numero	8
Actores	Actor logueado		
Descripción	El actor debe eliminar datos		
Precondición	Seleccionar un registros		
Secuencia normal		Acción	
	Paso	Actor	Sistema
	1	Pulsa sobre un registro	
	2		Selecciona el registro
	3	Pulsar el botón eliminar	Comprueba los datos
	4		Hace pregunta de seguridad
	5	Pulsa “Yes”	
	6		Elimina el registro
Casos alternativos		Acción	
	Paso	Actor	Sistema
	1a	Pulsa sobre un registro	

	2a		Selecciona el registro
	3a	Pulsa el botón eliminar	
	4a		Hace pregunta de seguridad
	5a	Pulsa “No”	
	5b		No elimina el registro

Caso de uso	Editar registros	Numero	9
Actores	Actor logueado		
Descripción	El actor debe poder modificar registros		
Precondición	Seleccionar un registro		
Secuencia normal	Paso	Acción	
		Actor	Sistema
	1	Pulsa sobre un registro	
	2		Selecciona el registro
	3	Edita los parámetros	
	4	Pulsa sobre el botón “Modificar”	
	5		Comprueba los datos
	6		Edita el registro
Casos alternativos	Paso	Acción	
		Actor	Sistema
	1a	Pulsa sobre un registro	
	2a		Selecciona el registro
	3a	Edita los parámetros	
	4a	Pulsa sobre el botón “Modificar”	
	5a		Comprueba los datos
	5b		Informa que el formato los datos son incorrectos
	5c	Revisa los parámetros	
	5d	Edita de nuevo los parámetros	
	5e		Vuelta al paso 5

Como bien se ha dicho antes, estas tablas de especificaciones se extrapolan a las ventanas pertenecientes a las otras tablas. En nuestro caso, solo quedaría realizar las tablas de especificaciones de la ventana consultas, que se realizarán a continuación.

Caso de uso	Acceso al menú	Numero	10
Actores	Actor logueado		
Descripción	El actor debe poder acceder al menú principal		
Precondición	El actor debe estar logueado para poder acceder al menú		
Secuencia normal	Paso	Acción	
		Actor	Sistema
	1	Accede al Login	
	2	Introduce los datos de inicio de sesión	
	3		El sistema comprueba los datos
	4		Permite el acceso al menú principal

Caso de uso	Acceso a consultas	Numero	11
Actores	Actor logueado		
Descripción	El actor debe poder acceder a al ventana consultas		
Precondición	Pulsar el botón personal del menú principal		
Secuencia normal	Paso	Acción	
		Actor	Sistema
	1	Pulsa el botón consultas	
	2		Permite el acceso a la ventana consultas

Caso de uso	Insertar datos consulta	Numero	12
Actores	Actor logueado		
Descripción	El actor debe insertar datos de la consulta		
Precondición	Elegir donde realizar la consulta (Personal, armamento, vehículos, transmisiones o teléfonos,)		
Secuencia normal	Paso	Acción	
		Actor	Sistema
	1	Rellena los datos correspondientes	
	2	Pulsa el botón consulta	
	3		Comprueba los datos
	4		Muestra los datos de la consulta en la tabla
Casos alternativos	Paso	Acción	
		Actor	Sistema

	1a	Rellena los datos	
	2a	Pulsa el botón consulta	
	4a		No encuentra datos de la consulta
	4b	Revisa los datos	
	4c	Reenvía los datos	
	4d		Vuelta al paso 3

Caso de uso	Generar pdf de la consulta	Numero	13
Actores	Actor logueado		
Descripción	El actor debe generar un pdf de la consulta		
Precondición	Elegir donde realizar la consulta datos (Personal, armamento, vehículos, transmisiones o teléfonos,)		
Secuencia normal		Acción	
	Paso	Actor	Sistema
	1	Pulsa sobre el botón generar pdf	
	2		Comprueba los datos
	3		Crea y abre un archivo pdf, con los datos de la consulta
Casos alternativos		Acción	
	Paso	Actor	Sistema
	1a	Pulsa sobre el botón generar pdf	
	2a		Comprueba los datos
	3a		No genera el archivo pdf
	4b	Revisa los datos	
	3c	Vuelve a pulsar el botón generar pdf	
	3d		Vuelta al paso 2

Una vez finalizadas las tablas de especificaciones pertenecientes al diagrama de casos de uso, llegamos al siguiente punto, el diseño del proyecto.

6. Diseño del proyecto

Ya en la fase de planificación, se realizó un boceto previo del proyecto en una hoja de papel para tener bien estructurados al menos, en un primer momento, los JFrame con los que se iba a trabajar, tal y como podemos ver en la imagen.

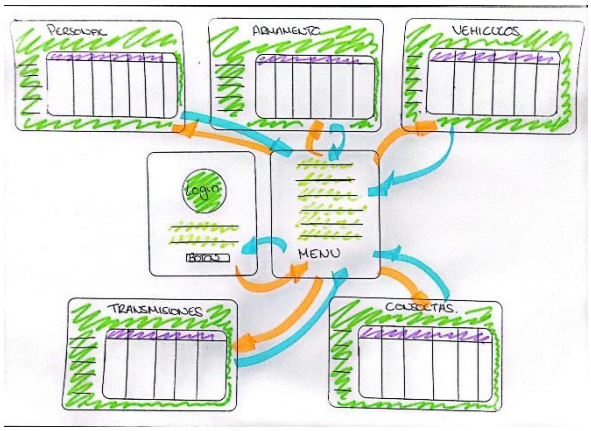

Figura 10 – Boceto prototipo. Fuente: Elaboración propia, a mano y en formato papel

Esto nos facilitará la parte del diseño a la hora de implementar nuestra aplicación, ya que agilizaremos el proceso de colocación de todos los componentes en nuestro **JFrame**. Por otro lado, también podemos ver la navegación entre los distintos **JFrames**.

Se escogió esta forma de realizar el primer boceto, o diseño de la aplicación por su sencillez y la información con relación a las interacciones entre JFrames que nos iba aportar. Mas tarde, este boceto se ha llevado a nuestro código, donde sufrió algunos cambios, no importantes, pero que si ayudaría a mejorar la facilidad en el uso de nuestra aplicación.

Antes de comenzar con la creación de la interfaz, creamos una serie de paquetes con el fin de modular nuestra aplicación, y al mismo tiempo tener nuestro código lo más ordenado posible. Los paquetes creados han sido **Interfaces, Clases, ReportesPDF, Img** y el paquete con el método main, que este ultimo se añadió por defecto al crear nuestro proyecto.

En el paquete **Interfaces**, todo lo relacionado con la interfaz de la aplicación, en el cual hemos creado diferentes JFrames, que hacen referencia a las que teníamos en el boceto inicial.

Por otro lado en el paquete **Clases**, tenemos las diferentes clases con las que vamos añadir funcionalidad a la aplicación. Tenemos clases identificadas como “nombreclase + Class”, donde tenemos los constructores, y los métodos setters y getters.

Además, dentro de este mismo paquete también tenemos diferentes clases, que hace referencia a las entidades previamente identificadas en puntos anteriores, como pueden ser

personal, armamento etc.… donde tenemos los diferentes métodos creados para iterar con nuestra base de datos.

Ya para finalizar, tenemos el paquete **ReportesPDF**, donde tenemos la plantilla del pdf que podremos generar utilizando la aplicación.

Esto lo desarrollaremos en este mismo punto de forma mas extensa, ya que considero interesante hablar sobre el proceso de diseño e implementación de nuestra aplicación, y como la hemos modularizado.

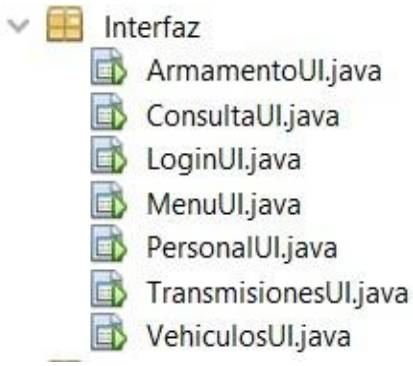

Figura 11 – Paquete Interfaz. Fuente: Captura de pantalla en NetBeans

En primer lugar, vamos a comenzar hablando del paquete **interfaz**, ya que ha sido el primer paquete en ser creado, junto con sus JFrames correspondientes. Como podemos ver en la imagen, tenemos sietes JFrames creados.

De estos siete JFrames, 4 de ellos se corresponden a las entidades que tenemos identificadas en nuestro diagrama entidad- relación. Las tres restantes se corresponden con la interfaz de Login, menú y Consultas.

Cabe destacar que estos JFrames tienen añadido a su nombre “UI”, que significaría user interface, que traducido al castellano sería interfaz de usuario. Esto se hizo para tener identificados los JFrames, cuando se estuviera trabajando con mas de una clase.

Ya introduciéndonos en el desarrollo, en primer lugar se creo la interfaz principal que sería el Login, y a la cual hemos llamado **LoginUI**. A través de esta interfaz, accederemos a nuestra aplicación, y a las funcionalidades que esta nos va a ofrecer.

En esta primera ventana, o ventana de Login, tenemos un diseño bastante sencillo. En el lado derecho tenemos un **JLabel**, con el nombre de nuestra aplicación

Además de esto, se han usado dos tonos de verdes para terminar de desarrollar la interfaz. En base a estos colores se continuo con el desarrollo de las siguientes interfaces.

Además, tenemos un JLabel correspondiente al usuario junto con su **JTextField**, y otro JLabel correspondiente a la contraseña, con su **JPasswordField**. Estos serán los campos para rellenar para que el sistema nos de acceso al menú principal.

Seguidamente, se realizó la interfaz del menú, llamada **MenuUi**. Esta será la primera pantalla que veremos al iniciar nuestra aplicación, y ella nos encontramos con los botones, que nos llevarán a los diferentes JFrames de la aplicación. Además, hemos añadido un botón que nos permitirá cerrar sesión, el cual nos devolverá a la pantalla del Login. Respecto al diseño, sigue el mismo estilo que teníamos en la ventana de Login.

En el menú personal, llamado **PersonalUI**, tenemos un JLabel en la zona central superior con el nombre de la ventana en la que nos encontramos.

Ya en la zona media tenemos un **JTabbedPane**, en el cual tenemos varias pestañas correspondientes a las tablas que están relacionadas con esta entidad. Dentro de cada pestaña nos encontramos con los JTextField que podremos rellenar, identificados con el nombre del campo correspondiente.

Una vez tengamos estos campos rellenos, podremos realizar las operaciones que creamos oportunas, dándonos la posibilidad de insertar, modificar y eliminar registros.

Esto lo realizaremos con sus los botones correspondientes, que hemos creado con JButtons. Además, tenemos otro botón, llamado “registros”, que nos permitirá mostrar en nuestra tabla, todos los registros que tengamos introducidos en nuestra base de datos.

Por otro lado, hay que destacar que tenemos varios botones en la parte superior del **JTabbedPane**, que nos permitirán navegar entre las diferentes interfaces de nuestra aplicación. También, he añadido en la parte inferior del panel dos JLabel a modo, donde se especifica el formato de los campos DNI y TIM.

Las siguientes interfaces **ArmamentoUI**, **TransmisionesUI**, y **VehiculoUI**, tienen un diseño similar al comentado anteriormente, con la única diferencia que el JTabbedPane, no tiene pestañas. En el centro, tenemos otro **JLabel** más con el nombre de la interfaz en la que nos encontramos.

Tal y como ocurre en la ventana PersonalUI, en la parte inferior tenemos una tabla con varios que podremos rellenar, identificados por su **JLabel** correspondiente. Además, nos encontramos con los botones que nos van a permitir insertar, eliminar, modificar y mostrar los registros.

Ya para finalizar con nuestras interfaces tenemos la ventana consultas, llamada **ConsultasUI**. En cuando al diseño, es el mismo que el ya mencionado antes, al menos en lo que se refiere al nombre de la ventana en la que nos encontramos, situado en la zona central de esta.

En la zona media, tenemos un JTabbedPane tal y como ocurre en la ventana PersonalUI, donde tenemos varias pestañas para realizar nuestras consultas.

La primera pestaña que tenemos dentro de esta interfaz es la pestaña personal. En ella podemos realizar consultas filtrando por tres tipos de datos que son **nombre, TIM y estado**. El dato que considero mas interesante es el de estado, ya que este nos permitirá conocer por ejemplo, cuantas personas se encuentran de alta, baja, de permiso etc…

Además, tenemos un botón para generar Pdf, el cual explicaremos con más detalle cuando lleguemos al paquete **ReportesPDF**.

En las siguientes pestañas tenemos armamento, vehículos y transmisiones. Las tres pestañas tienen la misma disposición. Tienen dos tablas y en cada una podemos filtrar por dos tipos de datos diferentes, ya que me pareció más cómodo a la hora de realizar consultas. Estos datos son TIM, nombre, numero y estado.

Como pasaba en la pestaña de personal, considero que el dato más importante por el que vamos a filtrar es estado, ya que nos permitirá conocer que material se encuentra operativo, inoperativo, en mantenimiento etc.…

También disponemos de un botón para generar Pdf, pero este botón solo se aplica a la segunda tabla, ya que es la que mas información va a aportarnos, sobre todo, y como ya hemos mencionado antes, filtrando por el campo estado.

Para finalizar con el apartado de nuestras interfaces, pasaremos hablar de la última pestaña, la pestaña teléfonos, donde filtraremos tanto por TIM, como por nombre. Esta pestaña es exactamente igual a la pestaña personal, con la única diferencia en el botón para generar Pdf. En este caso no lo tenemos, ya que esta consulta que se realizaran en casos puntuales

Una vez hemos acabado con nuestras interfaces, y con el diseño de estas, pasaremos a nuestro siguiente paquete de aplicación, el cual he llamado “**Clases**”. Dentro de este paquete, tenemos un total de 15 clases.

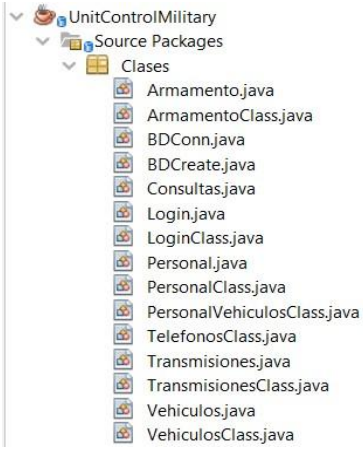

Figura 12 – Paquete Clases. Fuente: Captura de pantalla en NetBeans

Dentro de este paquete tenemos las clases destinadas a crear los constructores de las entidades, juntos con sus métodos setters y getters correspondientes. Estas clases las identificamos porque se les ha añadido por tener “**Class**” en su nombre.

Entre las diferentes clases tenemos “**BDCreate**”. Esta clase es la más importante de todas, ya que es la encargada de crear nuestra base de datos, llamada “**Unit Control Military**”. Esto lo haremos con el usuario por defecto “**root**”, que tiene los privilegios necesarios para realizar la operación, y con la sentencia “**CREATE DATABASE IF NOT EXISTS**”.

De esta forma nos aseguramos de que la base de datos se cree tan solo una vez. También crearemos un usuario, con los suficientes privilegios para poder iterar adecuadamente con la base de datos.

Además, crearemos las diferentes tablas que conforman la base de datos, y con las que iteraremos desde nuestra interfaz. Estas tablas son la tabla **personal, armamento, vehículos, transmisiones, personal_vehiculos y teléfonos**. En este caso hemos usado “**CREATE TABLE IF NOT EXISTS”,** para asegurarnos de que tan solo se creen una vez.

También, crearemos la tabla usuario, junto con su usuario “**Admin**” y su contraseña “**1234**”, que serán las credenciales que tendremos que introducir desde nuestra interfaz principal de **LoginUI**, para poder acceder a nuestra aplicación.

Todo esto lo tenemos estructurado dentro de la clase, donde hemos separado estas operaciones por métodos. Tenemos un método encargado de crear la base de datos, otro de crear la tabla usuario junto con su contraseña, un método para crear el usuario que usaremos para iterar posteriormente con la base de datos, y por último, un método para crear las

diferentes tablas. Cabe destacar, que todo esto lo realizamos gracias a nuestro conector **JDBC**, ya que sin el, no sería posible realizar la conexión, ni crear la base de datos.

Otra particularidad, es que tanto la base de datos, como el usuario, y las diferentes tablas, se crearán al iniciarse nuestra interfaz **LoginUI**. Esto lo haremos llamando a los métodos correspondientes de la clase **BDCreate**.

Una vez hemos acabado con la clase BDCreate, pasaremos a la clase **BDConn**. La clase **BDConn**, ha sido creada para que se itere con la aplicación desde el usuario que hemos creado previamente en la clase **BDCreate**.

De esta forma, dejamos de utilizar el usuario root, y utilizamos el usuario Admin, que es el nombre que le hemos dado, junto con su contraseña. Esta es la clase que usará para realizar las conexiones con la base de datos, y realizar la inserción, modificación y eliminación de registros en las diferentes tablas.

Una vez tenemos hecha toda la configuración de la base de datos, pasaremos a crear nuestra clase **Login**, y **LoginClass**. En la clase LoginClass, tenemos dos métodos constructores, uno vacío y otro con parámetros.

Estos parámetros son id, usuario y contraseña, de los cuales, estos dos últimos hacen referencia al usuario y la contraseña que debemos introducir en la interfaz **LoginUI**. Además de los métodos constructores, esta clase tiene sus correspondientes métodos setter y getter.

Esta clase solo contiene lo ya mencionado antes, así que pasaremos a comentar clase **Login**. Aqui tenemos un solo método, en el cual realizaremos una consulta a la base de datos, a la tabla usuario, más concretamente, para verificar que los datos introducidos en nuestra interfaz coincidan con los de la tabla.

Esto lo realizaremos desde LoginUI, desde donde llamaremos al método correspondiente de la clase Login. La consulta a realizaremos utilizando un **SELECT**.

Una vez tenemos configurado el acceso a nuestra aplicación, pasamos a implementar las funcionalidades de la aplicación en las siguientes interfaces.

Vamos a comenzar con los métodos PersonalClass, PersonalVehiculosClass y TelefonoClass. Estas clases, al igual que todas las clases que terminan en "**Class**", tienen tanto sus propios constructores, como sus métodos setter y getter.

El motivo por el cual estoy hablando de ellas en el mismo punto, es porque todas van a implementar sus métodos en la clase Personal. Esto lo decidí así, porque dentro de la interfaz **PersonalUI**, tenemos una pestaña correspondiente a cada una de estas clases.

En la clase **Personal**, tenemos los métodos con los cuales realizaremos las principales operaciones con nuestra base de datos. Entre esos métodos, tenemos los insert. Estos insert los hemos implementado con **sentencias SQL,** más en concreto con "**INSERTS**" , los cuales hemos ejecutamos posteriormente usando la clase **PreparedStatement**. Seguidamente, ejecutaremos el insert con el método **execute**.

Antes de ejecutar la sentencia, hemos llamado a nuestra clase **BDConn** para realizar la conexión a la base de datos, y que, como hemos dicho antes, la usaríamos para realizar todas las operaciones con nuestro usuario principal.

Ya para finalizar, llamaremos al método en la interfaz **PersonalUI**, donde hemos creado otro método, con el que introduciremos los valores en nuestra base de datos. Esto lo haremos desde los **JTextField** de la interfaz. También hemos utilizado condicionales **"if"** para verificar que los campos no estén vacíos y que además, estos tenga el formato correcto.

Para realizar las **modificaciones** de los registros, hemos hecho lo mismo que en los insert. La única diferencia la encontramos en la sentencia SQL, que se ha usado "**UPDATE**", vez de "**INSERT**", pero el resto, incluyendo los condicionales, es exactamente igual que en el método anterior.

Pasando al botón **eliminar**, lo ejecutamos también de la misma forma, utilizando en este caso "**DELETE**", donde, además, realizamos una pregunta de seguridad, antes de eliminar el registro. De esta forma, podemos evitar eliminar por error datos importantes.

En el caso de mostrar los registros, lo que realizamos es una **consulta**. Con una sentencia SQL donde utilizamos un "**SELECT**", obtenemos los registros de nuestra base de datos y los mostramos en la tabla de la interfaz.

En nuestro caso hemos configurado cada botón, para que la lista de registros se actualice automáticamente con cada inserción, modificación o eliminación de registros.

Una vez hemos acabado con las clases pertenecientes a la tabla personal, pasamos a implementar las pertenecientes al resto de tablas que son **armamento, vehículos transmisiones.**

Estas tres clases están implementadas de la misma forma, con la única diferencia que, en transmisiones y armamento, tenemos el campo "**TIM**" que hace referencia a la tabla personal. Como hemos hecho anteriormente, hemos creado la clase **TransmisionesClass**, **ArmamentoClass**, y **VehículosClass**.

Dentro de estas clases, tenemos encapsuladas las propiedades de cada una de las tablas, así como sus métodos **constructores** sin parámetros, y sus métodos constructores con parámetros. Además, tenemos los métodos **setter y getter** en cada una de ellas.

Después desde las clases **Transmisiones, Vehículos y Armamento,** se implementarían los métodos para realizar los inserción, modificación y eliminación de datos. Al igual que hicimos en la clase personal, se utilizarán sentencias SQL, una conexión a la base de datos etc.…

En el caso de las tablas **Armamento y Transmisiones** contamos con un campo numérico que hace referencia al número de arma o de transmisión, el cual también tiene un formato específico. Respecto a los registros, tenemos nuevamente un botón para que los muestre en la tabla de nuestra interfaz al ser presionado.

Con la clase **vehículos** pasa exactamente igual que en Transmisiones y Armamento, con la diferencia que en esta no tenemos campo TIM. Esto es debido a que ya tenemos nuestra tabla intermedia **Personal_vehiculos**, la cual hemos mencionado antes, y con la que podremos asignar vehículos al personal. En cuanto a sus métodos, todos están implementados de la misma igual que en las clases mencionadas anteriormente.

Dentro de estas clases, al igual que en las anteriores, tenemos los métodos para insertar, modificar, eliminar y mostrar registros, los cuales se han implementados de forma independiente. Esto ha ayudado a tener el código mejor organizado y resolver problemas de forma eficiente.

Por último, pasamos a la clase **Consultas**, que ha sido la última en implementarse dentro de nuestra aplicación. Para implementar esta clase hemos tenido en cuenta varias cosas. Por un lado, las consultas a realizar, que en nuestro caso se realizan a la tabla Personal, Armamento, Vehículos, Transmisiones y Teléfonos.

En el caso de **Personal**, hemos creado una tabla en la cual se mostrarán los registros al presionar el botón consulta. Además, hemos filtrado por tres tipos de datos como son Nombre, Tim y Estado.

La consulta la hemos ejecutado utilizando **PreparedStatement**, que como ya hemos dicho antes, es una clase utilizada para enviar consultas, y la hemos ejecutado con el método **ExecuteQuery**, el cual nos devuelve un objeto **Resulset**. Resulset nos permitirá acceder a los datos de las consultas, los cuales serán devueltos.

En el caso de las consultas de **teléfonos**, tenemos también una sola tabla donde mostraremos las consultas, tal y como ocurría con personal. En este caso filtramos por dos tipos de datos como son Nombre y TIM,

En, cuanto, a **armamento, transmisiones y vehículos**, tenemos dos tablas en cada pestaña para mostrar los registros. Resulta más interesante a la hora de manejar la información o para comparar algún tipo de dato.

En la tabla superior tenemos consultas simples, como son nombre y TIM, que nos devolverán muy pocos datos. En la inferior tenemos consultas por el número del arma, transmisión o vehículo, y por el estado del material, que nos aportará mas información.

Es este el motivo por el cual he decidido que el botón "Generar Pdf", tan solo obtenga los datos de las consultas realizadas en la tabla inferior, ya que nos devolverá más datos y más información de las tablas. Con esto, ya daríamos por finalizado el paquete "Clases".

El último paquete que se creo fue el de "**ReportesPDF**", en el cual tenemos tan solo una clase llamada **PlantillaPDF**. Esta clase es la que utilizaremos para generar nuestro archivo PDF, cuando pulsemos en el botón correspondiente de nuestra interfaz.

Para implementar nuestro PDF, hemos hecho uso de la librea **iText**, la cual hemos importado a nuestro proyecto. Esta librearía nos permitirá configurar de forma precisa nuestro documento, que en nuestro caso es un documento Pdf, permitiéndonos crear tablas, cambiar el tipo de letra, añadir líneas de texto, etc.…

La implementación se ha hecho con diferentes métodos, ya que cada método corresponde a cada una de las tablas a las cuales vamos a realizar las consultas. Respecto al diseño de esta plantilla, he optado por un diseño muy sencillo, ya que considero más importante obtener la información correspondiente a la consulta que estamos realizando, a que este documento sea muy vistoso.

Es por eso por lo que he decidido introducir simplemente la fecha, que estaría situada en la parte superior derecha y la cual se obtendría directamente de la hora actual del sistema, y dos títulos, de los cuales uno correspondería al título del informe, y el otro a la tabla en la que hemos hecho la consulta.

Para obtener la información de la tabla, he optado por obtener los datos directamente del **JTable** de nuestra interfaz. Es decir, en primer lugar, haríamos la consulta, y una vez obtengamos los datos de la consulta en la tabla de nuestra interfaz, los recataremos para introducirlos en el pdf que vamos a generar.

La tabla del nuestro Pdf la he creado con la clase **PdfTable**, la cual pertenece a la librería la mencionada antes, **iText**. Para obtener el nombre de las columnas, he recorrido con un bloque "**for**", las columnas de la tabla. De la misma forma se ha realizado para obtener los registros, solo que en este caso hemos utilizado dos bucles "**for**" anidados.

Cabe destacar, que el Pdf generado, es un archivo temporal, el cual he creado usado el método **createTempfile** de la clase **File**. Decidí hacerlo de esta manera para solventar el problema que iba a causar la ubicación donde quedaría guardado el archivo.

Siendo consciente de ello, opte por esta opción, ya que una vez generado el archivo, el usuario podría guardarlo en una ubicación de sus sistema, o directamente imprimirlo.

Por otro lado, y buscando solventar posibles problemas a la hora de ejecutar la aplicación en otros ordenadores, el archivo Pdf generado, se abrirá con la aplicación que tengamos predeterminada en nuestro sistema, para la lectura de este tipo de archivos.

Esto lo he conseguido usando el método **getDesktop** de la propia clase **Desktop**. De esta forma, me aseguro de que el usuario, no tenga problemas a la hora de usar la aplicación y pueda acceder, a todas sus funcionalidades.

Ya finalizando con el paquete “ReportesPDF”, tenemos que mencionar el paquete **Img**, en el cual he introducido la imagen que hemos utilizado para el diseño de la interfaz.

Para finalizar, tenemos el paquete “**unitcontrolmilitary**”, donde tenemos nuestro método main. Dentro de este método, he creado una instancia de nuestra interfaz de **Login**, ya que será la primera ventana que veremos al iniciar nuestra aplicación. Una vez hecho esto, ya tenemos nuestra aplicación totalmente funcional y lista para crear el archivo ejecutable.

7. Despliegue y pruebas

A la hora de realizar las diferentes pruebas de nuestra aplicación, estas, se han ido realizando a medida que se iba avanzando en su implementación. Esto nos ha ayudado a la hora de identificar errores, y así poder solventarlos inmediatamente.

Por otro lado, se ha estado consultando la base de datos de forma continua, para asegurar que la inserción de datos, la modificación y la eliminación de estos, se realizaba de forma correcta al tratar de hacerlo usando la interfaz de la aplicación.

Llegados a este punto, detallaremos las pruebas realizadas.

Prueba numero 1: Conexión con la base de datos
Objetivos probados: Conexión con la base de datos mediante XAMPP.

Requisitos probados: Comprobamos que la conexión con la base de datos sea correcta cuando iniciamos la aplicación.
Pruebas que realizar: Iniciar la aplicación y comprobar que se crean automáticamente la base de datos, las diferentes tablas de la base de datos y el usuario que vamos a utilizar para interactuar con ella.

Prueba numero 2: Usuario o contraseña incorrecto.
Objetivos probados: Acceso denegado con credenciales incorrectos.
Requisitos probados: Comprobamos que no se puede acceder a la aplicación si no se aportar el usuario con la contraseña correcta.
Pruebas que realizar: Intentar acceder a la aplicación con un usuario y contraseña incorrecto.

Prueba numero 3: Login de usuario.
Objetivos probados: Acceso permitidos con el usuario y contraseña correctos.
Requisitos probados: Comprobamos que podemos acceder a las funcionalidades de la aplicación aportando el usuario y la contraseña correctos.
Pruebas que realizar: Intentar acceder a la aplicación con el usuario y contraseña correctos.

Prueba numero 4: Desplazamientos a través del menú.
Objetivos probados: Acceso a las diferentes ventanas de la aplicación mediante el menú.
Requisitos probados: Comprobamos que todos los botones del menú principal nos llevan a las ventanas correspondientes, según el botón pulsado.
Pruebas que realizar: Intentar desplazarnos por las diferentes interfaces, presionando sobre los botones del menú.

Prueba numero 4: Insertar datos personal.
Objetivos probados: Insertar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos insertar datos en la base de datos rellenando los campos disponibles de la ventana personal, y que estos se insertan al pulsar el botón insertar.
Pruebas que realizar: Intentar insertar datos en nuestra base de datos a través de la interfaz.

Prueba numero 5: Modificar datos personal.
Objetivos probados: Modificar datos desde la interfaz en nuestra base de datos.

Requisitos probados: Comprobamos que podemos modificar datos en la base de datos modificando los campos disponibles de la ventana personal, y que estos se modifican al pulsar el botón modificar.
Pruebas que realizar: Intentar modificar datos en nuestra base de datos a través de la interfaz.

Prueba numero 6: Eliminar datos personal.
Objetivos probados: Eliminar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos eliminar datos en la base de datos, y que estos se eliminan al pulsar el botón eliminar.
Pruebas que realizar: Intentar eliminar datos en nuestra base de datos a través de la interfaz.

Prueba numero 7: Insertar datos armamento.
Objetivos probados: Insertar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos insertar datos en la base de datos rellenando los campos disponibles de la ventana armamento, y que estos se insertan al pulsar el botón insertar.
Pruebas que realizar: Intentar insertar datos en nuestra base de datos a través de la interfaz.

Prueba numero 8: Modificar datos armamento.
Objetivos probados: Modificar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos modificar datos en la base de datos modificando los campos disponibles de la ventana armamento, y que estos se modifican al pulsar el botón modificar.
Pruebas que realizar: Intentar modificar datos en nuestra base de datos a través de la interfaz.

Prueba numero 9: Eliminar datos armamento.
Objetivos probados: Eliminar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos eliminar datos en la base de datos, y que estos se eliminan al pulsar el botón eliminar.
Pruebas que realizar: Intentar eliminar datos en nuestra base de datos a través de la interfaz.

Prueba numero 10: Insertar datos vehículo

Objetivos probados: Insertar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos insertar datos en la base de datos rellenando los campos disponibles de la ventana vehículo, y que estos se insertan al pulsar el botón insertar.
Pruebas que realizar: Intentar insertar datos en nuestra base de datos a través de la interfaz.

Prueba numero 11: Modificar datos vehículo.
Objetivos probados: Modificar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos modificar datos en la base de datos modificando los campos disponibles de la ventana vehículo, y que estos se modifican al pulsar el botón modificar.
Pruebas que realizar: Intentar modificar datos en nuestra base de datos a través de la interfaz.

Prueba numero 12: Eliminar datos vehículo.
Objetivos probados: Eliminar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos eliminar datos en la base de datos, y que estos se eliminan al pulsar el botón eliminar.
Pruebas que realizar: Intentar eliminar datos en nuestra base de datos a través de la interfaz.

Prueba numero 13: Insertar datos transmisiones.
Objetivos probados: Insertar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos insertar datos en la base de datos rellenando los campos disponibles de la ventana transmisiones, y que estos se insertan al pulsar el botón insertar.
Pruebas que realizar: Intentar insertar datos en nuestra base de datos a través de la interfaz.

Prueba numero 14: Modificar datos transmisiones
Objetivos probados: Modificar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos modificar datos en la base de datos modificando los campos disponibles de la ventana transmisiones, y que estos se modifican al pulsar el botón modificar.
Pruebas que realizar: Intentar modificar datos en nuestra base de datos a través de la interfaz.

Prueba numero 15: Eliminar datos transmisiones.
Objetivos probados: Eliminar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos eliminar datos en la base de datos, y que estos se eliminan al pulsar el botón eliminar.
Pruebas que realizar: Intentar eliminar datos en nuestra base de datos a través de la interfaz.

Prueba numero 16: Insertar datos teléfono.
Objetivos probados: Insertar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos insertar datos en la base de datos rellenando los campos disponibles de la ventana teléfono, y que estos se insertan al pulsar el botón insertar.
Pruebas que realizar: Intentar insertar datos en nuestra base de datos a través de la interfaz.

Prueba numero 17: Modificar datos teléfono
Objetivos probados: Modificar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos modificar datos en la base de datos modificando los campos disponibles de la ventana teléfono, y que estos se modifican al pulsar el botón modificar.
Pruebas que realizar: Intentar modificar datos en nuestra base de datos a través de la interfaz.

Prueba numero 18: Eliminar datos teléfono.
Objetivos probados: Eliminar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos eliminar datos en la base de datos, y que estos se eliminan al pulsar el botón eliminar.
Pruebas que realizar: Intentar eliminar datos en nuestra base de datos a través de la interfaz.

Prueba numero 19: Insertar datos personal_vehiculos.
Objetivos probados: Insertar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos insertar datos en la base de datos rellenando los campos disponibles de la ventana personal_vehiculos, y que estos se insertan al pulsar el botón insertar.
Pruebas que realizar: Intentar insertar datos en nuestra base de datos a través de la interfaz.

Prueba numero 20: Modificar datos personal_vehiculos.
Objetivos probados: Modificar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos modificar datos en la base de datos modificando los campos disponibles de la ventana personal_vehiculos, y que estos se modifican al pulsar el botón modificar.
Pruebas que realizar: Intentar modificar datos en nuestra base de datos a través de la interfaz.

Prueba numero 21: Eliminar datos en la tabla personal_vehiculos.
Objetivos probados: Eliminar datos desde la interfaz en nuestra base de datos.
Requisitos probados: Comprobamos que podemos eliminar datos en la base de datos, y que estos se eliminan al pulsar el botón eliminar.
Pruebas que realizar: Intentar eliminar datos en nuestra base de datos a través de la interfaz.

Prueba numero 22: Consultar datos en la tabla personal
Objetivos probados: Consultar datos en la base de datos
Requisitos probados: Comprobamos que podemos realizar consultas filtrando por tipo de dato, y que esta consulta se muestra en la tabla de la interfaz.
Pruebas que realizar: Intentar realizar consultas a la base de datos mediante la interfaz, pulsando sobre el botón consulta.

Prueba numero 23: Generar pdf personal.
Objetivos probados: Generar archivo pdf.
Requisitos probados: Comprobamos que podemos generar un archivo pdf con la consulta que hemos realizado previamente.
Pruebas que realizar: Intentar generar un archivo pdf pulsando sobre el botón “generar pdf”.

Prueba numero 22: Consultar datos en la tabla armamento
Objetivos probados: Consultar datos en la base de datos
Requisitos probados: Comprobamos que podemos realizar consultas filtrando por tipo de dato, y que esta consulta se muestra en la tabla de la interfaz .
Pruebas que realizar: Intentar realizar consultas a la base de datos mediante la interfaz, pulsando sobre el botón consulta.

Prueba numero 23: Generar pdf armamento.

Objetivos probados: Generar archivo pdf.
Requisitos probados: Comprobamos que podemos generar un archivo pdf con la consulta que hemos realizado previamente.
Pruebas que realizar: Intentar generar un archivo pdf pulsando sobre el botón “generar pdf”.

Prueba numero 22: Consultar datos en la tabla vehículos
Objetivos probados: Consultar datos en la base de datos
Requisitos probados: Comprobamos que podemos realizar consultas filtrando por tipo de dato, y que esta consulta se muestra en la tabla de la interfaz .
Pruebas que realizar: Intentar realizar consultas a la base de datos mediante la interfaz, pulsando sobre el botón consulta.

Prueba numero 23: Generar pdf vehículos.
Objetivos probados: Generar archivo pdf.
Requisitos probados: Comprobamos que podemos generar un archivo pdf con la consulta que hemos realizado previamente.
Pruebas que realizar: Intentar generar un archivo pdf pulsando sobre el botón “generar pdf”.

Prueba numero 22: Consultar datos en la tabla transmisiones
Objetivos probados: Consultar datos en la base de datos
Requisitos probados: Comprobamos que podemos realizar consultas filtrando por tipo de dato, y que esta consulta se muestra en la tabla de la interfaz .
Pruebas que realizar: Intentar realizar consultas a la base de datos mediante la interfaz, pulsando sobre el botón consulta.

Prueba numero 23: Generar pdf transmisiones.
Objetivos probados: Generar archivo pdf.
Requisitos probados: Comprobamos que podemos generar un archivo pdf con la consulta que hemos realizado previamente.
Pruebas que realizar: Intentar generar un archivo pdf pulsando sobre el botón “generar pdf”.

Prueba numero 22: Consultar datos en la tabla teléfono
Objetivos probados: Consultar datos en la base de datos

Requisitos probados: Comprobamos que podemos realizar consultas filtrando por tipo de dato, y que esta consulta se muestra en la tabla de la interfaz .
Pruebas que realizar: Intentar realizar consultas a la base de datos mediante la interfaz, pulsando sobre el botón consulta.

8. Conclusiones

La implementación de este proyecto ha supuesto un reto, en lo que se refiere alcanzar los objetivos fijados. Por un lado, me preocupaba la implementación de la base de datos, ya que en un primer momento no sabía cómo realizarla para que se crease de forma automática al iniciar la aplicación.

Mi idea era que el usuario no tuviera que hacer nada, simplemente iniciar la aplicación, introducir las credenciales, y que pudiese iterar con la base de datos mediante la interfaz. Finalmente, todo esto fue posible, gracias a los conocimientos adquiridos durante el curso, y a la ampliación de estos realizando cursos en otras plataformas, y consultando documentación.

Por otro lado, el botón generar Pdf supuso otro reto, ya que mi idea era manejar los distintos escenarios en los que se pudiese utilizar la aplicación. En primer lugar, me encontré con la problemática de guardar el archivo generado, algo que descarté y que solucioné finalmente creando un archivo temporal. Y por último me encontré con el problema de que este archivo tenía que abrirse automáticamente, pero con el software que el usuario tuviera por defecto en su equipo.

Ya de forma general, he tenido la oportunidad de aplicar todos los conocimientos adquiridos en las distintas asignaturas que se han impartido durante el grado superior, asentarlos, y al mismo tiempo ampliarlos, gracias a la búsqueda de información en plataformas online dedicadas al campo de la programación

Un aspecto positivo que puedo destacar respecto al proyecto es la mejora de confianza. Al comienzo del proyecto, no sabía cómo afrontarlo, y llegue a sentir miedo, ya que me enfrentaba a algo totalmente nuevo y desconocido.

Una vez comencé su desarrollo, me di cuenta de que simplemente había que unir todos los conocimientos adquiridos, e ir paso a paso. Otro punto a favor es la realización de esta memoria, ya que me ha ayudado a llevar una organización en la realización del proyecto, y cumplir cada una de las tareas, antes de avanzar a la siguiente.

Resulta muy satisfactorio ver funcionar algo que has creado desde cero, y a lo que les has dedicado tanto tiempo, y sin duda, este es el principal motivo por el que me gusta este mundo. La satisfacción que da conseguir todos tus objetivos, y solventar todos los problemas que van apareciendo.

9. Bibliografía/Webgrafía

Aguilera, A. (16 de Junio de 2021). *¿Qué es NetBeans? – Entorno de desarrollo*. Obtenido de tecno-simple: https://tecno-simple.com/que-es-netbeans-entorno-de-desarrollo/

Allue, J. B. (11 de Noviembre de 2022). *CANVA, una herramienta de diseño web gratuita*. Obtenido de Palbin: https://www.palbin.com/es/blog/p858-canva-una-herramienta-de-diseno-web-gratuita.html

Alvarado, M. (7 de Febrero de 2023). *¿Qué es Word y para qué sirve? Características y funciones*. Obtenido de Internetastic: https://www.internetastic.com/word/

B., G. (10 de Enero de 2023). *Qué es GitHub y cómo empezar a usarlo*. Obtenido de www.hostinger.es: https://www.hostinger.es/tutoriales/que-es-github

Calderon, I. (16 de Diciembre de 2020). *Cómo manejar tus repositorios con GitHub Desktop*. Obtenido de www.medium.com: https://medium.com/streamelopers/c%C3%B3mo-manejar-tus-repositorios-con-github-desktop-8ba9fa85e31d

Coppola, M. (13 de Febrero de 2023). *Qué es Java, para qué sirve, características e historia*. Obtenido de blog.hubspot.es: https://blog.hubspot.es/website/que-es-java

Demera, R. (2 de Febrero de 2021). *Metodologías… ¿tradicional vs ágil?* Obtenido de Tribalyte Technologies: https://tech.tribalyte.eu/blog-metodologias-tradicional-vs-agil

FreePik. (s.f.). *Placa icono gratis*. Obtenido de Flaticon: https://www.flaticon.es/icono-gratis/placa_2369514?term=insignia+militar&page=1&position=6&origin=search&related_id=2369514

Garcia, M. (30 de Mayo de 2020). *¿QUE ES XAMPP Y COMO PUEDO USARLO?* Obtenido de nettix: https://www.nettix.com.pe/blog/web-blog/que-es-xampp-y-como-puedo-usarlo/

Ilerna Online S.L. (1.ª Edición: Enero de 2021). *Entornos de Desarrollo.*

keepcoding. (6 de Mayo de 2022). *¿Qué es JDBC (Java Database Connectivity)?* Obtenido de keepcoding: https://keepcoding.io/blog/jdbc-java-database-connectivity/

KeepCoding Team. (4 de Enero de 2023). *¿Qué es Draw.io?* Obtenido de KeepCoding: https://keepcoding.io/blog/que-es-drawio/

Oyarzún, G. (14 de Marzo de 2023). *Qué es Lucidchart y por qué facilita la gestión de proyectos*. Obtenido de comparasoftware: https://blog.comparasoftware.com/lucidchart/

Pereira, M. (9 de Diciembre de 2021). *Cómo usar OBS para hacer una transmisión en directo*. Obtenido de hotmart: https://hotmart.com/es/blog/como-usar-obs

Puig, L. E. (1 de Agosto de 2022). *¿Qué es la biblioteca Swing?* Obtenido de aluracursos: https://www.aluracursos.com/blog/biblioteca-swing

Robledano, A. (24 de Septiembre de 2019). *Qué es MySQL: Características y ventajas*. Obtenido de openwebinars: https://openwebinars.net/blog/que-es-mysql/

Solé, R. (17 de Septiembre de 2021). *GitHub, el repositorio de código abierto más importante del mundo*. Obtenido de profesionalreview: https://www.profesionalreview.com/2021/09/17/que-es-github/

Spuzi. (s.f). *El ciclo de vida de los Sistemas de información. Modelos de Ciclo de Vida.* Obtenido de GitHub: Recuperado el día 6 de Marzo de 2023 de https://spuzi.github.io/Spuzipedia/076ModelosCicloVida/076ModelosCicloVida.html

Tiobe. (Febrero de 2023). *Índice TIOBE de febrero de 2023*. Obtenido de Tiobe: https://www.tiobe.com/tiobe-index/

10. Anexos

Anexo 1

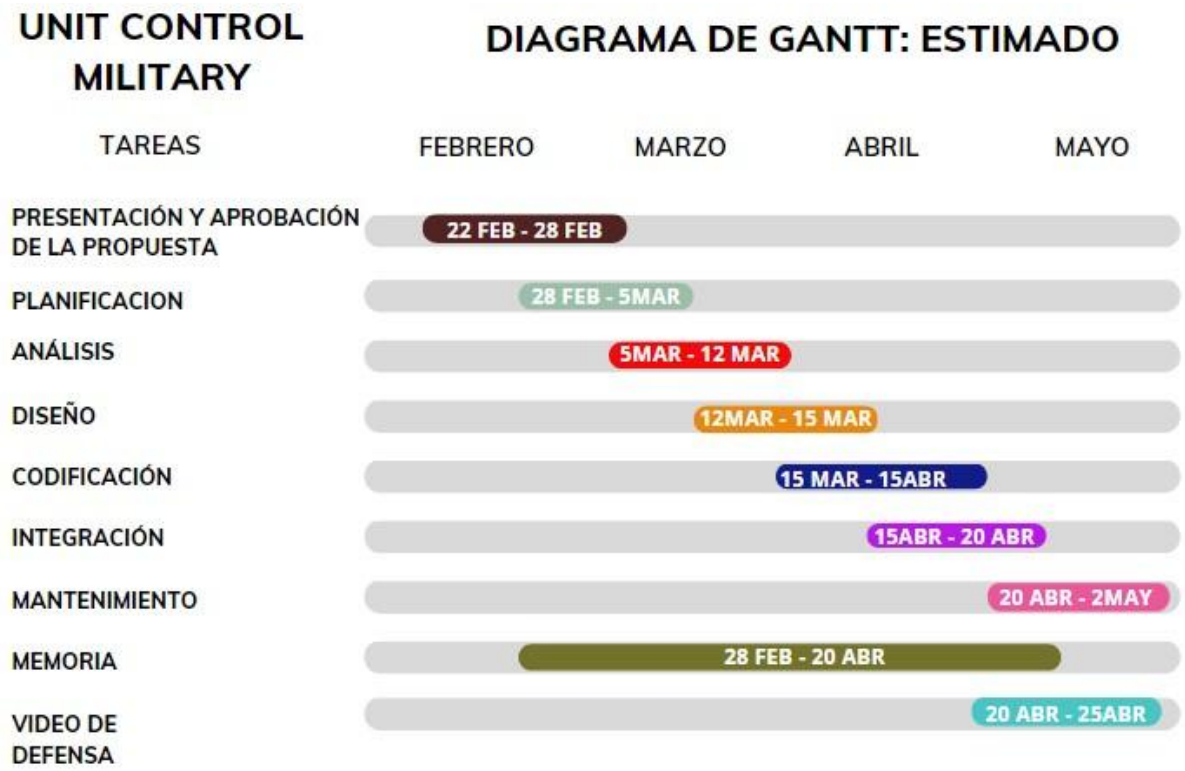

Figura 1 - Diagrama de Gantt: Estimado. Fuente: Elaboración propia en canvas.com

Anexo 2

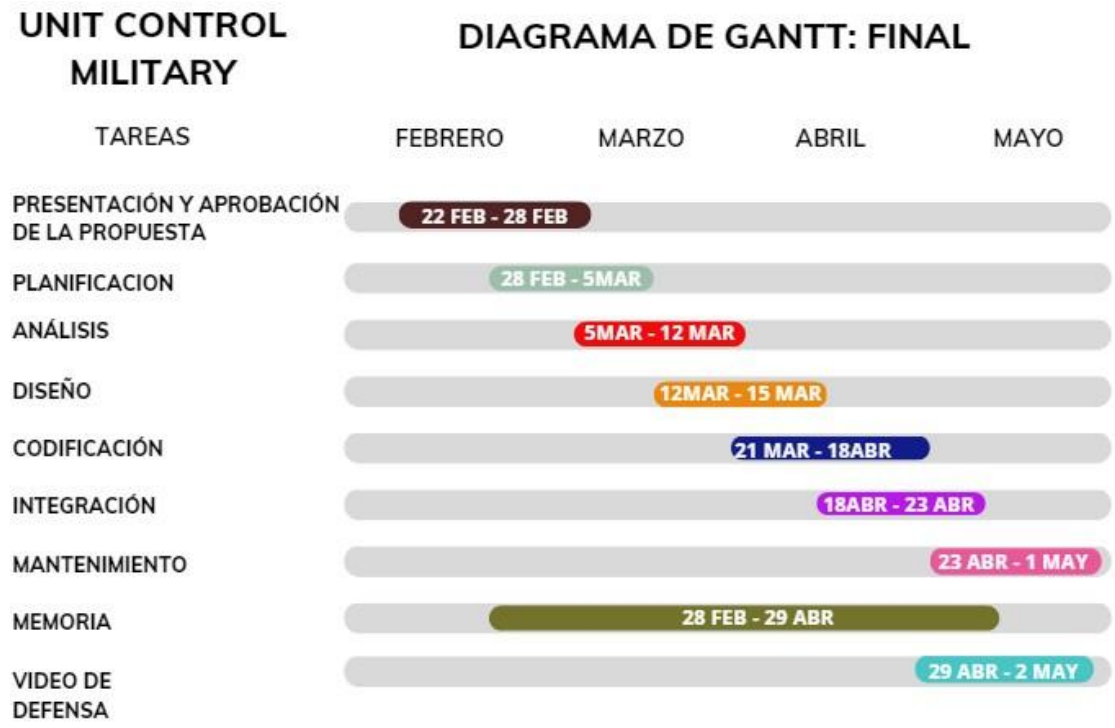

Figura 2 - Diagrama de Gantt: Final. Fuente: Elaboración propia en canvas.com

Anexo 3

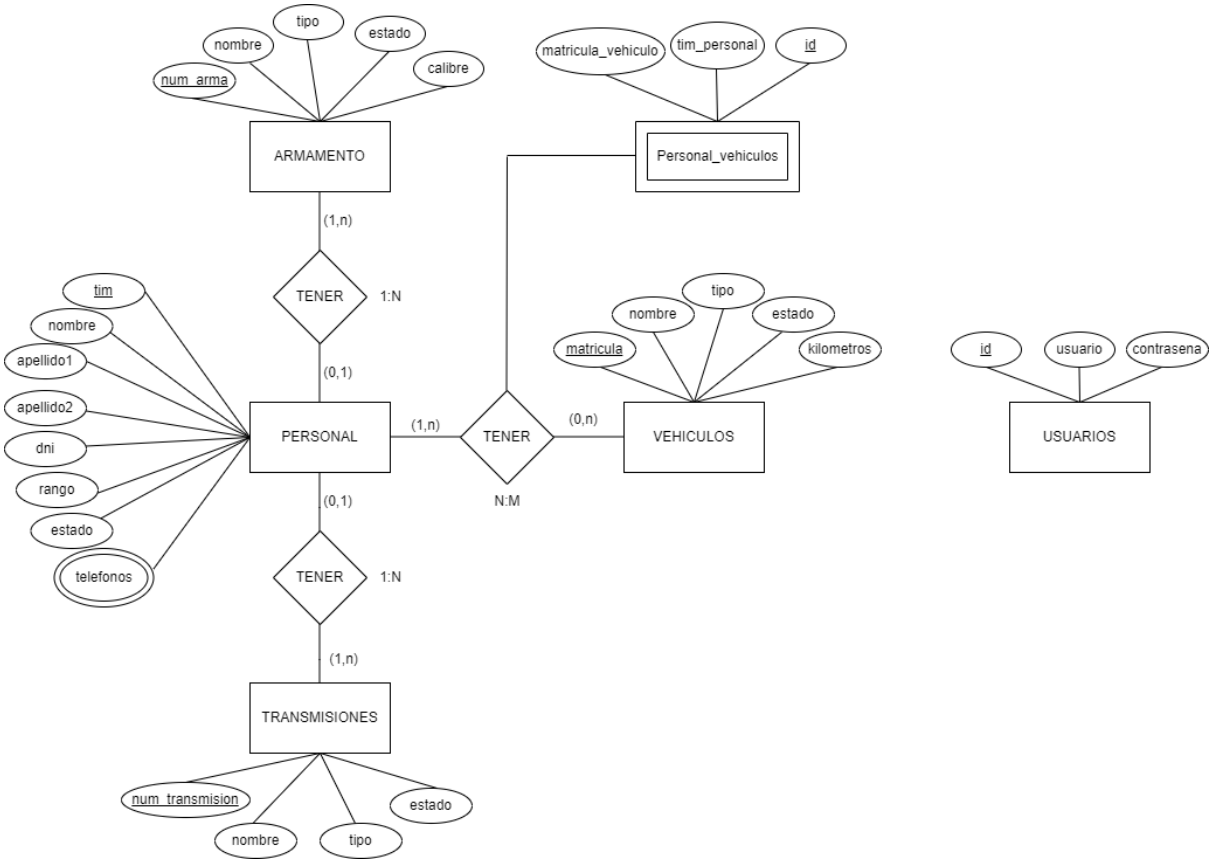

Figura 3 - Diagrama entidad-relación: Fuente: Elaboración propia en Draw.io

Anexo 4

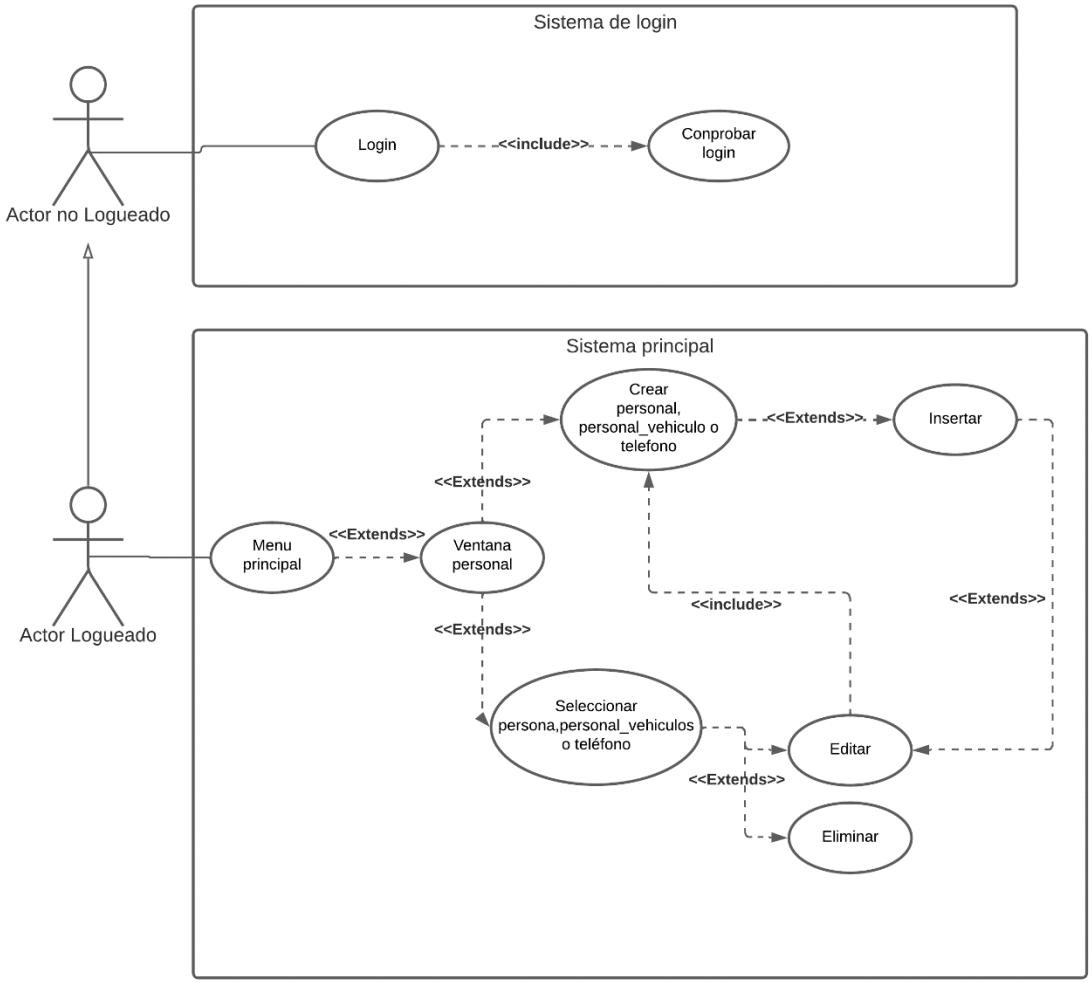

Figura 5 – Diagrama de casos de uso. Fuente: Elaboración propia en LucidChart.com

Anexo 5

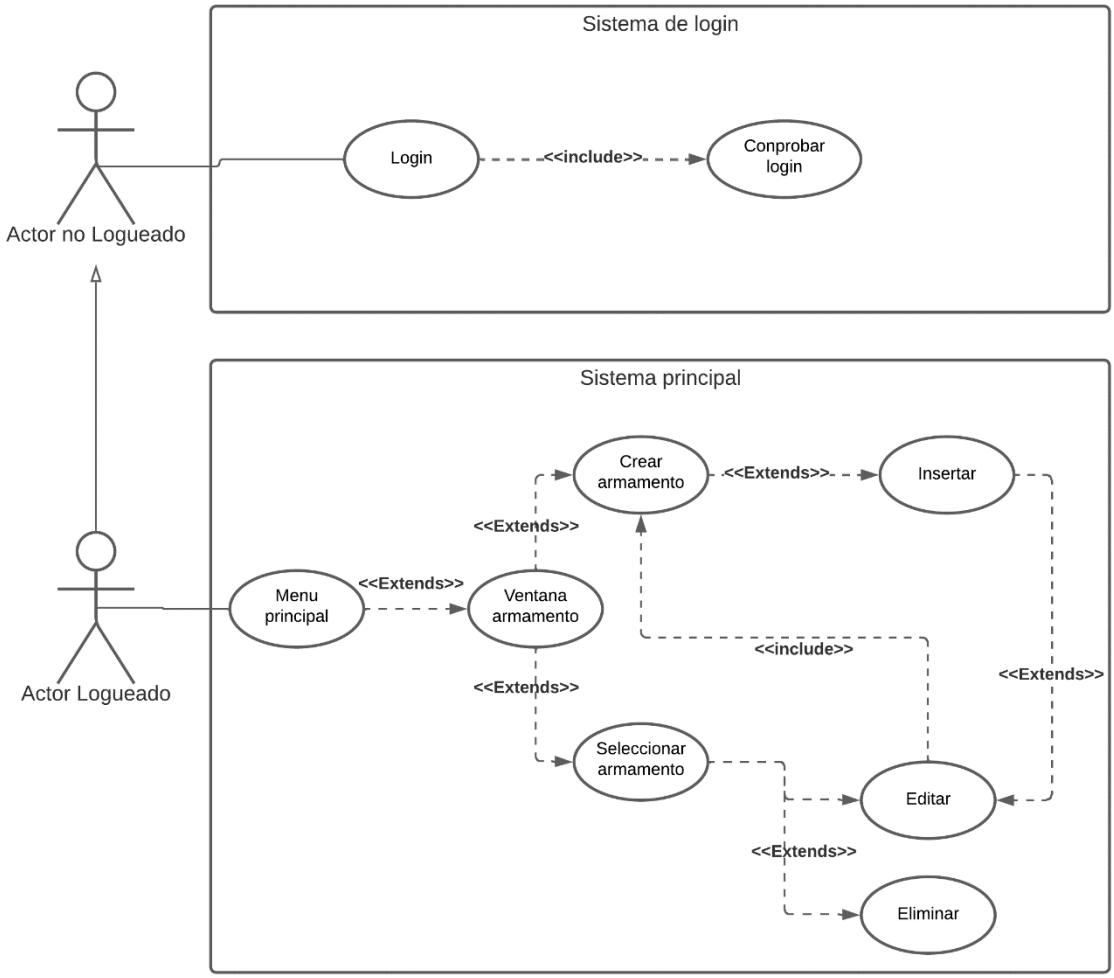

Figura 6 – Diagrama de casos de uso. Fuente: Elaboración propia en LucidChart.com

Anexo 6

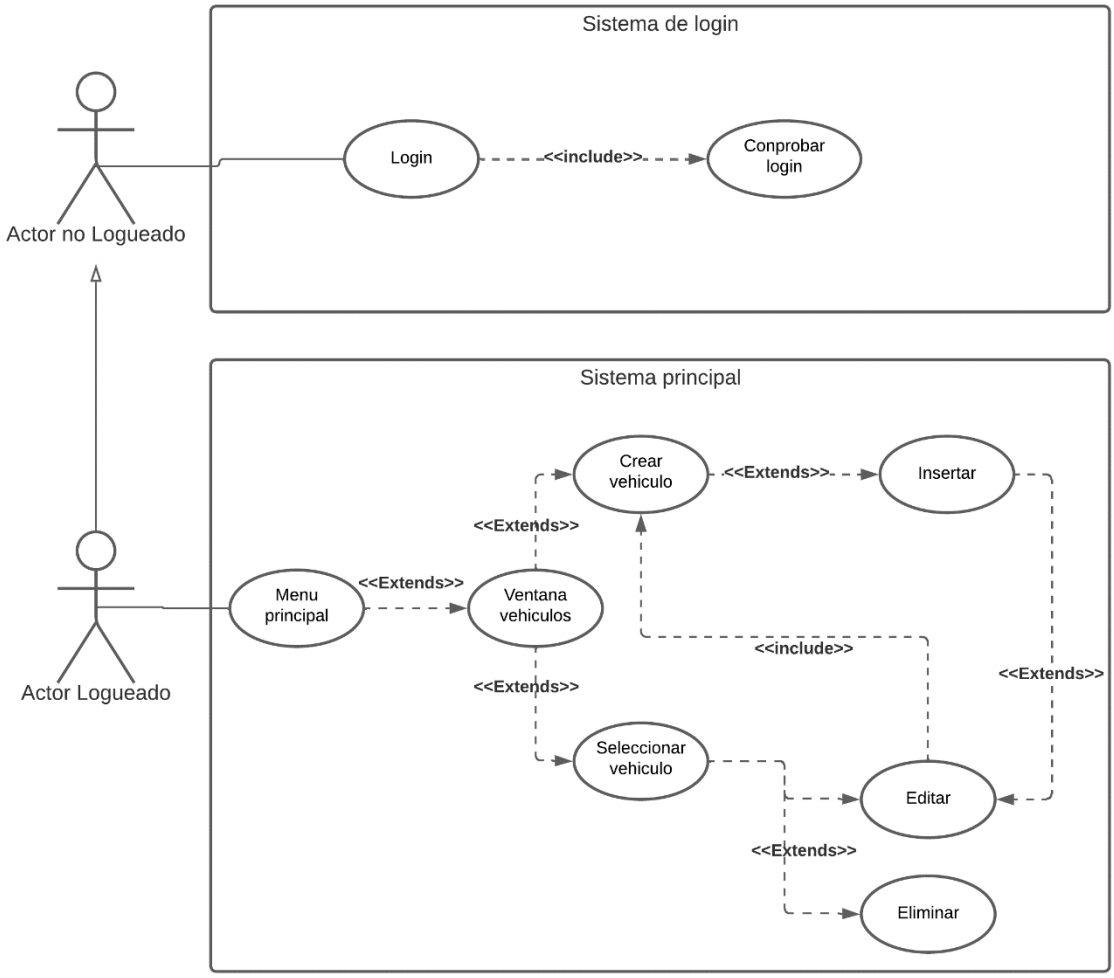

Figura 7 - Diagrama de casos de uso. Fuente: Elaboración propia en LucidChart.com

Anexo 7

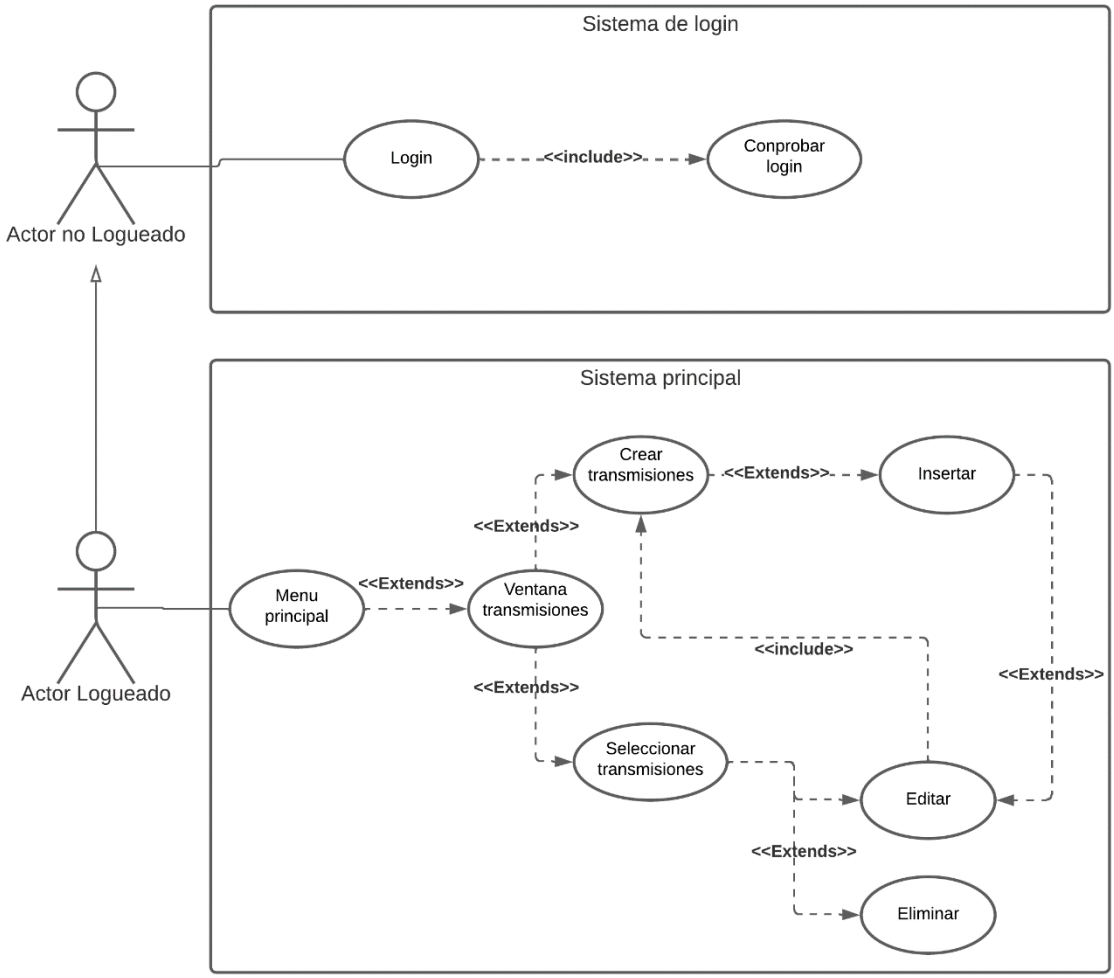

Figura 8 - Diagrama de casos de uso. Fuente: Elaboración propia en LucidChart.com

Anexo 8

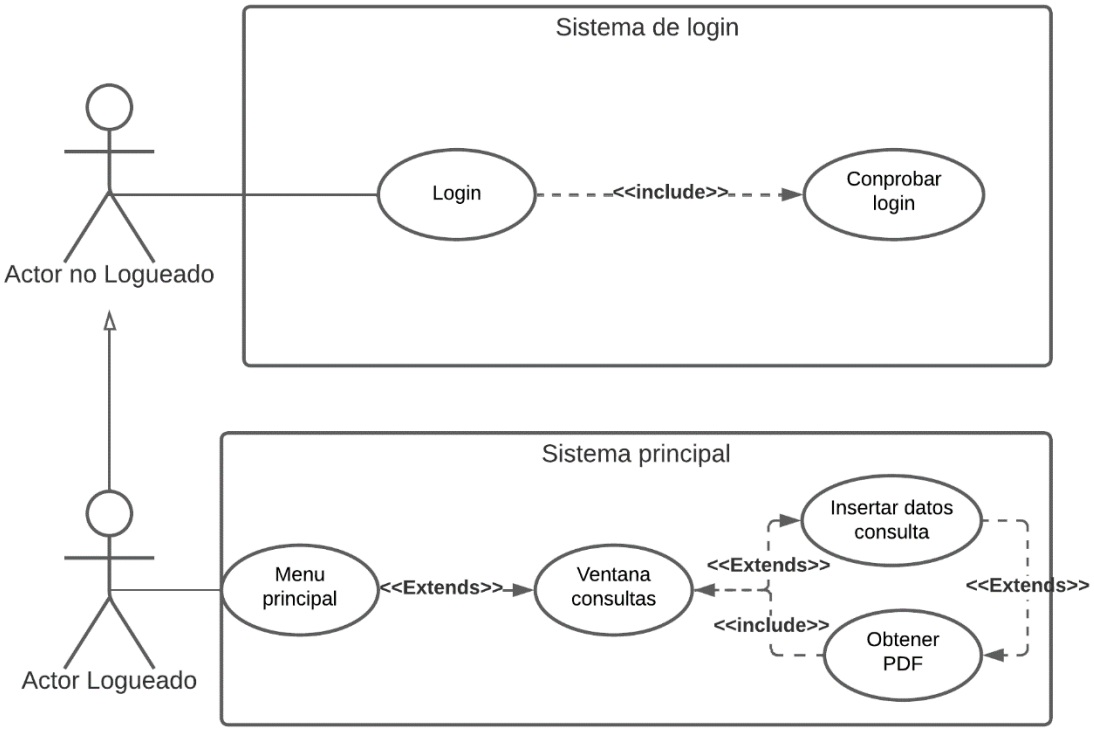

Figura 9 - Diagrama de casos de uso. Fuente: Elaboración propia en LucidChart.com

10.1. GLOSARIO

A lo largo del proyecto hemos tratado con terminologías que pueden traer consigo algún tipo de duda. Dentro de este apartado, definiremos algunos de los términos que podamos necesitar para comprender el contenido del documento.

- **IDE**: Un IDE es un software utilizado por programadores, para desarrollar el código de una aplicación.
- **NetBeans**: NetBeans es una herramienta que facilita el diseño de software por parte de un programador, y que utiliza de forma principal el lenguaje de programación Java.
- **Java**: Es un lenguaje de programación orientado a objetos, que es utilizado para desarrollar diferentes tipos de software.
- **Base de datos**: Una base de datos es una herramienta usada principalmente para almacenar, gestionar, y tener organizada la información.
- **Primary key**: Una primary key es una columna dentro de una base de datos, que identifica de manera única las filas de una tabla.
- **Foreign key**: Una foreign key es un tipo de datos que se utiliza dentro de las bases de datos para crear relaciones entre dos tablas.
- **Interfaz**: Una interfaz es la forma en la que un usuario se comunica con un ordenador, y la emplea para iterar con el.
- **Login**: Es un sistema que se emplea para acceder a un sistema, mediante la autenticación de un usuario con contraseña.
- **Librería**: Es un grupo de archivos que proporcionan funcionalidades a nuestra aplicación, y que facilitarán el desarrollo de esta.

¡Gracias por leer!