INSTITUTO TECNOLÓGICO SUPERIOR

"SAN GABRIEL"

ESPECIALIDAD INFORMÁTICA

TRABAJO DE INVESTIGACIÓN

PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

TECNÓLOGO EN INFORMÁTICA MENCIÓN ANÁLISIS EN SISTEMAS

TEMA:

"DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA INFORMÁTICO PARA EL CONTROL DE LAS COMISIONES VEHICULARES EN EL COMANDO DE APOYO LOGISTICO Nº 11 "CALICUCHIMA" AGREGADO A LA BRIGADA DE CABALLERIA BLINDADA Nº 11 "GALAPAGOS" REALIZADO EN JAVA Y GESTOR DE BASE DE DATOS MYSQL EN EL AÑO 2018"

PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN INFORMÁTICA MENCIÓN ANÁLISIS DE SISTEMAS

AUTOR:

LOVATO SANGACHA LUIS ALBERTO

RIOBAMBA – ECUADOR

AÑO 2018

CERTIFICACIÓN

PRESENTACIÓN

APROBACIÓN DE LA TESIS

FIRMAS DE RESPONSABILIDAD

DEDICATORIA

La presente tesis está dedicada a Dios por haberme dado la oportunidad de permitirme seguir cumpliendo con mis propósitos, así mismo, para mi familia por el apoyo incondicional y moral que me han brindado ya que ha sido fruto de perseverancia y orgullo.

Al Instituto Tecnológico Superior "San Gabriel", conjuntamente con su personal administrativo y docente, cuyo objetivo principal es el de aportar a la sociedad con profesionales capacitados para asumir responsabilidades y poder enfrentar adversidades con ética y responsabilidad.

AGRADECIMIENTO

Para las distinguidas autoridades del Instituto Tecnológico Superior “San Gabriel” por poseer una institución educativa de alto nivel, formando así profesionales responsables para apoyar al desarrollo del país.

Hacia el personal administrativo, docente y académico del Instituto Tecnológico Superior “San Gabriel”, por impartir sus conocimientos ligados con la experiencia cuya finalidad fue enriquecer el aprendizaje de los alumnos, ya que sin los conocimientos adquiridos no hubiese sido posible la realización del presente trabajo.

GLOSARIO DE TÉRMINOS

OPEN-SOURCE.- Código abierto.

SGBD.- Sistema gestor de base de datos.

JAVA.- Lenguaje de programación.

MYSQL.- Base de datos de código abierto.

GPL.- Licencia publica general de código abierto.

YEEPLY.- Plataforma de profesionales de desarrollo de apps y juegos para móviles.

HTML.- Sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web.

MOBINCUBE.- Es la forma más rápida y sencilla de crear aplicaciones para dispositivos móviles.

WAZE.- Es la aplicación de tráfico y navegación basada en la comunidad más grande del mundo.

NETBEANS.- IDE es un entorno de desarrollo visual de código abierto para aplicaciones programadas mediante Java.

XAMPP.- Es el entorno más popular de desarrollo con PHP, adicional es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl.

JASPERREPORTS.- Es una biblioteca de creación de informes que tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML.

IREPORT.- Aplicación para la instalación de pluggins de netbeans para la impresión de reportes.

INDICE GENERAL

ÍNDICE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La historia del Ejército ecuatoriano va de la mano con el servicio de la logística de tal manera que dentro del fortalecimiento de la institución armada se crea el Servicio de Transportes, cuya labor en más de dos siglos ha contribuido indiscutiblemente a la edificación del Ecuador cumpliendo con el traslado de personal, material y equipo para las tropas que se encuentran en servicio y en general para la ayuda comunitaria ante la sociedad, hay nuevos escenarios de enfrentamientos, es decir, cambiando las expresiones de un conflicto convencional a la protección de derechos, garantías y libertades de los ciudadanos.

El Ejercito tiene determinados sus sectores de responsabilidad de las cuales el servicio de transportes ha cumplido un papel importante al trasladar personal, material y equipo, para obtener resultados significativos como por ejemplo el decomiso de armas, munición y explosivos, entonces, es innegable que el nivel de poder combativo de las tropas al realizar sus maniobras se ven reflejadas, ante el oportuno abastecimiento realizado con los medios de transporte de los cuales está provisto el ejército.

Cabe reconocer que el éxito de las operaciones se basa en el oportuno apoyo y abastecimiento de los medios, en este caso primordialmente del servicio de transportes permitiendo que toda la información que se pueda requerir de dichos movimientos se los maneje detalladamente mediante un sistema informático que hace referencia a la actualidad informática en la que nos encontramos lo que permitirá obtener resultados rápidos, precisos y concisos para el cumplimiento de las misiones encomendadas.

Bajo este punto de vista, el presente trabajo de investigación tiene por finalidad desarrollar mediante lenguaje de programación java y gestor de base de datos MySQL un sistema informático para el control de las Comisiones Vehiculares del Comando de Apoyo Logístico Nº 11 “CALICUCHIMA” para determinar sucesos relevantes de los vehículos designados a la unidad y además facilitar una búsqueda rápida de información.

Para la realización del proyecto se requirió la siguiente estructura:

RESUMEN

El desarrollo e implementación de un sistema informático para el control de las comisiones vehiculares se lo realizo en una Unidad Logística que tiene por misión la movilidad de medios logísticos como es el Comando de Apoyo Logístico Nº 11 "CALICUCHIMA", por tal motivo para el presente proyecto se utilizó la metodología XP aplicada mediante: La búsqueda de requerimientos agiles que permiten la ejecución de un producto final con prontitud, permitiendo observar el proceso actual de la elaboración de órdenes de marcha, aplicando los conocimientos ya adquiridos para el desarrollo del sistema.

Además de la aplicación de métodos científicos, por ser un objetivo encaminado al desarrollo de un proyecto informático, este sistema permite la obtención de reportes amigables relacionados a vehículos, conductores y ordenes de marcha, documento que demanda de responsabilidad ya que permite la autorización conjuntamente con la legalización por parte de los responsables de la Unidad Militar, obteniendo un respaldo legal optimizando el sistema anterior y economizando medios como el tiempo, de manera clara sobre todo almacenando la información en una base de datos confiable, segura, capaz de brindarnos una búsqueda de documentos generados de forma oportuna. Finalmente con el desarrollo del sistema de control se obtuvo como resultados: reportes de los conductores disponibles, reportes de vehículos existentes, reportes de vehículos por conductor, reporte de la orden de marcha en menor tiempo con un respaldo seguro en la base de datos lo que cubrió las expectativas de la unidad militar.

CAPÍTULO I
MARCO REFERENCIAL

1.1. ANTECEDENTES DEL PROBLEMA

El Ejército Ecuatoriano se caracteriza por tener armas y servicios tanto e unidades administrativas, operativas y logísticas, en tal virtud el Comando de Apoyo Logístico Nº 11 “CALICUCHIMA” , se vio en la necesidad de automatizar el control de comisiones vehiculares con el desarrollo e implementación de un sistema informático de control que por pertenecer al servicio de Transportes y estar inmerso a diario en este tipo de comisiones vehiculares, obligatoriamente debe generar ordenes de marcha respaldando las mismas en un lugar seguro como en este caso lo será la base de datos, lo que garantizara una pronta creación de la orden y almacenamiento de la información con la seguridad de encontrar cualquier orden que haya sido generada con su respectivo reporte, mejorando el manejo de este control en la institución.

1.2. DEFINICIÓN DEL PROBLEMA

Actualmente el Comando de Apoyo Logístico Nº 11 “CALICUCHIMA” , no cuenta con un sistema informático para el control de las comisiones vehiculares que permita obtener una información general respaldada en una base de datos, la misma que podrá servir para poder determinar varios puntos específicos que se puedan presentar como inquietud dentro del servicio de Transportes, logrando satisfacer los requerimientos necesarios que ayuden al buen desarrollo de la unidad antes mencionada, generando así perdidas innecesarias de tiempo vinculando al información completa y detalla tomando en cuenta que se toma en cuenta factores que intervienen directamente en el normal desempeño y desarrollo de la institución.

1.3. JUSTIFICACIÓN

La presente investigación responde a la necesidad de aprovechar la tecnología actual dando a conocer soluciones informáticas al personal de transportes que pertenecen al Ejército Ecuatoriano, facilitando conocimientos sobre el uso de programas informáticos que permitan realizar una búsqueda más eficaz y a corto tiempo sobre todas las Comisiones Vehiculares realizadas hacia cada rincón del Ecuador, lo que permitirá operar y dar mantenimiento a los vehículos de la institución armada, además ayudara a la conservación del parque automotor y así poder apoyar al trabajo que día a día realizan los soldados del Ecuador ante los desastres naturales inclusive.

El proyecto esta creado para mantener en óptimas condiciones de uso, mantenimiento y Operabilidad los medios de transporte de los que está provisto el ejército, para cumplir con este objetivo se realizara un estudio de factibilidad en la ejecución de un sistema informático logrando obtener un control minucioso dentro de varios campos permitiendo ganar lapsos de tiempo y economía de medios para poder lograr cumplir con éxito las comisiones vehiculares encomendadas al personal de transportes, el cual permitirá evaluar las condiciones de viabilidad que se presentaran con el desarrollo de este sistema esclareciendo cualquier duda y facilitando la toma de decisiones dentro de un Teatro de Operaciones.

1.4. OBJETIVOS

1.4.1. Objetivo General

Diseñar e implementar un sistema informático para el control de las Comisiones Vehiculares del Comando de Apoyo Logístico Nº 11 "CALICUCHIMA", desarrollado en JAVA con Gestor de base de datos MYSQL.

1.4.2. Objetivos Específicos

- Investigar la compatibilidad que tiene el lenguaje de programación JAVA con el SGBD MYSQL a la hora de realizar sistemas informáticos.
- Determinar las necesidades que posee el Comando de Apoyo Logístico Nº 11 "CALICUCHIMA", al momento de generar el salvoconducto con el respaldo de la orden de marcha para seguridad del movimiento.
- Implementar un sistema informático para el control de las Comisiones Vehiculares del Comando de Apoyo Logístico Nº 11 "CALICUCHIMA.

CAPÍTULO II
MARCO TEÓRICO

2.1.- SISTEMAS INFORMÁTICOS[1]

INTRODUCCION

La relación del hombre en nuestra sociedad ha creado la necesidad de enviar y tratar la información de una forma continua, en tal virtud, en el transcurso del tiempo se ha perfeccionado varias técnicas y medios para lograr mejor efectividad.

En las dos últimas décadas del siglo xx ha tenido un avance y en la primera del siglo xxi se desarrollaron herramientas mucho más complejas e importantes para cubrir esta necesidad con gran precisión y rapidez.

El ordenador es una herramienta que actualmente nos facilita un tratamiento automático con la información, viabilizando en gran medida la organización, el proceso, la transmisión y almacenamiento de la misma, en el medio el término informática ha evolucionado notoriamente, pero actualmente llego a ser una ciencia que estudia el tratamiento automático de la información, se da su significado de la combinación de dos palabras: información y automática.

Su avance se refleja claramente en las dos últimas décadas del siglo xx, considerándose como una herramienta imprescindible en comunicaciones, telefonía, medicina, aeronáutica, vigilancia, control de tráfico, etc.

2.1.1.- EL SISTEMA INFORMÁTICO, SOFTWARE Y HARDWARE

A un ordenador lo podemos definir como una máquina conformada de elementos físicos (HARDWARE), mayoritariamente de origen eléctrico-electrónico, capaz de realizar una diversidad de trabajos a gran velocidad y con gran precisión, el ordenador se encuentra compuesto por un conjunto de componentes electrónicos que por sí mismos no son capaces de realizar muchas funciones.

Los componentes electrónicos necesitan de otros componentes no físicos que le permitan entrar en funcionamiento; nos referimos a programas (SOFTWARE), los programas sirven para nuestro beneficio: procesar datos (información).

[1] http://www.alegsa.com.ar/Dic/sistema_informatico.php

Por tal motivo para permitir que los componentes electrónicos de un ordenador sean capaces de funcionar y realicen un proceso determinado, es indispensable ejecutar un conjunto de órdenes o instrucciones, las instrucciones, ordenadas y agrupadas de forma adecuada, materializan un programa, la unión de varios programas se denomina aplicación informática.

Debemos recalcar que un programa no funciona por sí solo, necesariamente tenemos los componentes electrónicos, tenemos los programas que incluyen datos necesarios que se tienen que procesar, pero aún sigue faltando algo, el componente faltante que igual es un componente software y el objeto de la presente investigación, es el sistema operativo.

El sistema operativo es un componente software de un sistema informático con la capacidad de permitir que los programas procesen información sobre los componentes electrónicos de un ordenador o sistema informático, para ser más específico la unión de varios programas se denomina una aplicación informática, pero la aplicación informática puede darse por un único programa, en este caso no se llamaría aplicación informática sino simplemente programa.

En conclusión, una aplicación es un microprograma que se forma de varios programas independientes siempre y cuando estén interrelacionados; es decir, programas que funcionan de forma autónoma, pero que pueden necesitar información procesada por otros programas dentro del macro programa.

Aquellas instrucciones, todos los programas y una diversidad de aplicaciones informáticas quedan definidos bajo como software.

Un sistema informático se conforma conjunto de elementos físicos o hardware que son de vital importancia para el desarrollo de aplicaciones informáticas o software.

Un sistema informático o hardware es tangible, es decir, se puede ver y tocar (monitor, teclado, procesador, memoria).

El sistema operativo es intangible; o sea software, pero no se puede tocar ni ver el conjunto de instrucciones del cual se forma formado.

2.1.2.- TIPOS DE SISTEMAS INFORMÁTICOS

Dentro de los sistemas informáticos hay una división de los mismos que se puedan crear en base a la necesidad de cada usuario, tomando en cuenta el lugar de instalación y el uso que se le dará después de su creación, en tal virtud se divide en los siguientes:
Sistemas Informáticos web, de escritorio y móvil.

2.1.2.1.- SISTEMA WEB

Ilustración 1Diseño tradicional de un sistema informático

En informática denomina sistema o aplicación web a aquellas herramientas que muchos usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante cualquier navegador, es decir, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Las sistemas o aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, a la independencia del sistema operativo, así como a la como facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales, existen aplicaciones como los webmails, wikis, weblogs, tiendas en línea y la propia Wikipedia que son ejemplos bastante conocidos de aplicaciones web.

Cabe mencionar que una página Web puede contener muchos elementos que permiten una comunicación activa entre el usuario y la información, esto permite que el usuario

acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones, como llenar y enviar formularios, participar en juegos y acceder a gestores de base de datos de todo tipo.

ESTRUCTURA

Un sistema o aplicación web se estructura como una aplicación de tres-capas en su forma más común, el navegador web ofrece la primera capa, interpretando el código, la segunda es el servidor que ofrece este código, por último, una base de datos es la tercera capa.

El navegador web envía peticiones a la capa intermedia, la misma que ofrece servicios valiéndose de consultas y actualizaciones a la base de datos y proporciona una interfaz de usuario.

CONSIDERACIONES TÉCNICAS

Una ventaja relevante es que los sistemas o aplicaciones web deberían funcionar independientemente de la versión del sistema operativo instalado en el cliente, en vez de crear clientes para Windows, Mac OS X, GNU/Linux y otros sistemas operativos, el sistema o aplicación web se escribe una vez y se ejecuta en todas partes, sin embargo, hay aplicaciones inconsistentes escritas con HTML, CSS, DOM y otras especificaciones estándar para navegadores web que pueden ocasionar problemas en el desarrollo y soporte de estas aplicaciones, directamente debido a la falta de adhesión de los navegadores a dichos estándares web (especialmente versiones de Internet Explorer anteriores a la 7).

VENTAJAS

Ahorra tiempo ya que se pueden realizar tareas sencillas sin necesidad de descargar, ni instalar programas.

No hay problemas de compatibilidad porque basta tener un navegador actualizado para utilizarlas.

No ocupan demasiado espacio en nuestro disco duro.

Actualizaciones inmediatas en función de software ya que lo gestiona como el propio desarrollador, cuando nos conectamos estamos usando siempre la última versión que haya lanzado.

Consumo de recursos bajo puesto que toda o la mayoría de la aplicación no se encuentra en nuestra computadora.

2.1.2.2- SISTEMA INFORMÁTICO PARA PC (ESCRITORIO)

Ilustración 2Programacion para la elaboración del sistema informático

Un sistema informático o programa de computadora es una secuencia de instrucciones, escritas para realizar una tarea específica en una computadora, este dispositivo requiere programas para funcionar, ejecutando las instrucciones del programa en un procesador central, el programa tiene un formato ejecutable que la computadora puede utilizar directamente para ejecutar las instrucciones, el mismo programa en su formato de código es de fuente legible para los usuarios, del cual se derivan los programas ejecutables (por ejemplo, compilados), le permite a un programador estudiar y desarrollar sus algoritmos, una colección de programas de computadora y datos relacionados se conoce como software.

Normalmente, el código fuente lo escriben programadores de computadora, este código se escribe en un lenguaje de programación que sigue uno de los siguientes dos paradigmas: imperativo o declarativo y que después puede ser convertido en

un archivo ejecutable (usualmente llamado un programa ejecutable o un binario) por un compilador y más tarde ejecutado por una unidad central de procesamiento, adicionalmente, los programas de computadora se pueden ejecutar con la ayuda de un intérprete o pueden ser empotrados directamente en hardware.
De acuerdo a sus funciones, los programas informáticos se clasifican en software de sistema y software de aplicación.

PROGRAMACIÓN

```
#include <stdio.h>
#include <conio.h>

int main(void) {
   printf("Hola Mundo!\n");
   getch();
   return 0;
}
```

Código fuente del programa **Hola mundo** escrito en el Lenguaje de programación C

Ilustración 3Sentencias para el inicio de programación

Programación de computadoras es un proceso interactivo de editar código fuente, dicha edición implica probar, analizar y perfeccionar y a veces, coordinar con otros programadores, en el caso de un programa desarrollado en conjunto, así también una persona que practica esta técnica se le conoce como programador de computadoras, desarrollador de software o codificador, este proceso de programación de computadoras se lo conoce como desarrollo de software, el término ingeniería de software se está convirtiendo muy popular ya que esta actividad se ve como una disciplina de ingeniería.

CATEGORÍAS FUNCIONALES

Funcionales son software de sistema y software de aplicación, el software de sistema incluye al sistema operativo el cual acopla el hardware con el software de aplicación, el propósito del sistema operativo es proveer un ambiente en el cual el software de aplicación se ejecuta de una manera conveniente y eficiente, además del sistema operativo, el software de sistema incluye programas utilitarios que ayudan a manejar y configurar la computadora. Si un programa no es software de sistema entonces es software de aplicación, el middleware también es un software de aplicación que acopla el software de sistema con la interfaz de usuario, así como son software de aplicación los programas utilitarios que ayudan a los usuarios a resolver problemas de aplicaciones, como por ejemplo la necesidad de ordenamiento.

2.1.2.3.- APLICACIÓN MÓVIL

Ilustración 4Dispositivos móviles mostrando iconos de aplicaciones móviles.

Un sistema o aplicación móvil, app (en inglés) es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles y que permite al usuario efectuar una tarea concreta de cualquier tipo profesional, de ocio, educativas, de acceso a servicios, facilitando las actividades a desarrollar, por lo general, se encuentran disponibles a través de plataformas de distribución, operadas por las compañías propietarias de los sistemas operativos móviles como Android, iOS, BlackBerry OS, Windows Phone, etc.

Además existen aplicaciones móviles gratuitas u otras de pago, donde en promedio el 20-30 % del costo de la aplicación se destina al distribuidor y el resto es para el desarrollador, el término app se volvió relevante rápidamente, tanto que en 2010 fue listada como Word of the Year (Palabra del Año) por la American Dialect Society.

Al ser aplicaciones residentes en los dispositivos están escritas en algún lenguaje de programación compilado y su funcionamiento conjuntamente con sus recursos se encaminan a aportar una serie de ventajas como:

Un acceso más rápido y sencillo a la información necesaria sin necesidad de los datos de autenticación en cada acceso.

Un almacenamiento de datos personales que, a priori, es de una manera segura y de gran versatilidad en cuanto a su utilización o aplicación práctica.

La atribución de funcionalidades específicas.

Mejorar la capacidad de conectividad y disponibilidad de servicios y productos (usuario-usuario, usuario-proveedor de servicios).

DISEÑO Y DESARROLLO DE UNA APP

El desarrollo de sistemas o aplicaciones para dispositivos móviles requiere tener en cuenta las limitaciones de estos dispositivos, los dispositivos móviles funcionan con batería, hay que considerar una gran variedad de tamaños de pantalla, datos específicos de software y hardware como también distintas configuraciones, el desarrollo de aplicaciones móviles requiere el uso de entorno de desarrollo integrados.

Las aplicaciones móviles pueden aprovechar mucho más el contexto en el que se ejecutarán, sobre todo si se comparan con las aplicaciones tradicionales, ello se debe a diferentes factores, entre los que se encuentran las capacidades actuales en hardware de los dispositivos o la capacidad de acceder a la información del usuario a la que el propio dispositivo tiene acceso, los dispositivos actuales aportan mucha información sobre el entorno del usuario, por ejemplo, aportan información sobre la posición geográfica del mismo, lo cual permite desarrollar aplicaciones basadas en la localización, conocidas como (LBS, Servicio Basados en Localización), un ejemplo de tales aplicaciones es el Waze. Así mismo, existen otras informaciones (como por ejemplo, orientación, presión, luz, etc.).

Las aplicaciones móviles suelen ser probadas primero usando emuladores y más tarde se ponen en el mercado en periodo de prueba, actualmente un gran número de empresas se dedica a la creación profesional de aplicaciones, aun así, han surgido páginas web como Mobincube, donde un usuario común puede crear aplicaciones de manera gratuita y sin conocimiento de programación y plataformas como Yeeply, que te ayuda a encontrar desarrolladores y hacer de guía para crear tu app móvil.

2.1.3.- TIPOS DE LICENCIAS DE SOFTWARE

Licencia.- Es un contrato entre el desarrollador de un software sometido a propiedad intelectual y a derechos de autor y el usuario, en el cual se definen con precisión los derechos y deberes de ambas partes, es el desarrollador o aquél a quien éste haya cedido los derechos de explotación, quien elige la licencia según la cual distribuye el software.
Patente.- Es un conjunto de derechos exclusivos garantizados por un gobierno o autoridad al inventor de un nuevo producto (material o inmaterial) susceptible de ser explotado industrialmente para el bien del solicitante por un periodo de tiempo limitado.
Derecho de autor o copyright.- Es una forma de protección proporcionada por las leyes vigentes en la mayoría de los países para los autores de obras originales incluyendo obras literarias, dramáticas, musicales, artísticas e intelectuales, tanto publicadas como pendientes de publicar.

2.1.3.1.- SOFTWARE LIBRE

Proporciona la libertad de ejecutar el programa, para cualquier propósito, estudiar el funcionamiento del programa y adaptarlo a sus necesidades, redistribuir copias, mejorar el programa y poner sus mejoras a disposición del público, para beneficio de toda la comunidad.

2.1.3.2.- SOFTWARE DE FUENTE ABIERTA

Sus términos de distribución cumplen los criterios de distribución libre, Inclusión del código fuente, permitir modificaciones y trabajos derivados en las mismas condiciones

que el software original, integridad del código fuente del autor, pudiendo requerir que los trabajos derivados tengan distinto nombre o versión, no discriminación a personas o grupos, sin uso restringido a campo de actividad, los derechos otorgados a un programa serán válidos para todo el software redistribuido sin imponer condiciones complementarias, la licencia no debe ser específica para un producto determinado, la licencia no debe poner restricciones a otro producto que se distribuya junto con el software licenciado, la licencia debe ser tecnológicamente neutral. Estándar abierto: según Bruce Perens, el basado en los principios de disponibilidad, maximizar las opciones del usuario final, sin tasas sobre la implementación, sin discriminación de implementador, permiso de extensión o restricción, evitar prácticas predatorias por fabricantes dominantes.

2.1.3.3.- SOFTWARE CON COPYLEFT

Es un Software libre cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando lo redistribuyen o modifican, o sea, la versión modificada debe ser también libre.

Software semi libre es aquél que no es libre, pero viene con autorización de usar, copiar, distribuir y modificar para particulares sin fines de lucro.

Freeware se usa comúnmente para programas que permiten la redistribución pero no la modificación (y su código fuente no está disponible).

Shareware: software con autorización de redistribuir copias, pero debe pagarse cargo por licencia de uso continuado.

Software privativo es aquél cuyo uso, redistribución o modificación están prohibidos o necesitan una autorización.

2.2.- LENGUAJE DE PROGRAMACIÓN JAVA[2]

DEFINICIÓN

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

2.2.1.- EVOLUCIÓN HISTÓRICA.

Con el paso del tiempo, se va incrementando el nivel de abstracción, pero en la práctica, los de una generación no terminan de sustituir a los de la anterior:

- Lenguajes de primera generación (1GL): Código máquina.
- Lenguajes de segunda generación (2GL): Lenguajes ensamblador.
- Lenguajes de tercera generación (3GL): La mayoría de los lenguajes modernos, diseñados para facilitar la programación a los humanos. Ejemplos: C, Java.

[2] IBM. (16 de MAR de 1998). Recuperado el 12 de FEB de 2017, de https://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava1/index.html

- Lenguajes de cuarta generación (4GL): Diseñados con un propósito concreto, o sea, para abordar un tipo concreto de problemas. Ejemplos: NATURAL, Mathematica.
- Lenguajes de quinta generación (5GL): La intención es que el programador establezca el qué problema ha de ser resuelto y las condiciones a reunir, y la máquina lo resuelve. Se usan en inteligencia artificial. Ejemplo: Prolog.

2.2.2.- NIVEL DE ABSTRACCIÓN

Según el nivel de abstracción, o sea, según el grado de cercanía a la máquina:

- Lenguajes de bajo nivel: La programación se realiza teniendo muy en cuenta las características del procesador. Ejemplo: Lenguajes ensamblador.
- Lenguajes de nivel medio: Permiten un mayor grado de abstracción pero al mismo tiempo mantienen algunas cualidades de los lenguajes de bajo nivel. Ejemplo: C puede realizar operaciones lógicas y de desplazamiento con bits, tratar todos los tipos de datos como lo que son en realidad a bajo nivel (números), etc.
- Lenguajes de alto nivel: Más parecidos al lenguaje humano. Manejan conceptos, tipos de datos, etc., de una manera cercana al pensamiento humano ignorando (abstrayéndose) del funcionamiento de la máquina. Ejemplos: Java, Ruby.

2.2.3.- PROPÓSITO.

Según el propósito, es decir, el tipo de problemas a tratar con ellos:

- Lenguajes de propósito general: Aptos para todo tipo de tareas: Ejemplo: C.
- Lenguajes de propósito específico: Hechos para un objetivo muy concreto. Ejemplo: Csound (para crear ficheros de audio).
- Lenguajes de programación de sistemas: Diseñados para realizar sistemas operativos o drivers. Ejemplo: C.
- Lenguajes de script: Para realizar tareas varias de control y auxiliares. Antiguamente eran los llamados lenguajes de procesamiento por lotes (batch) o

JCL (“Job Control Languages”). Se subdividen en varias clases (de shell, de GUI, de programación web, etc.). Ejemplos: bash (shell), mIRC script, JavaScript (programación web).

2.2.4.- SEGÚN LA MANERA DE EJECUTARSE

Lenguajes compilados: Un programa traductor traduce el código del programa (código fuente) en código máquina (código objeto). Otro programa, el enlazador, unirá los ficheros de código objeto del programa principal con los de las librerías para producir el programa ejecutable. Ejemplo: C.
Lenguajes interpretados: Un programa (intérprete), ejecuta las instrucciones del programa de manera directa. Ejemplo: Lisp.
También los hay mixtos, como Java, que primero pasan por una fase de compilación en la que el código fuente se transforma en “bytecode”, y este “bytecode” puede ser ejecutado luego (interpretado) en ordenadores con distintas arquitecturas (procesadores) que tengan todos instalados la misma “máquina virtual” Java.

2.2.5.- SEGÚN LA MANERA DE ABORDAR LA TAREA A REALIZAR

Lenguajes imperativos: Indican cómo hay que hacer la tarea, es decir, expresan los pasos a realizar. Ejemplo: C.
Lenguajes declarativos: Indican qué hay que hacer. Ejemplos: Lisp, Prolog. Otros ejemplos de lenguajes declarativos, pero que no son lenguajes de programación, son HTML (para describir páginas web) o SQL (para consultar bases de datos).

2.2.6.- PARADIGMA DE PROGRAMACIÓN

El paradigma de programación es el estilo de programación empleado. Algunos lenguajes soportan varios paradigmas, y otros sólo uno. Se puede decir que históricamente han ido apareciendo para facilitar la tarea de programar según el tipo de problema a abordar, o para facilitar el mantenimiento del software, o por otra cuestión similar, por lo que todos corresponden a lenguajes de alto nivel (o nivel medio), estando

los lenguajes ensambladores “atados” a la arquitectura de su procesador correspondiente. Los principales son:

Lenguajes de programación procedural: Divide el problema en partes más pequeñas, que serán realizadas por subprogramas (subrutinas, funciones, procedimientos), que se llaman unas a otras para ser ejecutadas. Ejemplos: C, Pascal.

Lenguajes de programación orientada a objetos: Crean un sistema de clases y objetos siguiendo el ejemplo del mundo real, en el que unos objetos realizan acciones y se comunican con otros objetos. Ejemplos: C++, Java.

Lenguajes de programación funcional: La tarea se realiza evaluando funciones, (como en Matemáticas), de manera recursiva. Ejemplo: Lisp.

Lenguajes de programación lógica: La tarea a realizar se expresa empleando lógica formal matemática. Expresa qué computar. Ejemplo: Prolog.

Hay muchos paradigmas de programación: Programación genérica, programación reflexiva, programación orientada a procesos, etc.

2.2.7.- LUGAR DE EJECUCIÓN

En sistemas distribuidos, según dónde se ejecute:

Lenguajes de servidor: Se ejecutan en el servidor. Ejemplo: PHP es el más utilizado en servidores web.

Lenguajes de cliente: Se ejecutan en el cliente. Ejemplo: JavaScript en navegadores web.

2.2.8.- CONCURRENCIA

Según admitan o no concurrencia de procesos, esto es, la ejecución simultánea de varios procesos lanzados por el programa:

Lenguajes concurrentes. Ejemplo: Ada.

Lenguajes no concurrentes. Ejemplo: C.

2.2.9.- INTERACTIVIDAD

En la interactividad del programa con el usuario u otros programas:
Lenguajes orientados a sucesos: El flujo del programa es controlado por la interacción con el usuario o por mensajes de otros programas/sistema operativo, como editores de texto, interfaces gráficos de usuario (GUI) o kernels. Ejemplo: VisualBasic, lenguajes de programación declarativos.
Lenguajes no orientados a sucesos: El flujo del programa no depende de sucesos exteriores, sino que se conoce de antemano, siendo los procesos batch el ejemplo más claro (actualizaciones de bases de datos, colas de impresión de documentos, etc.). Ejemplos: Lenguajes de programación imperativos.

2.2.10.- REALIZACIÓN VISUAL

Según la realización visual o no del programa:
Lenguajes de programación visual: El programa se realiza moviendo bloques de construcción de programas (objetos visuales) en un interfaz adecuado para ello. No confundir con entornos de programación visual, como Microsoft Visual Studio y sus lenguajes de programación textuales (como Visual C#). Ejemplo: Mindscript.
Lenguajes de programación textual: El código del programa se realiza escribiéndolo. Ejemplos: C, Java, Lisp.

2.2.11.- HERRAMIENTAS QUE SE VA UTILIZAR EN JAVA

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios (¡y creciendo!) en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal del proyecto
NetBeans 8.2 proporciona analizadores de código y editores para trabajar con las últimas tecnologías Java 8, Java SE Embedded 8 y Java ME Embedded 8. El IDE también tiene una gama de nuevas herramientas para HTML5 / JavaScript, en particular

para Node.js, KnockoutJS y AngularJS; Mejoras que mejoran su soporte para Maven y Java EE con PrimeFaces; Y mejoras a PHP y soporte C / C ++.

2.2.12.- DOCUMENTACIÓN

Utilice lo siguiente para comenzar con NetBeans IDE 8.2:
Instrucciones de instalación
Screencasts y Tutoriales
Notas de la versión de Biblioteca de documentación NetBeans IDE 8.2

2.2.13.- PRIMEFACES

PrimeFaces es una librería de componentes visuales open source desarrollada y mantenida por Prime Technology, una compañía Turca de IT especializada en consultoría ágil, JSF, Java EE y Outsourcing. El proyecto es liderado por Çağatay Çivici, un miembro del "JSF Expert Group" (y forofo del Barça).
Las principales características de Primefaces son:
Kit para crear aplicaciones web para móviles.
Es compatible con otras librerías de componentes, como JBoss RichFaces.
Uso de javascript no intrusivo (no aparece en línea dentro de los elementos, sino dentro de un bloque <script>).
Es un proyecto open source, activo y bastante estable entre versiones.
Algunos inconvenientes podrían ser:
Para utilizar el soporte de Ajax tenemos que indicarlo explícitamente, por medio de atributos específicos de cada componente.
No podemos utilizar el soporte de Ajax de JSF 2 (mediante <f:ajax>) con los componentes de Primefaces.

2.2.14..-VENTAJAS Y DESVENTAJAS SOBRE EL USO DE JAVA[3]

2.2.14.1.- VENTAJAS DE JAVA

Al utilizar este tipo de programación para algún sistema informático se puede notar muchas situaciones útiles para el desenvolvimiento de las mismas como las que se detalla a continuación:

La multiplataforma.- Podría ser que sí que el precio por tenerla sea la lentitud, pero es que su principal orientación sea el rendimiento en sí sino la facilidad para desarrollar aplicaciones para distintas tecnologías (de escritorio, móvil, web...)

Una vez se aprenda su sintaxis, son muy fáciles de alternar.

Es el JDK, una librería de clases bastante completa que se puede aprovechar gracias a un lenguaje perfectamente orientado a objetos que corriendo sobre la máquina virtual, le permite olvidar de algo tan engorroso como es la gestión de memoria (punteros, reserva y liberación de memoria).

Algo primordial pero eso también puede ser bueno, deja las cosas 'claras'.

Gran cantidad de recursos disponibles, tanto en librerías como en documentación y comunidad.

Más rápido que lenguajes interpretados y no mucho más lento que lenguajes compilados, como siempre hay opción, no está atado a ningún ide, librería o modo de hacer las cosas.

2.2.15.- DESVENTAJAS DE JAVA

Su sintaxis.- Si la comparamos con C# o Python la veo bastante engorrosa y al contrario que su semejante en .NET, C#, es un lenguaje que evoluciona muy lentamente.

No es tan rico en características (LINQ, tipiado anónimo).

[3] BLOGSPOT.COM. (17 de ABR de 2014). Recuperado el 15 de FEB de 2017, de http://programemosenjava.blogspot.com/2014/04/ventajas-y-desventajas-de-java.html

El principal objetivo no es su rendimiento y esto aunque aporta una ventaja también un inconveniente.
Es para la creación de aplicaciones multimedia o que impliquen funcionalidad mínimamente avanzada (de visión por computador).
Es difícil de aprender, tomándolo como el primer lenguaje que se pretende conocer; para poder empezar a desarrollar aplicaciones con en él son necesarias unas nociones de orientación a objetos mínimas que para otros lenguajes (Python o C) no serían necesarias y con los que los conocimientos podrían ir escalándose poco a poco.
No sé, al fin y al cabo es una herramienta que como profesional hay que saber dominar y donde aplicarla).

2.2.16.- LA IMPORTANCIA DE USAR JAVA ACTUALMENTE

Java es el lenguaje de programación más usado del mundo y además, su uso sigue creciendo cada día más y más, ya que la demanda de expertos en Java es de las más altas, si aprendes este lenguaje de programación te asegurarás un gran futuro laboral que no debes desaprovechar.
Para crear aplicaciones Android necesitas saber Java y con los conocimientos necesarios tienes la oportunidad de desarrollar aplicaciones para un mercado con un crecimiento enorme, podrás desarrollar aplicaciones para que miles de usuarios puedan descargarla y usarla en sus dispositivos móviles.
Java es multiplataforma, podrás desarrollar una sola aplicación que funcione en cualquier plataforma, ya sea Windows, Mac o Linux y no tendrás que pagar ninguna licencia porque es completamente gratuito usar esta tecnología.
La comunidad de Java tiene disponible un gran soporte y documentación para seguir aprendiendo y resolver todas las dudas y problemas que te surjan a la hora de desarrollar tus aplicaciones y lo mejor en español.
Java es código abierto, es presente y futuro de la programación de propósito general y orientado a objetos, grandes marcas como Google, Amazon o netflix lo usan no esperes más y aprende Java desde cero y hazte experto Java.

2.3.- GESTOR DE BASE DE DATOS MYSQL

2.3.1.- MYSQL: UN SGBD DE CÓDIGO ABIERTO

MYSQL es un sistema gestor de bases de datos que se ubica dentro de la categoría de los programas OPEN-SOURCE, para definir un programa open-source se puede decir que cuyo código fuente se encuentra disponible para los usuarios y abierto a modificaciones.

El siguiente termino OPEN-SOURCE se basó al free-software, que hacía referencia a los programas cuyo código estaba disponible para el usuario, de manera gratuita o no, ya que free en inglés significa libre pero a la vez gratis, lo que causaba confusión, por lo que finalmente se decidió utilizar OPEN-SOURCE que significa código abierto, hay que tener en cuenta que OPEN-SOURCE no siempre refiere a que su uso sea gratuito, MYSQL sí que lo es.

Si en este caso el programa antes mencionado se lo puede adquirir gratis, el usuario está en la obligación de transmitir gratis también los cambios que tenga a bien realizarlos, o los productos que pueda desarrollar en beneficio de mejoras basándose en el original, caso contrario, si el usuario desea optar por hacer un negocio con el producto, se ve en la obligación de adquirir la licencia comercial de pago.

2.3.2.- ORÍGENES E HISTORIA DE MYSQL

MYSQL es un caso especial, ya que por tratarse de un pro-grama con licencia OPEN-SOURCE y gratuito pero que, sin embargo, se rige por una empresa, MYSQL AB, con sede en Suecia.

El código fuente de MYSQL está sólo relativamente abierto y disponible para modificaciones, puesto que es la empresa MYSQL AB la que contrata y coordina los trabajos de mantenimiento del producto, no obstante, los trabajadores contratados, procedentes de todo el mundo, son usuarios del producto que realizan sus encargos a través de Internet.

El origen de MYSQL viene desde década de los ochenta, en donde Michael Widenius, o conocido también como Monty, era joven programador que desarrollaba complejas

aplicaciones en lenguaje BASIC, al no identificar un sistema de almacenamiento de archivos que le resultara satisfactorio, se le vino la idea de construir el suyo propio.
Años más tarde, en 1995, y con la colaboración de David Axmark, Widenius desarrolló un producto que básicamente fue el resultado de varias de sus investigaciones, conjuntamente con dos aportaciones nuevas que fueron el uso del lenguaje SQL y la accesibilidad a través de Internet, de esta manera nació MySQL y a la par la empresa MySQL AB.El origen de MySQL viene desde década de los ochenta, en donde Michael Widenius, o conocido también como Monty, era joven programador que desarrollaba complejas aplicaciones en lenguaje BASIC, al no identificar un sistema de almacenamiento de archivos que le resultara satisfactorio, se le vino la idea de construir el suyo propio.
Años más tarde, en 1995, y con la colaboración de David Axmark, Widenius desarrolló un producto que básicamente fue el resultado de varias de sus investigaciones, conjuntamente con dos aportaciones nuevas que fueron el uso del lenguaje SQL y la accesibilidad a través de Internet, de esta manera nació MySQL y a la par la empresa MySQL AB.

2.3.3.- BENEFICIOS

MySQL es un gestor de base de datos bastante versátil, que cuenta con una gran cantidad de beneficio para trabajar:
Cuenta con la capacidad de realizar tareas multiprocesador, debido a que posee la opción de trabajo multihilo.
Puede ingresar una enorme cantidad de datos por columna de trabajo.
Cuenta con API’s disponibles para los principales lenguajes de programación que existen.
Aplicación con una portabilidad sobresaliente.
Capacidad de soportar hasta 32 índices de tablas diferentes.
Estupendo nivel de seguridad que permite gestionar varios usuarios con login y contraseñas individuales.

2.3.4. - VENTAJAS

- MySQL software es Open Source2.
- Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación. Soporta gran variedad de Sistemas Operativos.

  Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- Su conectividad, velocidad, y seguridad hacen de MySQL Server altamente apropiado para acceder bases de datos en Internet.
- El software MySQL usa la licencia GPL.

2.3.5.- DESVENTAJAS

- Un gran porcentaje de las utilidades de MySQL no están documentadas.
- No es intuitivo, como otros programas (ACCESS).

2.4.- MÓDULO DE APACHE

El módulo apache parea Linux se llama “mod_omnis.so” y se encuentra en la carpeta “webelient/server dentro del directorio principal de Omnis.

El módulo de Apache para Red Hat versión 6.0, y otras distribuciones de Linux, resultan más rápido que si se usa la extensión de servidor “nph-omnisegi”. (Muñoz, 2008, pág. 195)

Las principales razones por las cuales se usa este módulo son:

Arranca en casi todos los S.O., es decir es multiplataforma.

Apache es de libre uso permitiendo observar su instalación como servidor sin ninguna restricción.
Fácil de configurar y de ampliamente de capacidad como servidor.
Este módulo soporta una gran cantidad de lenguajes orientados al desarrollo de páginas dinámicas, tales como: PHP, PERL, Java.

2.5.- XAMPP

Es un servidor independiente de plataforma, software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. (Ramos, 2011, pág. 14)
XAMPP incluye además servidores de bases de datos como MySQL ySQLite con sus respectivos gestores phpMyAdmin y phpSQLiteAdmin. Ademas incorpora también el intérprete de PHP, el intérprete de Perl, servidores de FTP como PrFTPD FileZilla FTP Serve, etc. entre muchas cosas más.
El XAMPP es una herramienta de desarrollo que te permite probar tu trabajo (páginas web o programación por ejemplo) en tu propio ordenador sin necesidad de tener que acceder al internet.
XAMPP provee de una configuración totalmente funcional desde el momento que lo instalas. Sin embargo, es bueno acotar que la seguridad de datos no es su punto fuerte, por lo cual no es suficientemente seguro para ambientes grandes o de producción.

2.5.1.- PAQUETES QUE VIENEN CON XAMPP

Apache, el servidor Web más famoso.
MySQL, una excelente base de datos de código libre.
PHP y Perl: lenguajes de programación.
ProFTPD: un servidor FTP.
OpenSSL: para soporte a la capa de sockets segura.

2.5.2.- PAQUETES GRÁFICOS

GD (Graphics Draw): la librería de dibujo de gráficos.

Libpng: la librería oficial de referencia de PNG.

Libpeg: la librería oficial de referencia de JPEG.

Ncurses: la librería de gráficos de caracteres.

2.5.3.- PAQUETE DE BASES DE DATOS

Gdbm: la implementación GNU de la librería standard dbm de UNIX.

SQLite: un motor de base de datos SQL muy pequeño y cero configuración.

FreeTDS: una librería de base de datos que da a los programas de Linux y UNIX la habilidad de comunicarse con Microsoft SQL y Sybase.

2.5.4.- PAQUETES XML

Expat: una librería parser de XML.

Salbotron: una toolkit de XML.

Libxml: un parser C de XML y un toolkit para GNOME.

2.5.5.- PAQUETES PHP

PEAR: la librería de PHP.

Una clase pdf que genera documentos PDF dinámicos con PHP.

TURCK MM Cache: un potenciador de la performance de PHP.

2.5.6.- OTROS PAQUETES

Zlib: una librería de compresión.

Mod_perl: empotra un intérprete de Perl en Apache.

Gettext: un conjunto de herramientas que asiste a los paquetes GNU para producir mensajes multilingües.

Mcrypt: un programa de encriptación.

Ming: una librería de salida en Flash.

IMAC C-Client: un API de correos

CAPÍTULO III
ANÁLISIS Y DISEÑO DEL SISTEMA

3.1.- RECOPILACIÓN DE INFORMACIÓN

TÉCNICAS DE RECOLECCIÓN DE DATOS

FUENTES

Los principales tipos de fuentes de donde se obtendrá la información para el desarrollo del presente proyecto se clasifican en dos:
Fuentes Primarias
Fuentes Secundarias

DATOS O FUENTES PRIMARIAS

Los datos primarios son aquellos que nosotros como investigadores obtenemos directamente de la realidad, recogiéndolos (produciéndolos) con nuestros propios instrumentos. Son datos de primera mano.

OBSERVACIÓN

Consiste en obtener la información sin tener que interrogar al individuo.
En este sentido los datos son más asépticos (J.Ferré, 2008, pág. 74).
Consiste en especificar la información para el proceso de estudio en este caso es:

RECOPILACIÓN DE INFORMACIÓN ACERCA DE LAS COMISIONES VEHICULARES, PREVIO AL DESARROLLO DE UN SISTEMA INFORMÁTICO PARA EL CONTROL DE LAS COMISIONES VEHICULARES EN EL COMANDO DE APOYO LOGISTICO Nº 11 "CALICUCHIMA"

Dentro del desarrollo de la recopilación de información para determinar el actual proceso y obtener requerimientos necesarios para mejorar el control de las comisiones vehiculares determinamos lo siguiente:

PROCESO ACTUAL

- La información se almacena en un disco de unidad de la PC de una oficina.
- El ingreso a la información almacenada no posee ningún tipo de seguridad ya que puede ingresar tanto el amanuense encargado como cualquier conductor que necesite generar una orden de marcha para las comisiones vehiculares.
- El proceso de asignación de comisiones vehiculares se lo realiza sin respaldo seguro.
- La realización de un nuevo documento para la asignación de una comisión vehicular se lo realiza solo corrigiendo ciertas líneas de texto mediante el mouse y teclado.
- El modo de generar un reporte es manual en un documento de Word.
- Se adjunta una réplica de una orden de marcha de una comisión vehicular en la cual se basa para recopilar la información.

NECESIDADES PREVISTAS

- Automatizar el proceso actual que mantiene la unidad para el control de comisiones vehiculares.
- Generar la orden de marcha para el cumplimiento de la comisión vehicular.
- Obtener un reporte amigable con la información de la comisión vehicular.
- Obtención de reportes con el listado general de conductores.
- Permitir un control de vehículos mediante un reporte.
- Acceso únicamente para dos usuarios principal y secundario para determinar responsabilidad.
- Poder respaldar la información ingresada en una base de datos.
- Poder utilizar la tecnología para mejorar nuestros procesos logísticos en el ámbito vehicular.

INFORME DE NECESIDADES ACERCA DEL DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA INFORMÁTICO PARA EL CONTROL DE COMISIONES VEHICULARES EN EL COMANDO DE APOYO LOGISTICO Nº 11 "CALICUCHIMA"

Mediante el estudio que se ha realizado en la unidad militar antes mencionada y analizando lo recopilación de información se ha determinado las siguientes necesidades técnicas para el desarrollo del sistema informático:

Seguridad en la creación de usuarios para que puedan acceder personas autorizadas al sistema.

Creación de una base de datos para respaldar la información.

Almacenamiento de datos personales de todo el personal de conductores de la unidad militar mencionada.

Almacenamiento de la información principal de vehículos existentes en la unidad militar.

Asignación automática de las comisiones vehiculares al personal militar.

Selección automática de conductores y vehículos para las comisiones vehiculares.

Reportes amigables para los conductores con la información detallada del movimiento vehicular.

3.2.- ANÁLISIS

ESTUDIO DE FACTIBILIDAD

TÈCNICA:

La aplicación se realizará en Java y Mysql con la ayuda de sus respectivos IDES y Netbeans, Workbench, programas de fácil acceso contando con todos los recursos, materiales y programas necesarios para la elaboración y ejecución tanto en Hardware y Software.

Por los antecedentes mencionados es viable la generación de esta herramienta.

HARDWARE	
Laptop Hp Core I5 32 bits RAM 4GB Disco Duro 640GB	
SOFTWARE	
Windows 7	Software libre
NETBEANS 8.2	Software libre
JAVA	Software Libre
MYSQL	Software Libre
PLUGINS IREPORT	Software Libre

Tabla 1Factibilidad Técnica

Fuente: Propia
Elaborador por: Luis Lovato

OPERATIVA:

ACTIVIDAD ANTERIOR	TIEMPO ANTERIOR	ACTIVIDAD ACTUAL	TIEMPO ACTUAL
Ingreso o ejecución del Sistema Informático	15 segundos	Ingreso o ejecución del Sistema Informático	5 segundos
Ingreso de usuario y contraseña de seguridad	No tiene	Ingreso de usuario y contraseña de seguridad	10 segundos
Revisión de la información almacenada en la Base de Datos	8 minutos	Revisión de la información almacenada en la Base de Datos	2 minutos
Modificación de información de la Base de Datos	15 minutos	Modificación de información de la Base de Datos	4 minutos
TOTAL	23 min, 15 seg.	**TOTAL**	6 min, 15 seg.

Tabla 2Factibilidad Operativa

Fuente: Propia
Elaborador por: Luis Lovato

ECONÓMICA

Para la elaboración del presente proyecto que utiliza varias herramientas y programas.
El investigador cubrirá el 100% de gastos que incurra en el uso de equipos para el desarrollo del sistema informático ya que los medios de software son libres que permiten economizar recursos económicos para el desarrollo e implementación del mismo.

LEGAL

Las Herramientas que se requieren utilizar son software de open source completamente legales que responden a la necesidad de controlar y respaldar la información de la Unidad Militar CAL-11, que cumple con todos los requisitos enmarcados en la constitución nacional vigente.
Además, sigue los lineamientos del reglamento interno sin afectar las normas, reglamentos y directivas establecidas en la misma.
Por otra parte se encuentra con todos los derechos de autor y propiedad intelectual.
Por los antecedentes mencionados legalmente es factible la realización de la investigación a implementarse en la institución.

3.2.- ANÁLISIS DE REQUERIMIENTOS (FUNCIONALES Y NO FUNCIONALES)

REQUERIMIENTOS FUNCIONALES (DISEÑO)

Introducir la información por medio del amanuense
Visualizar la información en un reporte
Almacenar la información en una base de datos
Desarrollar usuarios con designación especifica
Modificar la información por parte del amanuense
Materializar el documento final con el reporte final de la orden de marcha

REQUERIMIENTOS NO FUNCIONALES (EVALUACIÓN)

Usabilidad: Mide la cantidad de esfuerzo que requiere un usuario para el uso del producto.

Eficiencia: Define el desempeño del producto en cuanto a tiempo de respuesta, número de operaciones por segundo como las consultas y actualizaciones permanentes que se podrán realizar.

Disponibilidad: Disposición que tiene el sistema.

Confiabilidad: Continuidad del servicio

Escalabilidad: Acción que permite la incorporación de nuevos requerimientos sin afectar el servicio.

Flexibilidad: Hace referencia a la administración del sistema y la dinámica que se debe tener en los datos estructurados.

Mantenibilidad: Cualidad que permite realizar modificaciones o reparaciones sin afectar el funcionamiento del sistema.

3.2.1.- CASOS DE USO

FLUJO DE INFORMACIÓN

PRIMER CASO: DESIGNACION DE SEGURIDAD PARA EL INGRESO DEL AMANUENCE AL SISTEMA DE CONTROL

DESCRIPCIÓN: En este caso Inicialmente realizaremos la creación de usuario con una identificación y una contraseña para que pueda acceder al sistema de control solo la persona autorizada.

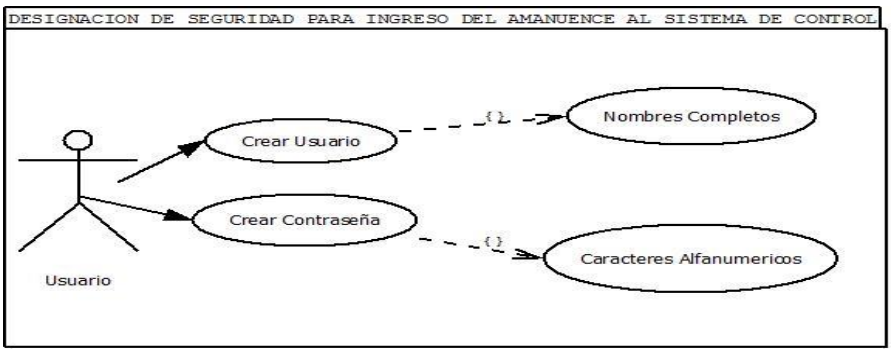


Grafico 1 Casos de uso – Usuario Login

Fuente: Propia
Elaborado por: Luis Lovato

FLUJO PRINCIPAL

El usuario crea un ID y una contraseña para que pueda acceder únicamente la persona autorizada.

FLUJO SECUNDARIO

La persona registrada para el ingreso al sistema de control deberá obligatoriamente ingresar sus datos personales, grado o función, así como su usuario y contraseña de acceso.

SEGUNDO CASO: REGISTRO DE INFORMACION GENERAL POR PARTE DEL AMANUENCE AL SISTEMA DE CONTROL.

DESCRIPCIÓN: El amanuense se registrara con su usuario y contraseña para ingresar al sistema de control y podrá hacer uso de las funciones principales para almacenamiento de la información general relacionada a conductores, vehículos, rutas para que finalmente emita un reporte.

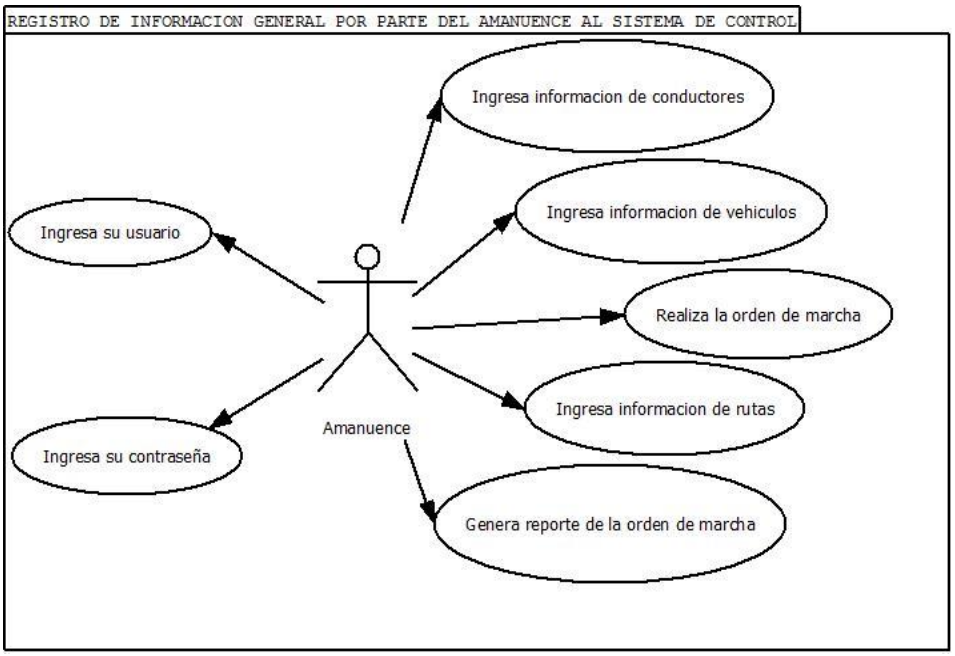


Grafico 2 Casos de uso – Registro información amanuense al sistema

Fuente: Propia
Elaborado por: Luis Lovato

FLUJO PRINCIPAL

El amanuense registrado ingresa su clave y contraseña para acceder al sistema de control.

El amanuense será en el encargado de ingresar toda la información relacionada a los conductores de la unidad.

El amanuense será en el encargado de ingresar toda la información relacionada a los vehículos existentes en la unidad.

El amanuense será el encargado de generar las órdenes de marcha que se requiera.

El amanuense será en el encargado de ingresar toda la información relacionada a las rutas establecidas para las comisiones vehiculares.

El amanuense será el encargado de generar el reporte de la orden de marcha realizada.

TERCER CASO: REGISTRO DE CONDUCTORES AL SISTEMA DE CONTROL.

DESCRIPCIÓN: El amanuense podrá hacer uso de las funciones principales, ingreso, registro sobre datos personales, jerarquías y categorías de licencia que ostenten cada conductor para que finalmente se pueda obtener un reporte.

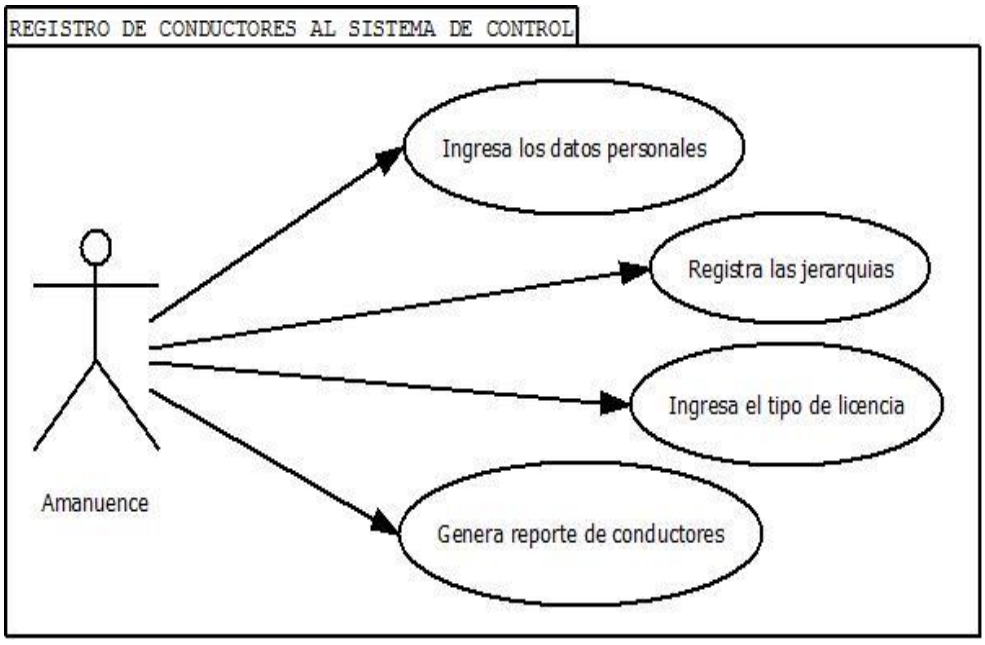


Grafico 3 Casos de uso – Registro de Conductores

Fuente: Propia
Elaborado por: Luis Lovato

FLUJO PRINCIPAL

El amanuense registrara los datos personales de cada conductor de la unidad.

El amanuense registrara las jerarquías de cada uno de los conductores.

El amanuense registrara los tipos de licencia que posean los conductores.

El amanuense será el encargado de generar un reporte de los conductores existentes en la unidad.

CUARTO CASO: REGISTRO DE VEHICULOS AL SISTEMA DE CONTROL.

DESCRIPCIÓN: El amanuense usara las funciones principales, ingresar, actualizar para poder almacenar datos técnicos, detalle de vehículos, kilometrajes para que finalmente pueda genera un reporte.

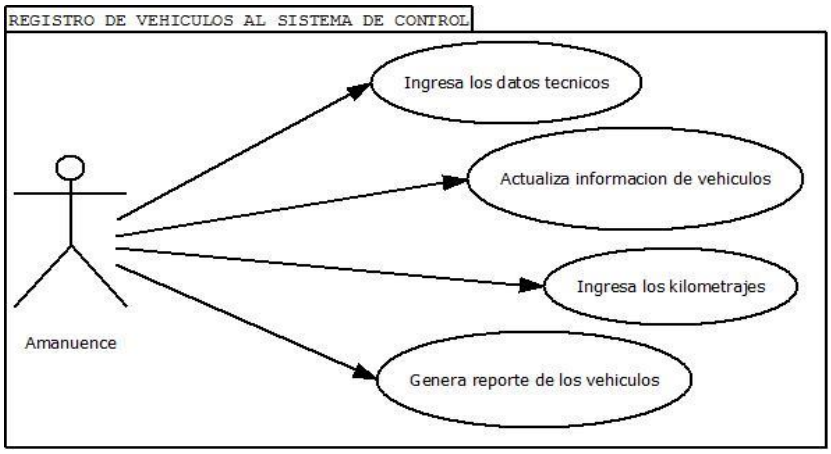


Grafico 4 Casos de uso – Registro de Vehiculos

Fuente: Propia
Elaborado por: Luis Lovato

FLUJO PRINCIPAL

El amanuense ingresara los datos técnicos de los vehículos existentes en la unidad.

El amanuense actualizara la información relacionada a los vehículos que nuevos o enviados al remate.

El amanuense registrara los kilometrajes de todos los vehículos de la unidad.

El amanuense generara un reporte de todos los vehículos existentes en la unidad.

QUINTO CASO: REGISTRO DE RUTAS EN EL SISTEMA DE CONTROL.

DESCRIPCIÓN: El amanuense usara las funciones principales para el ingreso y actualización de la información de las rutas así como algún detalle de aviso sobre las mismas.

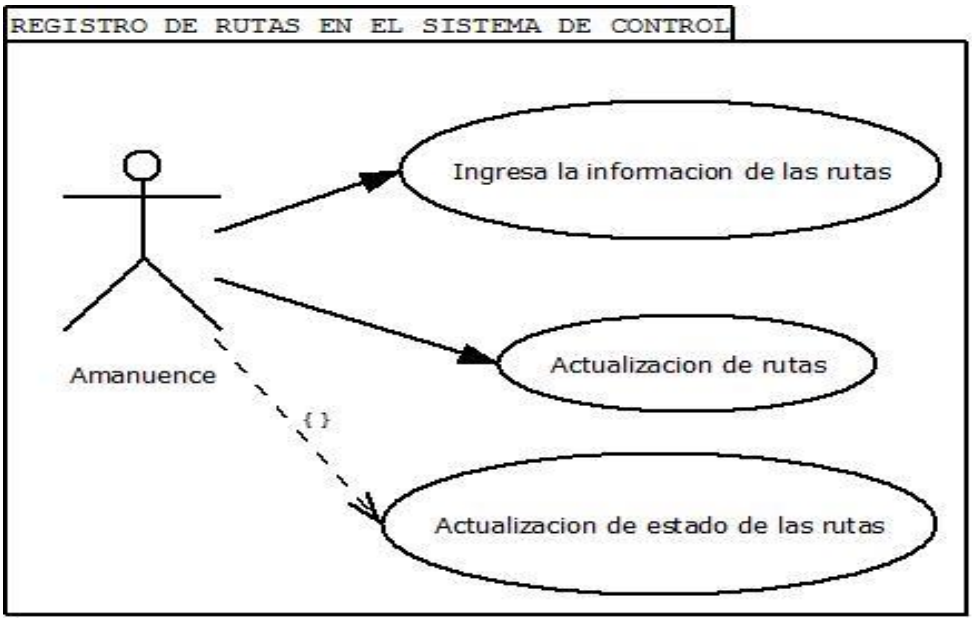


Grafico 5 Casos de uso – Registro de Rutas

Fuente: Propia
Elaborado por: Luis Lovato

FLUJO PRINCIPAL

El amanuense ingresara la información de todas las rutas establecidas para las comisiones vehiculares.

El amanuense se encargara de la actualización de nuevas rutas que dispongan para las comisiones vehiculares.

Actualizara cualquier inconveniente sobre nuevas rutas.

SEXTO CASO: GENERACION DE LA ORDEN DE MARCHA EN EL SISTEMA DE CONTROL

DESCRIPCIÓN: El amanuense podrá usar las funciones principales, ingresar, seleccionar, generar e imprimir para el almacenamiento de la información registrada así como la complementaria para finalmente obtener el reporte.

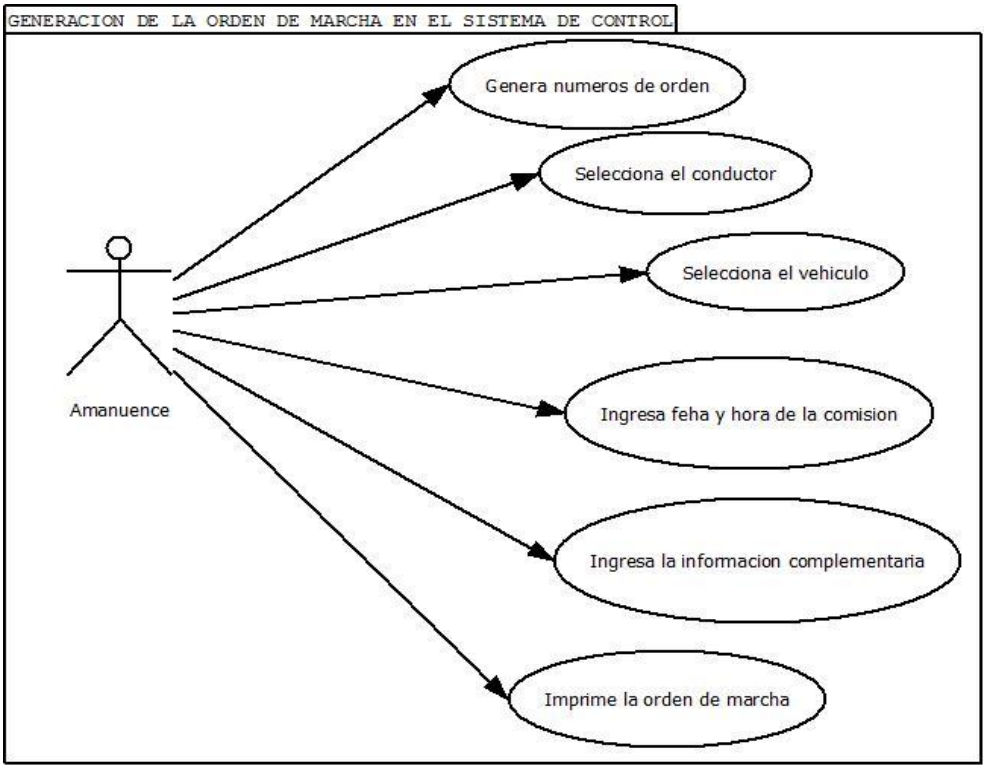


Grafico 6 Casos de uso – Generación orden de marcha

Fuente: Propia
Elaborado por: Luis Lovato

FLUJO PRINCIPAL

El amanuense genera los números de orden según se requiera.

El amanuense selecciona al conductor que sea designado para la comisión vehicular.

El amanuense selecciona el vehículo que sea designado para la comisión vehicular.

El amanuense ingresara los datos de fecha y hora del día que se realice la comisión vehicular.

El amanuense ingresara información complementaria de vital importancia para la orden de marcha.

El amanuense imprimirá la orden de marcha generada para la comisión vehicular.

3.1. DISEÑO

3.3.1. DISEÑO CONCEPTUAL

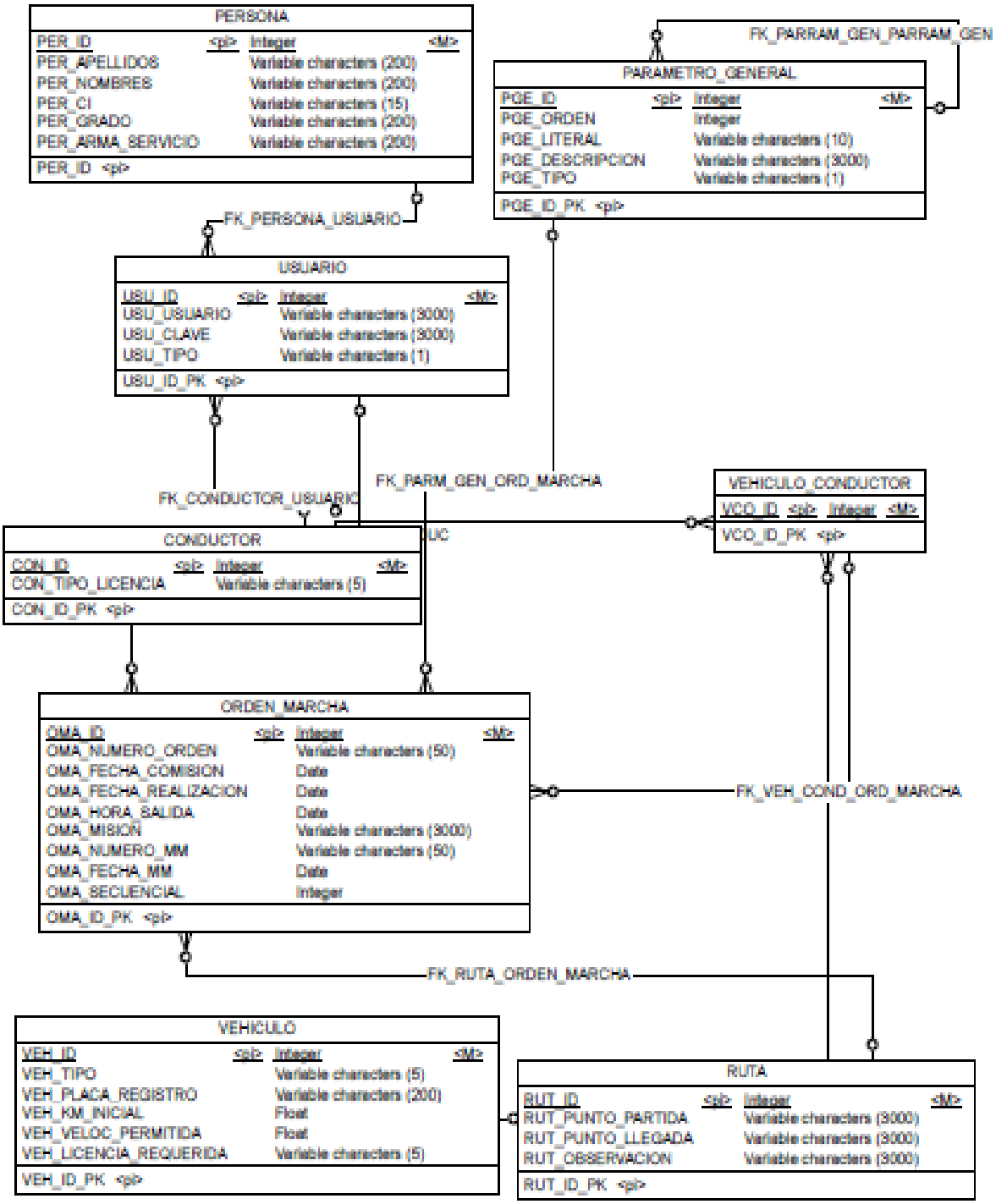


Grafico 7 Diseño Conceptual de la Base de Datos

Fuente: Propia
Elaborado por: Luis Lovato

3.3.2. MODELO RELACIONAL

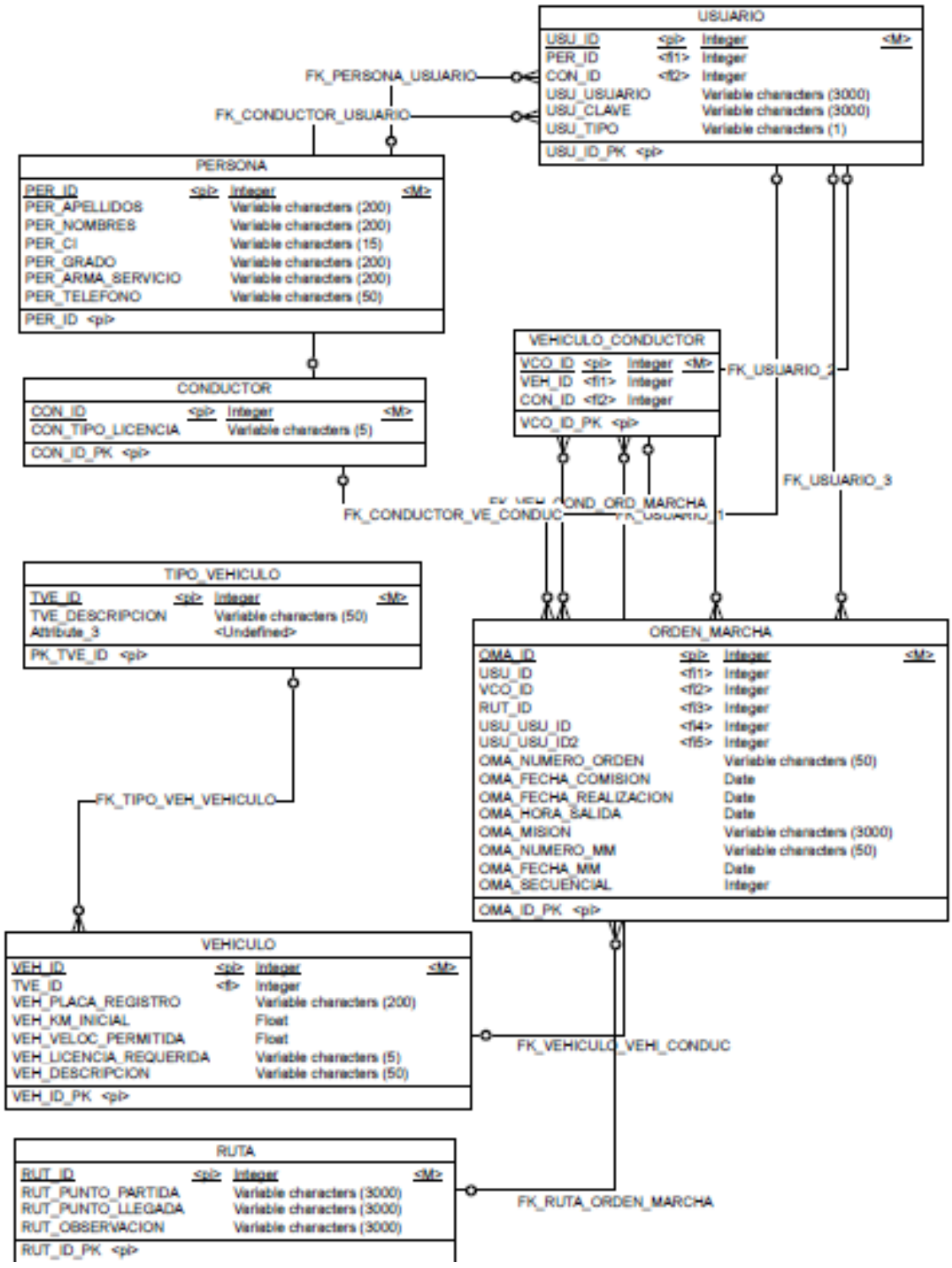


Grafico 8 Modelo Relacional de la Base de Datos

Fuente: Propia
Elaborado por: Luis Lovato

CAPÍTULO IV
IMPLEMENTACIÓN DEL SISTEMA

4.1.- ARQUITECTURA DEL SISTEMA

DISEÑO ARQUITECTÓNICO DEL SISTEMA ARQUITECTURA CLIENTE SERVIDOR

La estructura Cliente Servidor es una arquitectura de informática en la que se obtiene un procesamiento cooperativo de la información por medio de un conjunto de procesadores, de tal forma que uno o varios clientes, distribuidos solicitan servicios de información a uno o más servidores.

CLIENTE - SERVIDOR.

El cliente solicita una información al servidor.
El servidor recibe la petición del cliente.
El servidor procesa dicha solicitud.
El servidor envía el resultado obtenido al cliente.
El cliente recibe el resultado y lo procesa.

GRÁFICO DE LA ARQUITECTURA CLIENTE SERVIDOR PAR LA CREACIÓN DEL SISTEMA DE CONTROL PARA LAS COMISIONES VEHICULARES DEL CAL-11 "CALICUCHIMA"

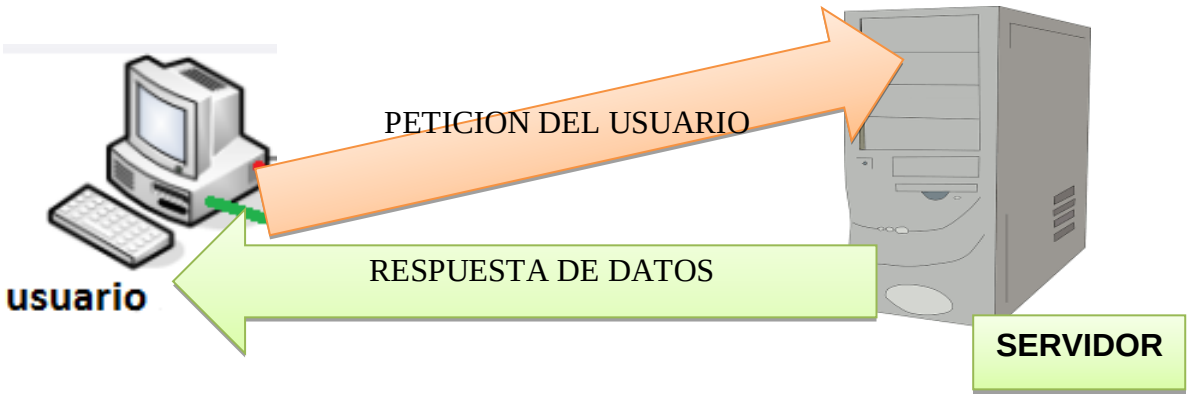


Grafico 9Arquitectura Cliente-Servidor

Fuente: Propia
Elaborado por: Luis Lovato

COMPONENTES DE LA ARQUITECTURA CLIENTE SERVIDOR

- Nivel de Presentación: Agrupa a todos los elementos asociados al componente Cliente
- Nivel de Aplicación: Agrupa a todos los elementos asociados al componente Servidor.
- Nivel de comunicación: Agrupa a todos los elementos que hacen posible la comunicación entre los componentes Cliente y servidor.
- Nivel de base de datos: Agrupa a todas las actividades asociadas al acceso de los datos.

ELEMENTOS PRINCIPALES QUE COMPONEN ESTA ARQUITECTURA CLIENTE

Comisión vehicular es el proceso que permite al usuario administrador(amanuense) formular los requerimientos y pasar al servidor y generar la orden de marcha.

- Se lo conoce con el término envió.
- Este puede acceder a los servicios distribuidos en cualquier parte del sistema de control.
- Las funciones que lleva a cabo el proceso comisión vehicular.
- El amanuense puede realizar lo siguiente:
- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica del sistema y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Actualizar resultados.

SERVIDOR

Es el proceso encargado de atender a un usuario administrador (amanuense) que hace peticiones de algún recurso administrado por él.

Al proceso servidor se lo conoce con el término respuesta de información.

El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas de las comisiones vehiculares y los recursos de datos para generar nuestra orden de marcha.

Las funciones que lleva a cabo el servidor:
Aceptar los requerimientos por parte del amanuense y además procesa información de la base de datos y envía reportes necesarios.
Procesa datos para trasmitirlos a los amanuenses.
Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

4.1.- PRUEBAS AL SISTEMA

En este aspecto se trata de probar los módulos que posee el sistema de control de comisiones vehiculares del CAL-11, con el fin de ejecutar el correcto funcionamiento del sistema tanto en código como en la entrega de información.

PRUEBAS DE CAJA NEGRA

Se llevan a cabo sobre la interfaz del software y pretenden demostrar que el software funciona adecuadamente; es decir que las entradas se aceptan de forma adecuada y se produce una salida correcta. Estas pruebas no tienen en cuenta la estructura lógica interna del software. (Normand, 2005, pág. 88)

Por tanto las técnicas de caja negra, permiten conocer la funcionabilidad del programa sin considerar su codificación, es decir, evalúa las entradas y salidas del sistema.

Módulos	Función a Probar	Resultados Esperados	Resultados Obtenidos
Usuario	Acceso a Módulos	Debe dirigir al usuario a los diferentes módulos de acuerdo a	Validación e ingreso correcto de usuario.

		sus privilegios.	
	Ingreso de Usuario	Mostrar una ventana en la cual permita registrar un máximo de 2 usuarios y modificar su contraseña.	Ingresó de datos correctos de los nuevos usuarios.
	Modificación de Usuario	Mostrará una ventana en la cual se permitirá hacer la modificación de los datos de los usuarios autorizados para su actualización.	Se actualiza y guarda los datos del usuario satisfactoriamente.
Vehículo	Ingreso de vehículos	Mostrará la ventana en la cual se ingresen los datos del nuevo vehículo para las comisiones vehiculares.	Ingresó de datos correctos del nuevo vehículo.
	Modificación de vehículo	Mostrar la ventana en la cual se visualizara el listado de los vehículos y permitirá la modificación de los datos para su control.	Se actualiza y almacena los datos del vehículo en dotación correctamente.
	Consulta de vehículo	Debe mostrar los datos de los vehículos ingresados como son: Placa, Tipo de Vehículo.	Reporte de todos los vehículos disponibles en la unidad.
	Eliminación de vehículo	Mostrar una ventana que permita eliminar vehículos que ya no	Se obtiene datos actualizados de vehículos

		están en la unidad.	pertenecientes a la unidad de manera satisfactoria.
Conductor	Ingreso de conductores	Mostrará la ventana en la cual se ingresen los datos de los conductores para las comisiones vehiculares.	Ingresó de datos correctos de los conductores existentes en la unidad.
	Modificación de conductores	Mostrar la ventana en la cual se visualizara el listado de los conductores y permitirá la modificación de los datos para su control.	Se actualiza y almacena los datos de los conductores correctamente.
	Consulta de conductores	Debe mostrar los datos de los conductores ingresados como son: Grado, Arma o Servicio, Nombres, Apellidos, Tipo de licencia.	Reporte de todos los conductores disponibles en la unidad.
	Eliminación de conductores	Mostrar una ventana que permita eliminar conductores que ya no están en la unidad.	Se obtiene datos actualizados de los conductores pertenecientes a la unidad de manera satisfactoria.
Orden de Marcha	Ingreso de la Orden de Marcha	Mostrará una ventana en la cual se ingresaran los datos de la comisión vehicular: ✓ Numero de orden ✓ Fecha, hora	Permite el ingreso y alamcenamiento satisfactorio de datos de la orden de marcha.

		✓ Numero de memo. ✓ Misión. ✓ Conductor. ✓ Vehículo. ✓ Ruta. ✓ Responsables. En vista que se pueden generar varias órdenes al día la numeración se realizara automáticamente.	
	Consulta de la orden de marcha	Desplegara un listado de todas las órdenes de marcha generadas por fecha y número de orden.	Mostrará satisfactoriamente de los datos de la orden de marcha con su respectivo reporte.
Reporte de la orden de marcha	Imprimir Reporte	Ya ingresados los datos y permitida la consulta permite imprimir las ordenes seleccionadas.	Imprimió exitosamente las ordenes de marcha seleccionadas.

Tabla 3 Prueba con caja negra

Fuente: Propia
Elaborador por: Luis Lovato

4.1.- CAPACITACIÓN AL PERSONAL

La capacitación al personal de amanuenses designados se lo realizara semestralmente previo el conocimiento de uso del sistema de manera presencial con un lapso de 4 horas por tres días, tanto personalmente con la verificación y tutoría de parte del administrador del sistema, esclareciendo así cualquier tipo de duda por parte del personal de usuarios del sistema de control y adicional se dejara el manual de uso e instalación del mismo en magnético y físico, acompañado del número de celular del administrador del sistema.

4.2.- MANTENIMIENTO

El mantenimiento que se le dará al sistema es de manera periódica trimestralmente, basándose en los complementos del computados como antivirus ya que en si el sistema, se basa en una razón de ser que es de control, posteriormente con el estudio realizado por parte de los usuarios se obtendrá recomendaciones de mejora, determinando nuevas necesidades y prioridades para optimizar de mejor manera el sistema, permitiendo generar u control estricto de las comisiones vehiculares.

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

- Java es un lenguaje de programación de código abierto compatible con cualquier SGBD uno de los más importantes es MYSQL, que facilita el desarrollo de sistemas informáticos.
- Las técnicas de recolección de datos ayudaron a la determinación de las necesidades indispensables que facilitaron el desarrollo del sistema para el Comando de Apoyo Logístico Nº 11.
- El sistema informático de control para las comisiones vehiculares utilizando lenguaje de programación JAVA y base de batos MYSQL permitieron economizar recursos económicos, garantizando su funcionamiento.

5.2. RECOMENDACIONES

Ya desarrollado el sistema de control de las comisiones vehiculares se recomienda a los usuarios:

- Dar a conocer a las instancias respetivas el beneficio del sistema para obtener la autorización del Escalón Superior para ejecutar la implementación y uso del sistema informático de control de comisiones vehiculares del CAL-11 con el fin de optimizar los recursos logísticos disponibles en el Ejército.
- Capacitar a los amanuenses encargados del sistema informático de control de comisiones vehiculares, que realicen el ingreso correcto de los datos de cada movimiento militar ejecutado.
- Manejar con seriedad el sistema informático con el fin de obtener búsqueda de información necesaria en el menor tiempo posible.

WEB BIBLIOGRAFÍA

http://www.alegsa.com.ar/Dic/sistema_informatico.php

BLOGSPOT.COM. (17 de ABR de 2014). Recuperado el 15 de FEB de 2017, de http://programemosenjava.blogspot.com/2014/04/ventajas-y-desventajas-de-java.html

IBM. (16 de MAR de 1998). Recuperado el 12 de FEB de 2017, de https://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava1/index.html

LAGSAM. (2002). *Libreria Informatica Programacion.* Santiago de Chile.

OCAMPO, I. F. (2009). *Tipos de Sistemas Informaticos.* CHILE.

OPENWEBINARS.NET. (12 de MAR de 2003). Recuperado el 14 de FEB de 2017, de https://openwebinars.net/blog/razones-para-aprender-programar-java/(ojo aquí es un tipo curso)

Pardo, O. C. (1999). *Introduccion a la Informatica.* Lima.

SCRIPTCASE. (15 de MAY de 1996). Recuperado el 18 de FEB de 2017, de http://www.scriptcase.net/?gclid=CjwKEAjwm7jKBRDE2_H_t8DVxzISJACwS9WbjVeocGp91-FtxAQWhGbJZTK4soHaIRw2RMJD1LcfohoCHBXw_wcB

Tecnologia Informatica. (1997). *Que son los sitemas Informaticos.* Paraguay.

YEDIYAH. (2000). *Libreria Informatica tipos de bases de datos.* Argentina.

ANEXOS

1. **Manual de Usuario**

MANUAL DE USUARIO DEL SISTEMA INFORMATICO

SISTEMA DE CONTROL DE COMISIONES VEHICULARES

INTRODUCCIÓN

Este sistema informático fue diseñado de manera amigable para el control de comisiones vehiculares del Comando de Apoyo Logístico nº 11 "CALICUCHIMA", aquel que nos brindara ingresar información así como modificarla con el fin de generar un reporte final de la orden de marcha, documento que servirá como salvoconducto para la autorización de movimientos vehiculares.

Seguidamente se detallara paso a paso el uso del sistema de control con el fin de optimizar este proceso.

HERRAMINETAS DE DESARROLLO DEL SISTEMA

Es indispensable tener en cuenta los siguientes puntos:

Los requisitos previos de software instalado para poder ejecutar el sistema serian:

1. **Gestor de Base de datos:** MYSQL actualizada
2. **Lenguaje de programación:** JAVA
3. **Herramientas de diseño:** JDK(JAVA)
4. **IDE:** NetBeans Versión 8.2
5. **Reportes:** Plugins Ireport para Netbeans
6. **Servidor de aplicación:** Servidor XAMPP Versión 3.2.2

Requisitos hardware:

1. Microsoft Windows 7.
2. Laptop HP core I5 32 bits RAM 4GB Disco Duro 640 GB.

En el presente manual se detalla paso a paso el manejo del sistema de control de comisiones vehiculares con el fin de causar un impacto amigable ante el usuario quien será el responsable junto a un ayudante del manejo de este sistema, a continuación la descripción secuencial de uso del sistema informático de control:

PASO Nº 1

ENCENDEMOS EL XAMPP CONTROL Y LO PONEMOS A CORRER

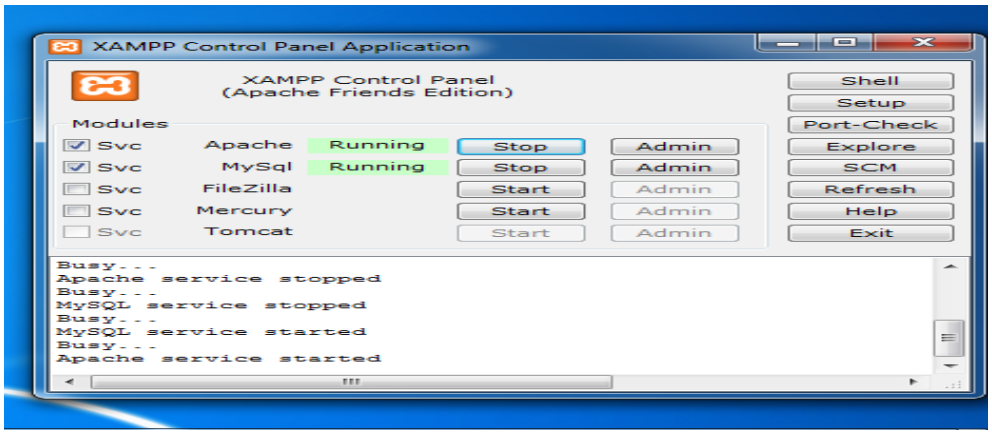


Figura 1 Pantalla de activación Xampp Control para la base de datos

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 2

EJECUTAMOS EL ACCESO DIRECTO DEL SISTEMA DE CONTROL ORDEN DE MARCHA

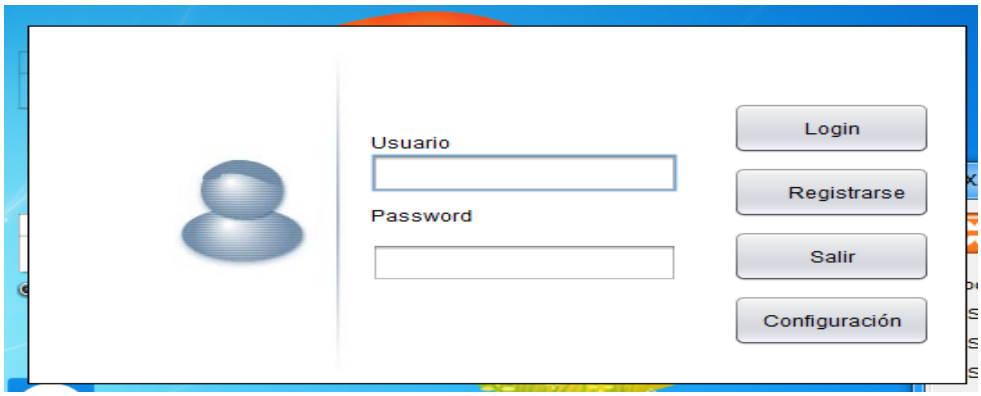


Figura 2 Pantalla de inicio del sistema de control

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 3

REGISTRAMOS A LAS DOS PERSONAS RESPONSABLES DEL SISTEMA TANTO PRINCIPAL COMO SECUNDARIO EN LA OPCION REGISTRARSE

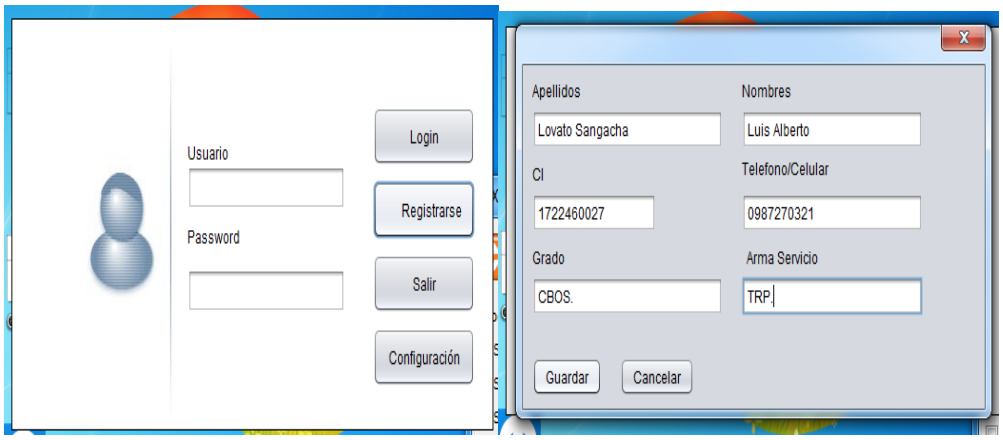


Figura 3 Pantallas de registro de usuarios responsables del sistema de control

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 4

INGRESAMOS EL USUARIO Y PASSWORD MODIFICADO E INGRESAMOS

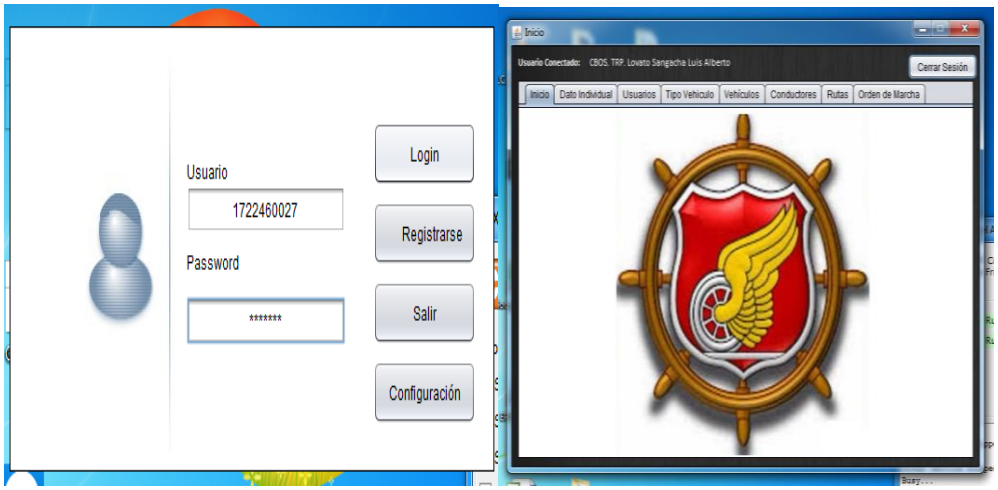


Figura 4 Pantallas de ingreso al sistema de control

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 5

INGRESAMOS EN LA TABLA USUARIOS TODO EL PERSONAL DE TRP., EXISTENTE EN LA UNIDAD MILITAR DE ACUERDO A SU JERARQUIA.

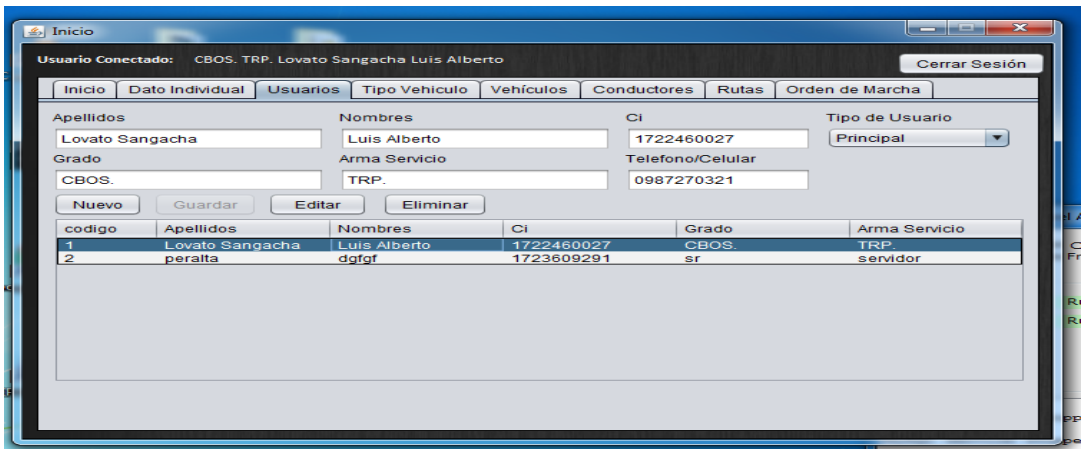


Figura 5 Pantalla de registro de usuarios de personal de la Unidad Militar

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 6

INGRESAMOS TODOS LOS TIPOS DE VEHICULOS EXISTENTES EN LA UNIDAD MILITAR

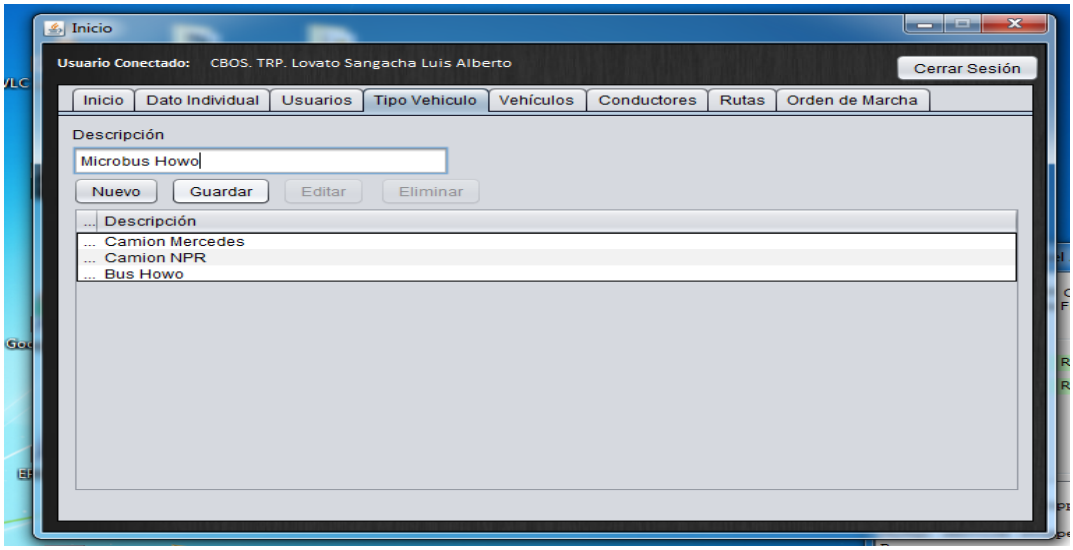


Figura 6 Pantalla de ingreso de tipos de vehículos de la Unidad Militar

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 7

INGRESAMOS TODOS LOS DATOS TECNICOS DE CADA VEHICULO EXISTENTE EN LA UNIDAD, ASI COMO SE PUEDE GENERAR UN REPORTE DEL MISMO

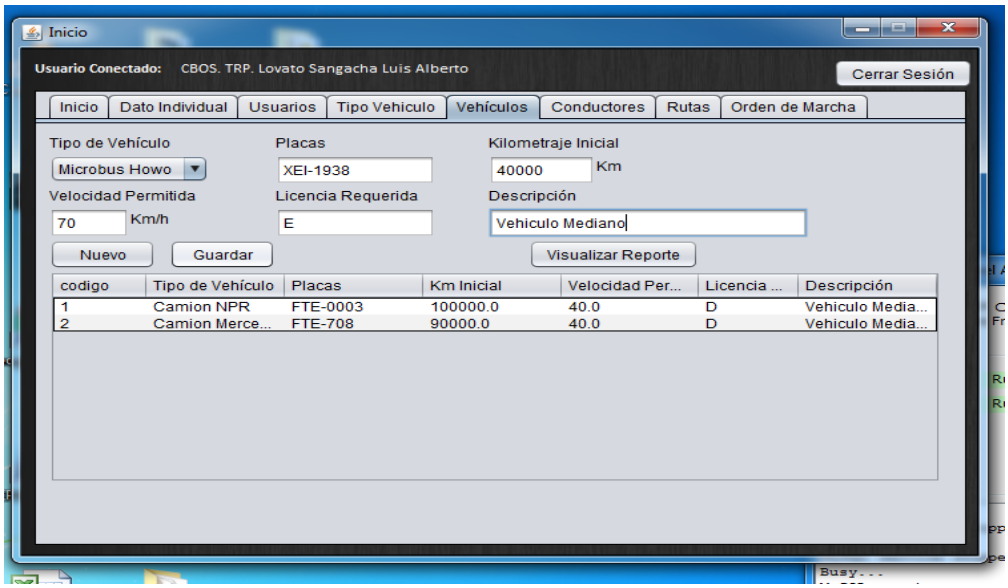


Figura 7 Pantalla de ingreso de vehículos de la Unidad Militar

Fuente: Propia
Elaborado por: Luis Lovato

Figura 8 Pantalla del reporte generado de los vehículos

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 8

INGRESAMOS EL PERSONAL DE CONDUCTORES CON SUS RESPECTIVOS VEHICULOS ASIGNADOS PARA LAS COMISOINES VEHICULARES CON SU REPECTIVO REPORTE

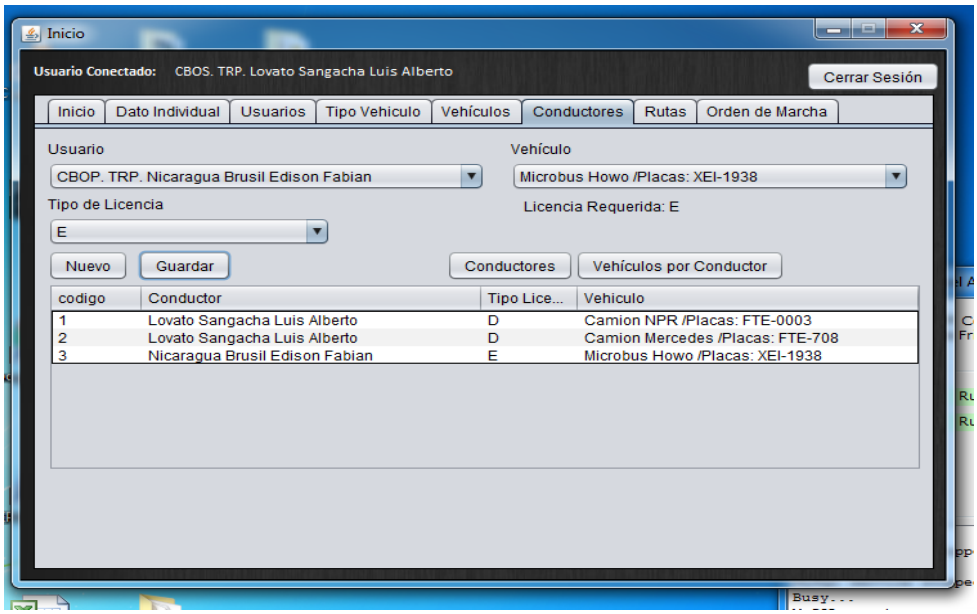


Figura 9 Pantalla de ingreso de conductores de la Unidad Militar

Fuente: Propia
Elaborado por: Luis Lovato

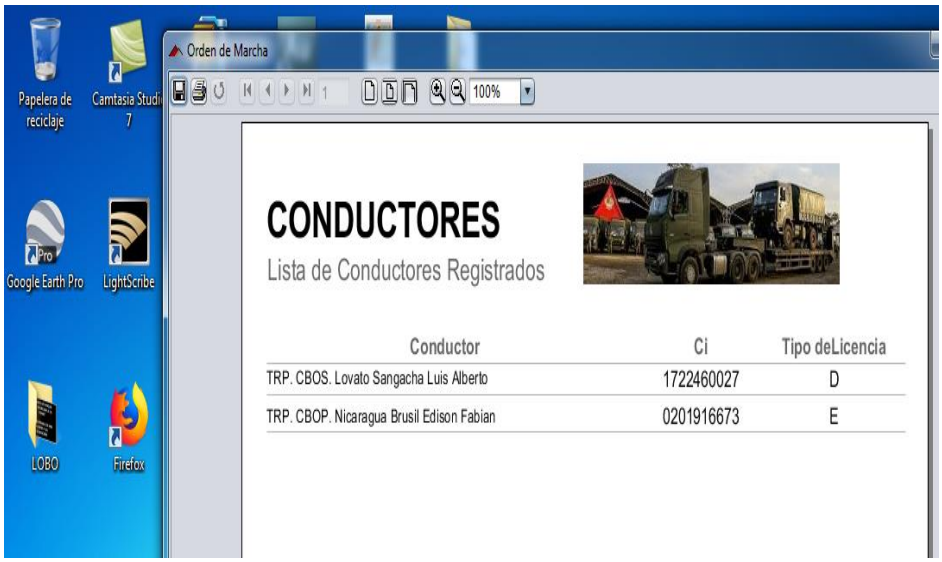


Figura 10 Pantalla del reporte generado de los conductores

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 9

INGRESAMOS LAS RUTAS ESTABLECIDAS PARA LAS COMISIONES VEHICULARES

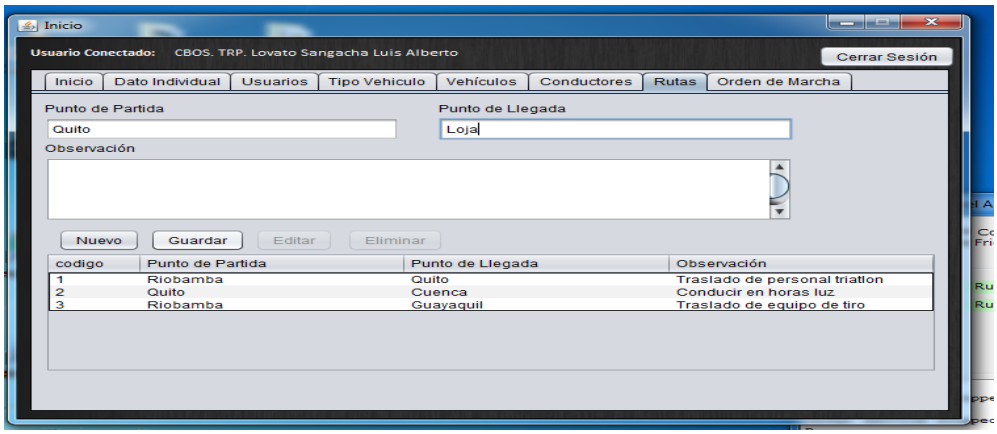


Figura 11 Pantalla de ingreso de rutas para las comisiones vehiculares

Fuente: Propia
Elaborado por: Luis Lovato

PASO Nº 10

FINALMENTE LA CREACION DE LA ORDEN DE MARCHA DE LA COMISION VEHICULAR ENCOMENDADA Y ALMACENAMIENTO DE LA MISMA CON SU RESPECTIVO REPORTE

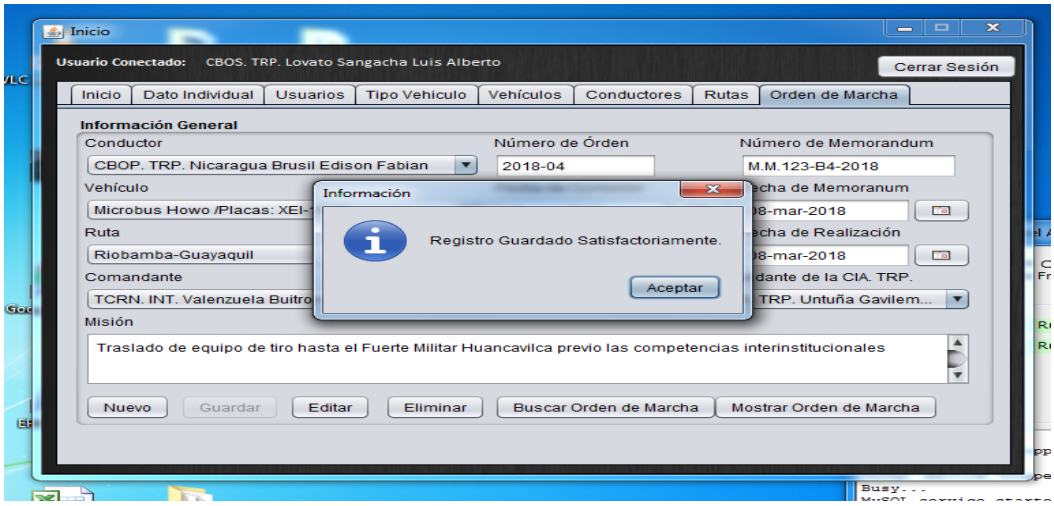


Figura 12 Pantalla de creación de la orden de marcha

Fuente: Propia
Elaborado por: Luis Lovato

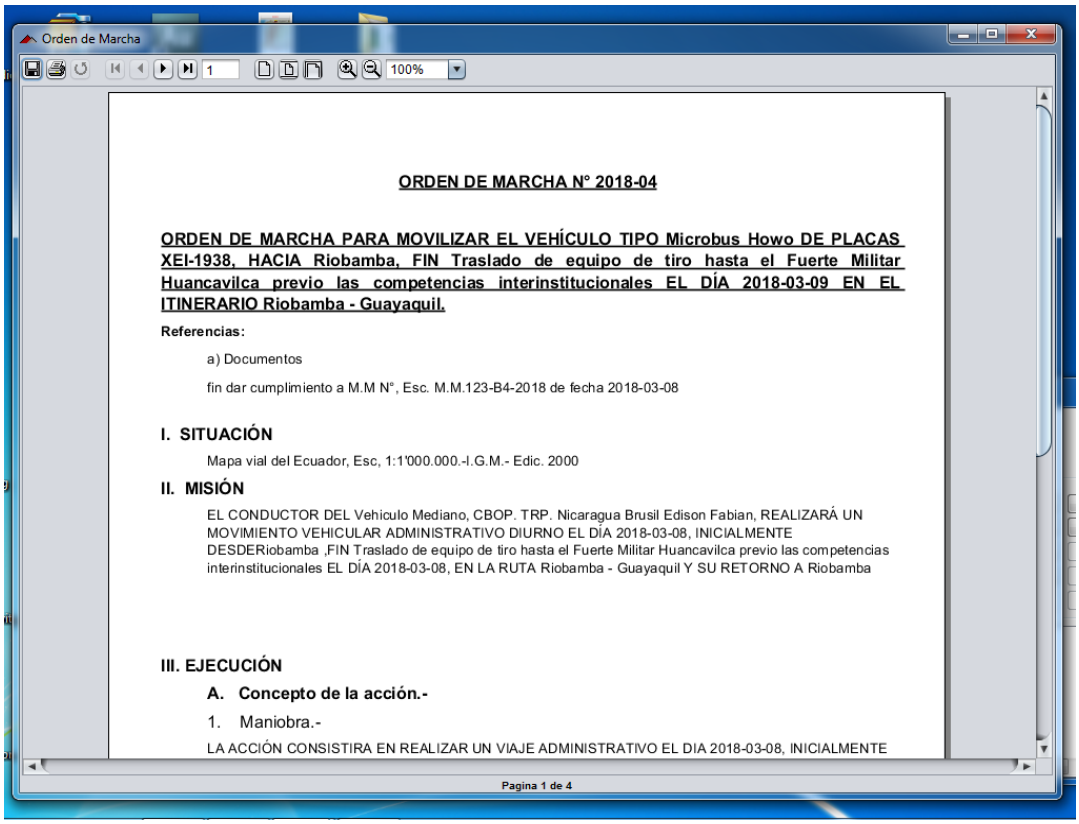


Figura 13 Pantalla del reporte final de la orden de marcha

Fuente: Propia
Elaborado por: Luis Lovato

Como podemos observar se cumplio con el objetivo de generar la orden de marcha con sus respectivos respaldo en la base de datos y adicional reduciendo el tiempo de demora en la generacion de la misma.

2. Manual Técnico

MANUAL TÉCNICO DEL SISTEMA INFORMATICO

SISTEMA DE CONTROL DE COMISIONES VEHICULARES

INTRODUCCIÓN

Este sistema informático fue diseñado para el control de comisiones vehiculares del Comando de Apoyo Logístico nº 11 "CALICUCHIMA", mismo que permitirá agilizar órdenes de marcha con sus respectivos respaldos almacenados de forma segura en la base de datos.

Seguidamente se detallara ciertas especificaciones técnicas que se debe tener en cuenta para el óptimo funcionamiento del sistema de control.

REQUERIMIENTOS DEL SISTEMA

Para el desarrollo del presente sistema de control es necesario contar con los siguientes requisitos, tanto hardware como software.

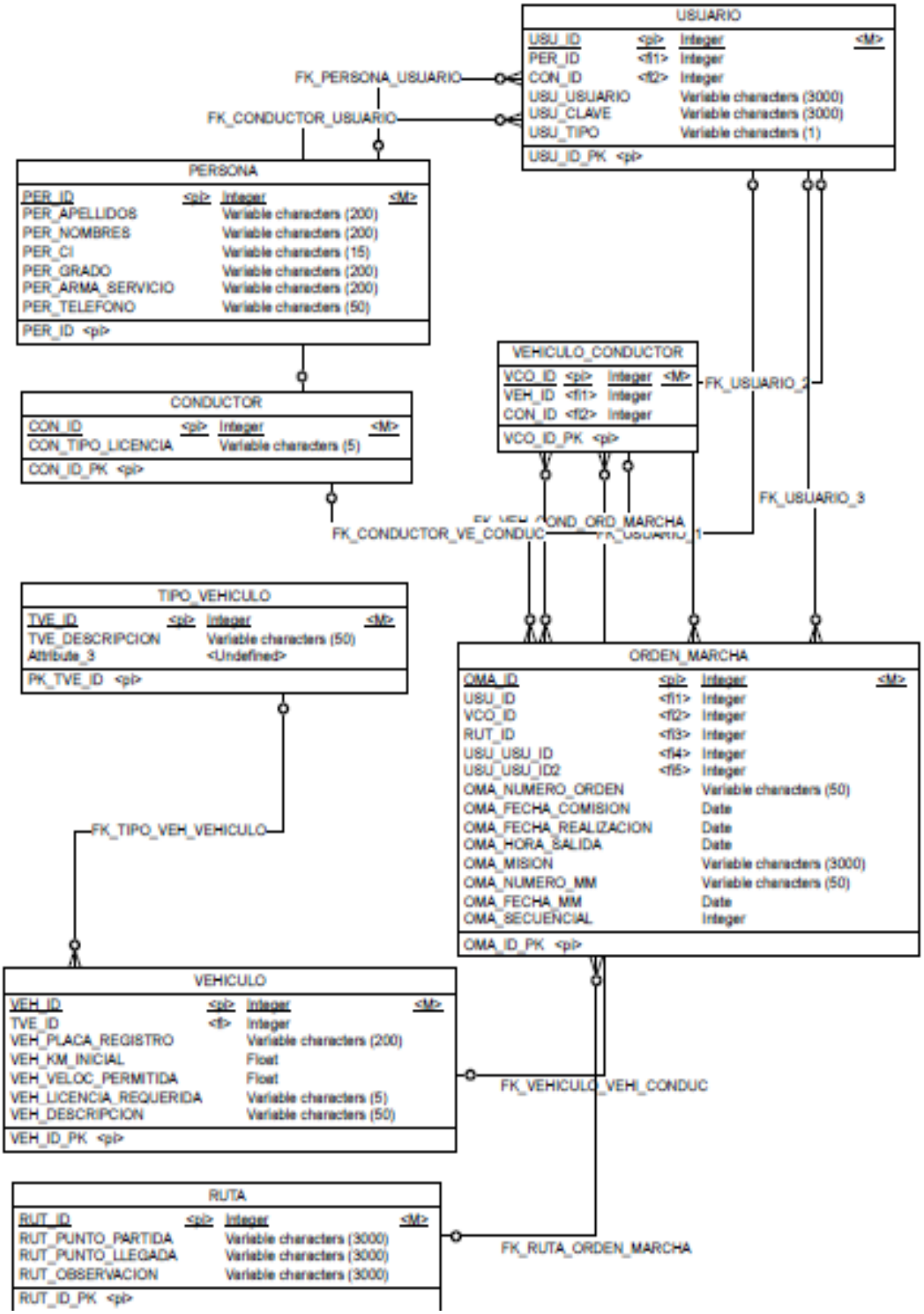
Los requisitos previos de software instalado para poder ejecutar el sistema serian:

7. **Gestor de Base de datos:** MYSQL actualizada
8. **Lenguaje de programación:** JAVA
9. **Herramientas de diseño:** JDK(JAVA)
10. **IDE:** NetBeans Versión 8.2
11. **Reportes:** Plugins Ireport para Netbeans
12. **Servidor de aplicación:** Servidor XAMPP Versión 3.2.2

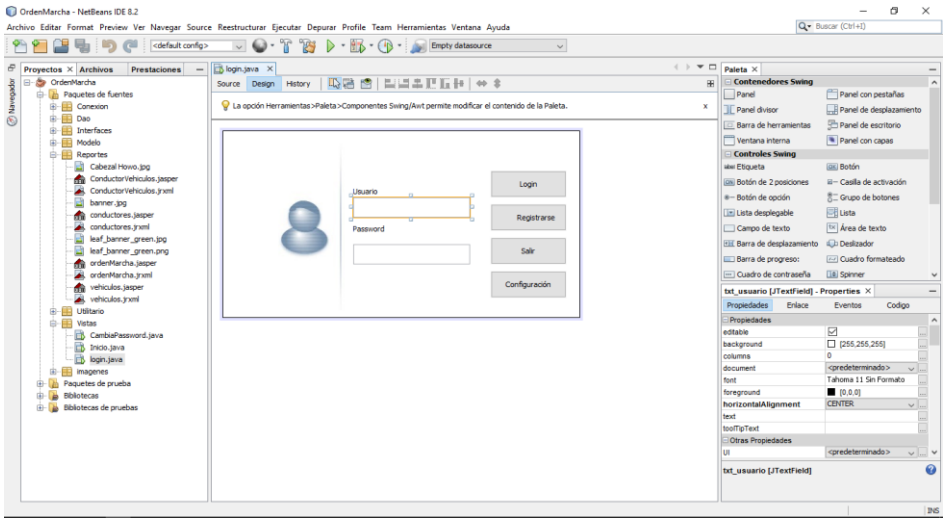
Requisitos hardware:

3. Microsoft Windows 7.
4. Laptop HP core I5 32 bits RAM 4GB Disco Duro 640 GB.

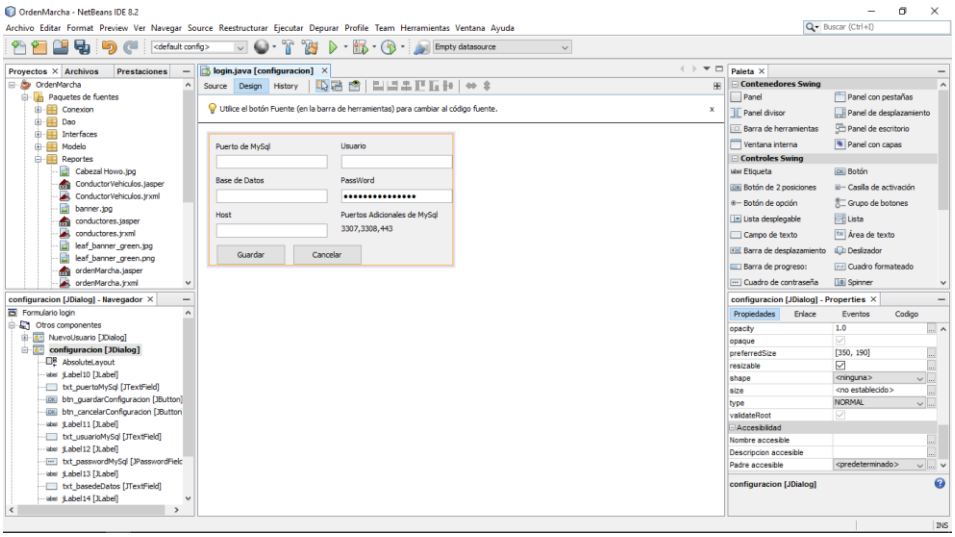
MODELO DE LA BASE DE DATOS DEL SISTEMA DE CONTROL



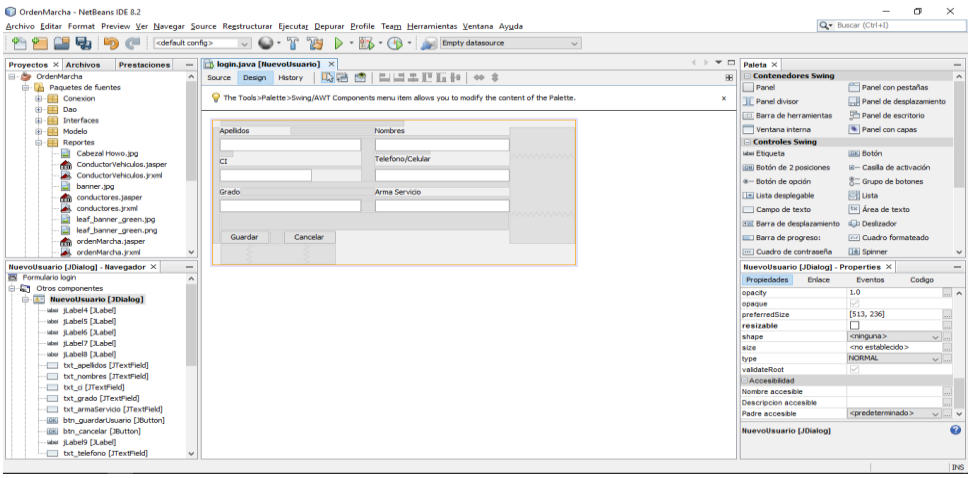
Proceso de creacion



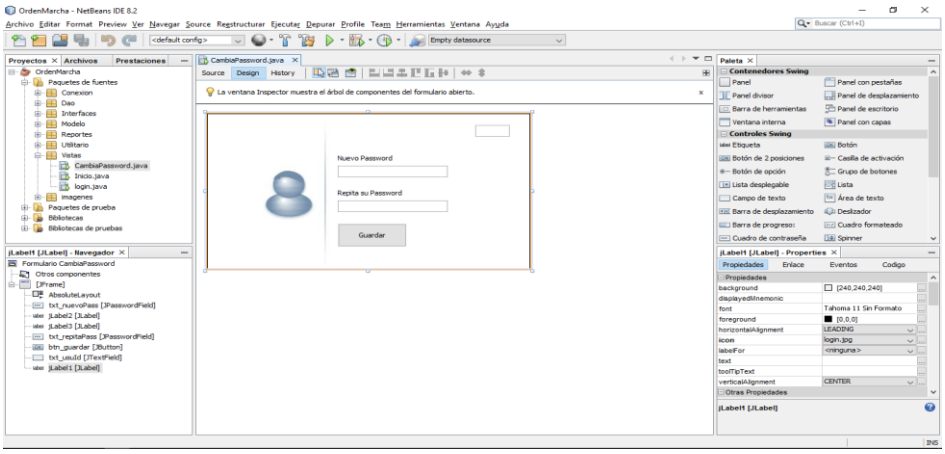
Diseño de la interface de login.



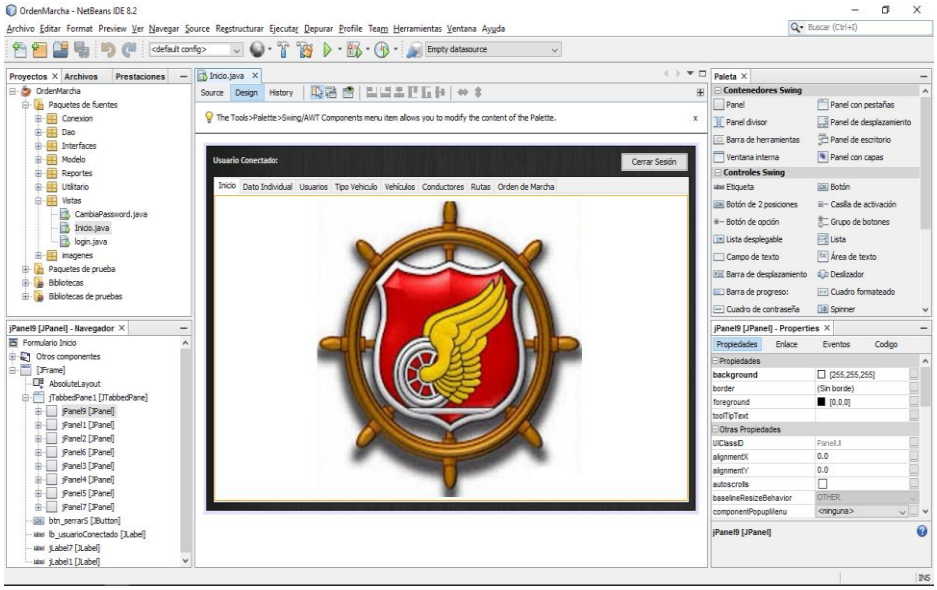
Diseño de la interface de configuración, conexión a la base de datos.



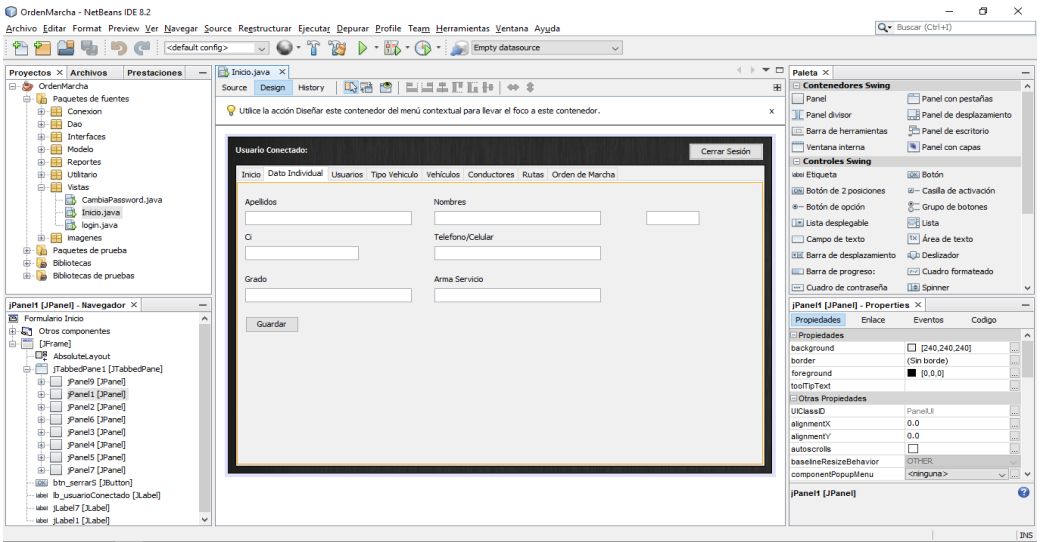
Diseño de la interface para crear un nuevo usuario.



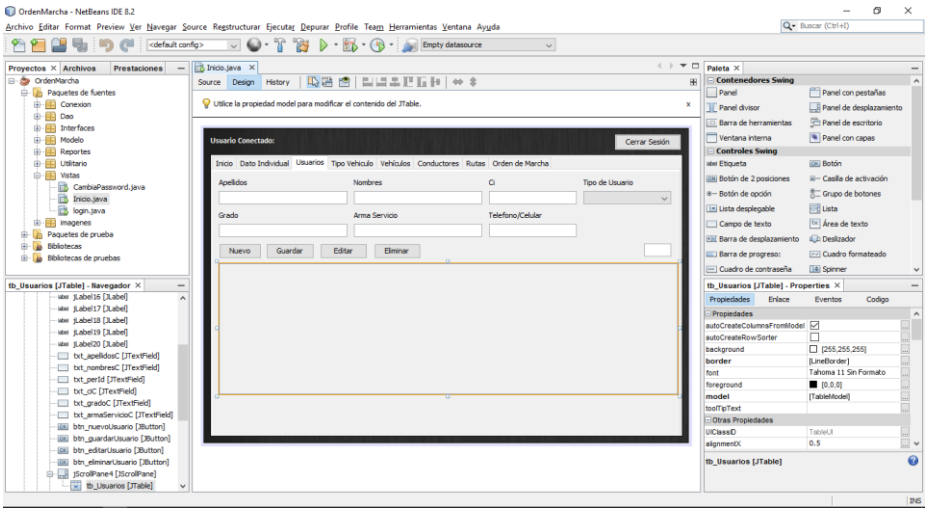
Diseño de interface de cambiar la contraseña.



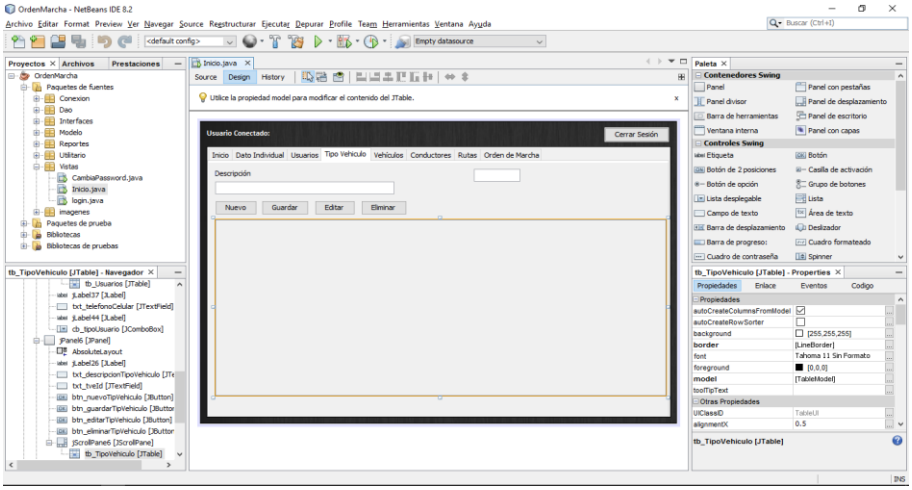
Interface de inicio al sistema.



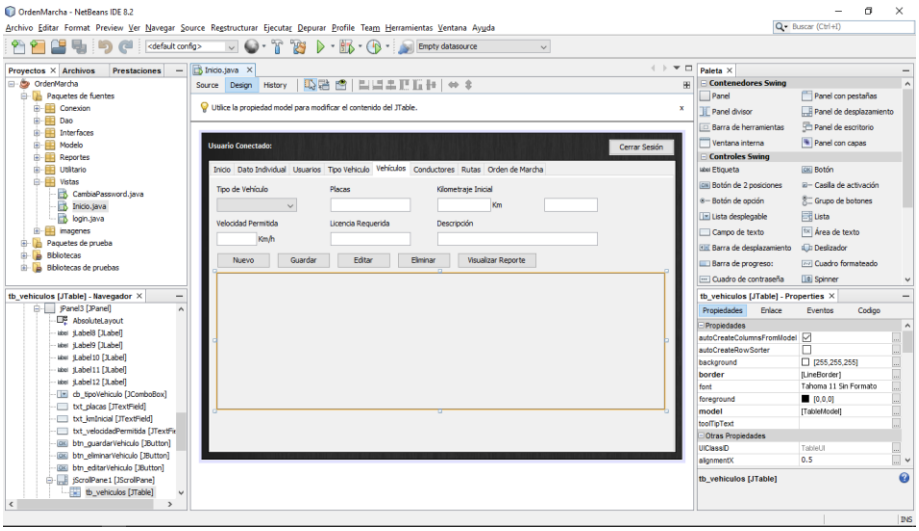
Interface de datos individual del usuario conectado.



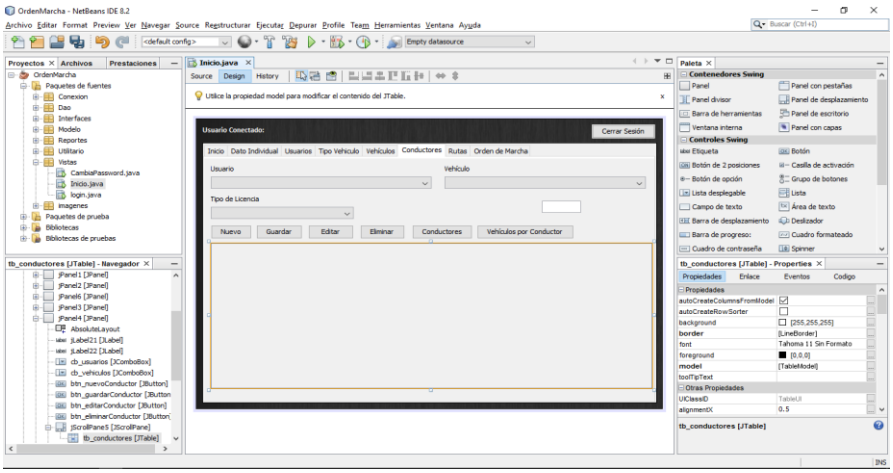
Interface para administrar usuarios del sistema.



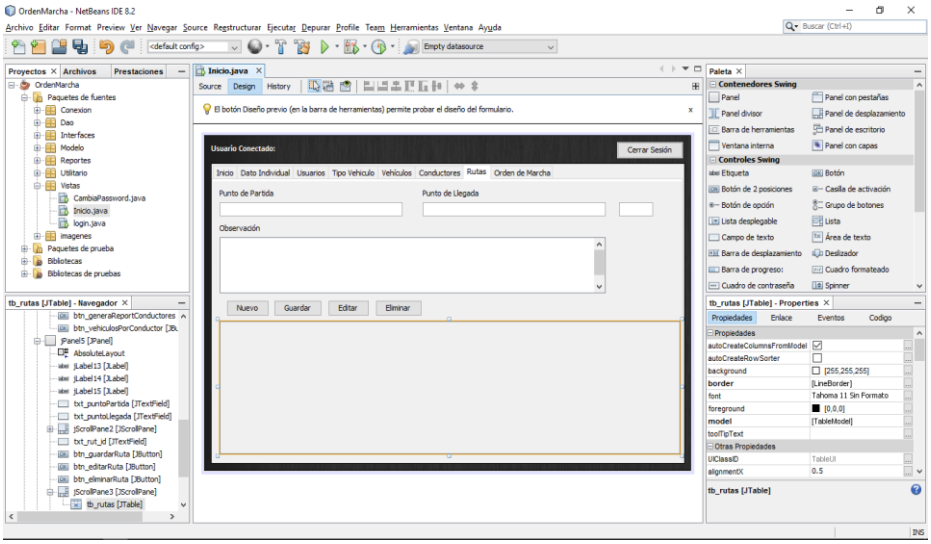
Interface para administrar tipos de vehículos.



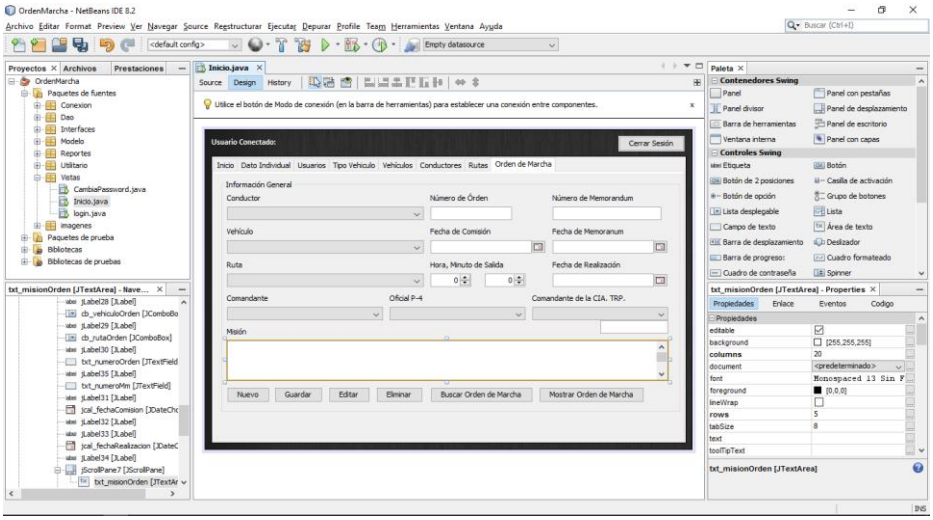
Interface para administrar los diferentes vehículos.



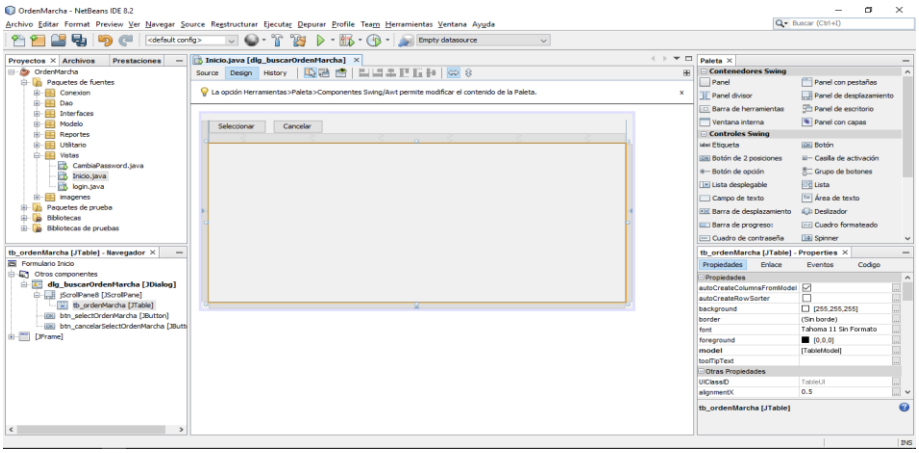
Interface para administrar los diferentes conductores.



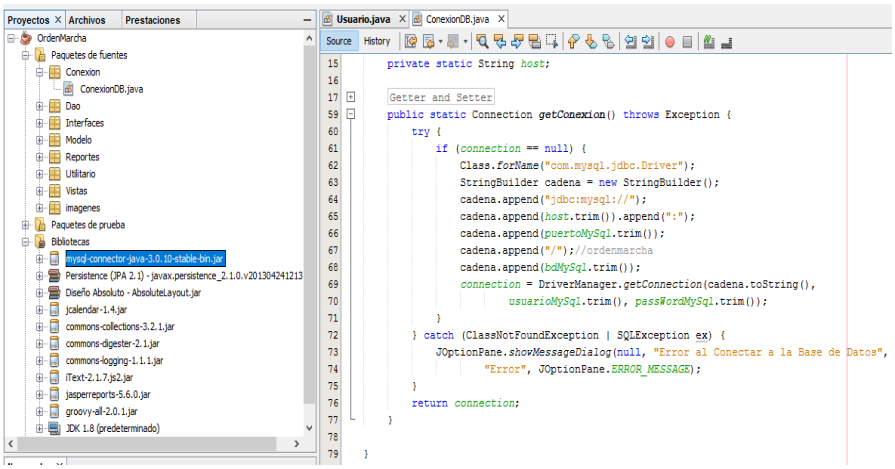
Interface para administrar las diferentes rutas



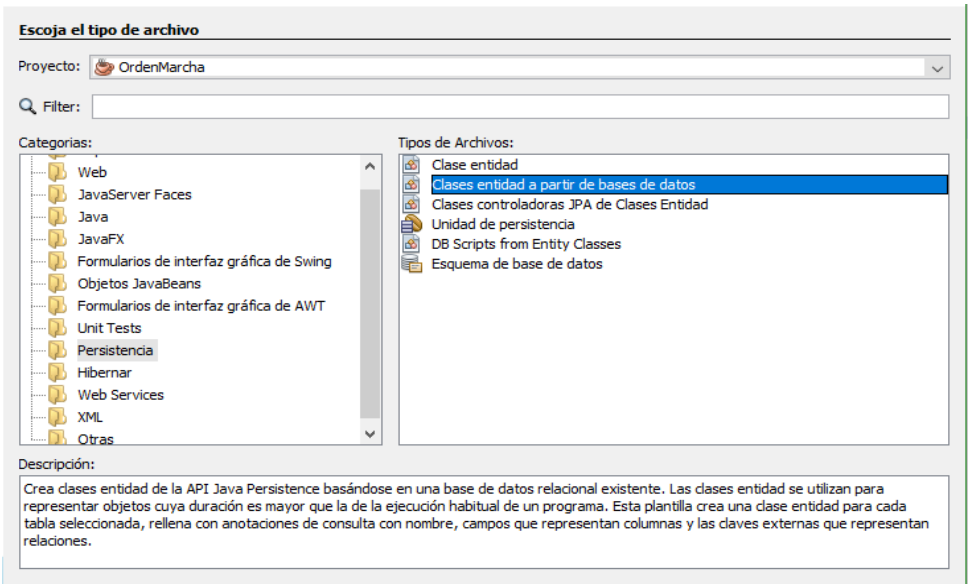
Interface para administrar las diferentes órdenes de marcha.



Dialogo para buscar las diferentes órdenes registras anteriormente en el sistema.



Método para gestionar la conexión a la base de datos mediante jdbc para MySql en su versión estable 3.0.10.

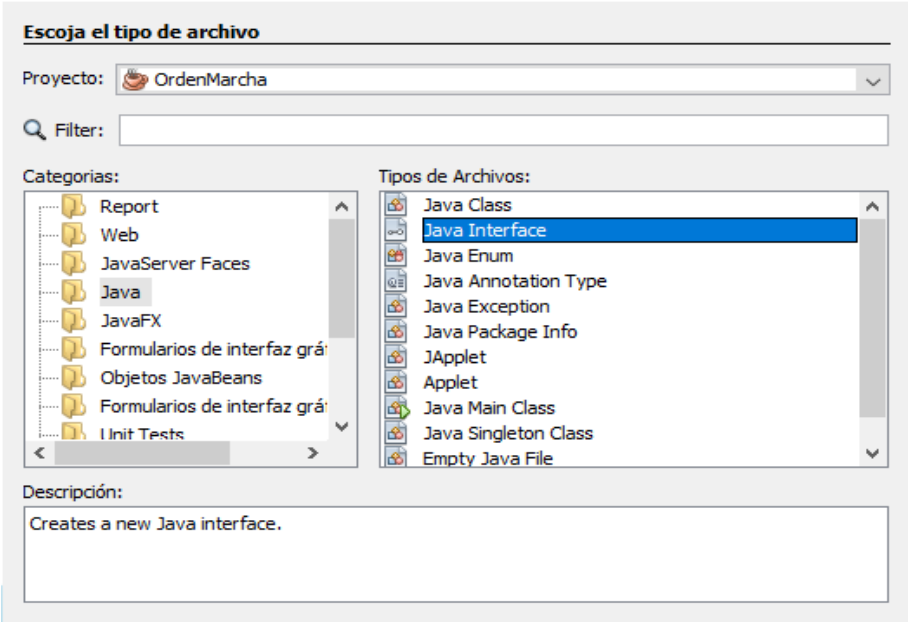


Pantalla de netbeans para generar las clases a partir de un modelo de base de datos, en persistencia.

```java
@Entity
@Table(name = "usuario")
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "USU_ID")
    private Integer usuId;
    @Column(name = "USU_USUARIO")
    private String usuUsuario;
    @Column(name = "USU_CLAVE")
    private String usuClave;
    @Column(name = "USU_TIPO")
    private String usuTipo;
    @JoinColumn(name = "CON_ID", referencedColumnName = "CON_ID")
    @ManyToOne
    private Conductor conductor;
    @JoinColumn(name = "PER_ID", referencedColumnName = "PER_ID")
    @ManyToOne
    private Persona persona;
    @OneToMany(mappedBy = "usuario")
    private List<OrdenMarcha> ordenMarchaList;
    @OneToMany(mappedBy = "usuario1")
    private List<OrdenMarcha> ordenMarchaList1;
    @OneToMany(mappedBy = "usuario2")
    private List<OrdenMarcha> ordenMarchaList2;
    @Column(name = "USU_ACTUALIZA")
    private String usuActualiza;
```

Clase Entidad USUARIO, generada a partir de un modelo de base de datos.



Pantalla de netbeans para generar una interface de java para administrar la clase entidad de usuario

```java
package Interfaces;

import Modelo.Usuario;
import java.util.List;

public interface UsuarioInterface {

    Usuario create(final Usuario object);

    boolean update(final Usuario object);

    boolean delete(final Usuario object);

    Usuario getById(final Integer id);

    Usuario getMax();

    List<Usuario> listByParameter(final Usuario object);

    Usuario inicializarObject();

    Usuario getByPerId(final Integer id);

    Usuario getByConId(final Integer id);

    List<Usuario> listConductores();

    Usuario validaUsuario(final String usuario, final String clave);

    List<Usuario> listByTipos(final String tipo);
}
```

Interface de java, con los diferentes métodos necesarios para administrar los diferentes usuarios del sistema.

```java
package Dao;

import Conexion.ConexionDB;
import Interfaces.UsuarioInterface;
import Modelo.Usuario;
import Utilitario.Utilitario;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;

public class UsuarioDao implements UsuarioInterface {

    private final Utilitario utilitario = new Utilitario();
    private final PersonaDao personaDao = new PersonaDao();
    private final ConductorDao conductorDao = new ConductorDao();
    private ResultSet resultSet = null;
```

Clase java de implementación "UsuarioDao" de la interface de la clase entidad de USUARIO.

```java
@Override
public Usuario create(Usuario object) {
    if (object != null) {
        try {
            StringBuilder sql = new StringBuilder();
            sql.append("INSERT INTO usuario");
            sql.append(" VALUES ( null");
            sql.append(object.getPersona() != null ? " , " + object.getPersona().getPerId() : " , null");
            sql.append(object.getConductor() != null ? " , " + object.getConductor().getConId() : " , null");
            sql.append(" , ? , ? , ? , ? )");
            PreparedStatement ps = ConexionDB.getConexion().prepareStatement(sql.toString());
            ps.setString(1, (object.getUsuUsuario() != null ? object.getUsuUsuario() : ""));
            ps.setString(2, (object.getUsuClave() != null ? object.getUsuClave() : ""));
            ps.setString(3, (object.getUsuTipo() != null ? object.getUsuTipo() : ""));
            ps.setString(4, (object.getUsuActualiza() != null ? object.getUsuActualiza() : ""));
            ps.execute();
            return getMax();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            return null;
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            return null;
        }
    } else {
        return null;
    }
}
```

Método de implementación de la interface de "UsuarioInterface" para crear usuarios del sistema.

```java
@Override
public boolean update(Usuario object) {
    if (object != null && object.getUsuId() != null) {
        try {
            StringBuilder sql = new StringBuilder();
            sql.append("UPDATE usuario");
            sql.append(" SET PER_ID = ").append(object.getPersona() != null ? object.getPersona().getPerId()
                    : " null ");
            sql.append(", CON_ID = ").append(object.getConductor() != null ? object.getConductor().getConId()
                    : " null ");
            sql.append(", USU_USUARIO = ? , USU_CLAVE = ? , USU_TIPO = ? , USU_ACTUALIZA = ? ");
            sql.append(" WHERE USU_ID = ").append(object.getUsuId());
            PreparedStatement ps = ConexionDB.getConexion().prepareStatement(sql.toString());
            ps.setString(1, (object.getUsuUsuario() != null ? object.getUsuUsuario() : ""));
            ps.setString(2, (object.getUsuClave() != null ? object.getUsuClave() : ""));
            ps.setString(3, (object.getUsuTipo() != null ? object.getUsuTipo() : ""));
            ps.setString(4, (object.getUsuActualiza() != null ? object.getUsuActualiza() : ""));
            ps.executeUpdate();
            return true;
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }
    } else {
        return false;
    }
}
```

Método de implementación de la interface de "UsuarioInterface" para editar información de los diferentes usuarios.

```java
@Override
public boolean delete(Usuario object) {
    if (object != null && object.getUsuId() != null) {
        try {
            StringBuilder sql = new StringBuilder();
            sql.append("DELETE FROM usuario WHERE USU_ID = ").append(object.getUsuId());
            ConexionDB.getConexion().createStatement().execute(sql.toString());
            return true;
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }
    } else {
        return false;
    }
}
```

Método de implementación de la interface de "UsuarioInterface" para eliminar un usuario por de id del registro.

```java
@Override
public Usuario getById(Integer id) {
    Usuario object = null;
    if (id != null) {
        StringBuilder sql = new StringBuilder();
        sql.append("SELECT * FROM usuario WHERE USU_ID = ").append(id);
        try {
            resultSet = ConexionDB.getConexion().createStatement().executeQuery(sql.toString());
            while (resultSet.next()) {
                object = inicializarObject();
                object.setUsuId(new Integer(resultSet.getString(1)));
                if (!utilitario.esnulo(resultSet.getString(2))) {
                    object.setPersona(personaDao.getById(new Integer(resultSet.getString(2))));
                }
                if (!utilitario.esnulo(resultSet.getString(3))) {
                    object.setConductor(conductorDao.getById(new Integer(resultSet.getString(3))));
                }
                object.setUsuUsuario(resultSet.getString(4));
                object.setUsuClave(resultSet.getString(5));
                object.setUsuTipo(resultSet.getString(6));
                object.setUsuActualiza(resultSet.getString(7));
            }
            resultSet.close();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    return object;
}
```

Método de implementación de la interface de "UsuarioInterface" para obtener la información de un usuario por el id del registro.

```java
@Override
public Usuario getMax() {
    Usuario object = null;
    StringBuilder sql = new StringBuilder();
    sql.append("SELECT * FROM usuario i");
    sql.append(" WHERE I.USU_ID IN(SELECT MAX(USU_ID) FROM usuario)");
    try {
        resultSet = ConexionDB.getConexion().createStatement().executeQuery(sql.toString());
        while (resultSet.next()) {
            object = inicializarObject();
            object.setUsuId(new Integer(resultSet.getString(1)));
            if (!utilitario.esnulo(resultSet.getString(2))) {
                object.setPersona(personaDao.getById(new Integer(resultSet.getString(2))));
            }
            if (!utilitario.esnulo(resultSet.getString(3))) {
                object.setConductor(conductorDao.getById(new Integer(resultSet.getString(3))));
            }
            object.setUsuUsuario(resultSet.getString(4));
            object.setUsuClave(resultSet.getString(5));
            object.setUsuTipo(resultSet.getString(6));
            object.setUsuActualiza(resultSet.getString(7));
        }
        resultSet.close();
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    return object;
}
```

Método de implementación de la interface de "UsuarioInterface" para obtener la información del último usuario registrado.

```java
@Override
public List<Usuario> listByParameter(Usuario object) {
    StringBuilder sql = new StringBuilder();
    List<Usuario> listObject = new ArrayList<>();
    try {
        sql.append("SELECT * FROM usuario WHERE 1 = 1 ");
        resultSet = ConexionDB.getConexion().createStatement().executeQuery(sql.toString());
        while (resultSet.next()) {
            object = inicializarObject();
            object.setUsuId(new Integer(resultSet.getString(1)));
            if (!utilitario.esnulo(resultSet.getString(2))) {
                object.setPersona(personaDao.getById(new Integer(resultSet.getString(2))));
            }
            if (!utilitario.esnulo(resultSet.getString(3))) {
                object.setConductor(conductorDao.getById(new Integer(resultSet.getString(3))));
            }
            object.setUsuUsuario(resultSet.getString(4));
            object.setUsuClave(resultSet.getString(5));
            object.setUsuTipo(resultSet.getString(6));
            object.setUsuActualiza(resultSet.getString(7));
            listObject.add(object);
        }
        resultSet.close();
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    return listObject;
}
```

Método de implementación de la interface de "UsuarioInterface" para obtener la información de los diferentes usuario por el id de la persona, id del conductor.

```java
@Override
public Usuario inicializarObject() {
    Usuario object = new Usuario();
    object.setUsuActualiza("N");
    object.setUsuTipo("N");
    return object;
}
```

Método de implementación de la interface de "UsuarioInterface" para inicializar un objeto de la clase usuario.

```java
@Override
public Usuario getByPerId(Integer id
) {
    Usuario object = null;
    if (id != null) {
        StringBuilder sql = new StringBuilder();
        sql.append("SELECT * FROM usuario WHERE PER_ID = ").append(id);
        try {
            resultSet = ConexionDB.getConexion().createStatement().executeQuery(sql.toString());
            while (resultSet.next()) {
                object = inicializarObject();
                object.setUsuId(new Integer(resultSet.getString(1)));
                if (!utilitario.esnulo(resultSet.getString(2))) {
                    object.setPersona(personaDao.getById(new Integer(resultSet.getString(2))));
                }
                if (!utilitario.esnulo(resultSet.getString(3))) {
                    object.setConductor(conductorDao.getById(new Integer(resultSet.getString(3))));
                }
                object.setUsuUsuario(resultSet.getString(4));
                object.setUsuClave(resultSet.getString(5));
                object.setUsuTipo(resultSet.getString(6));
                object.setUsuActualiza(resultSet.getString(7));
            }
            resultSet.close();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    return object;
```

Método de implementación de la interface de "UsuarioInterface" para buscar un usuario por el id de la persona asociado al registro.

```java
package Utilitario;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.swing.JOptionPane;

public class Utilitario {

    public boolean esnulo(String valor) {
        boolean retorno = true;
        if (valor != null) {
            if (!valor.equalsIgnoreCase("null")) {
                if (valor.trim().length() > 0) {
                    retorno = false;
                }
            }
        }
        return retorno;
    }

    public java.sql.Timestamp retornaFecha(Date fecha) {
        return new java.sql.Timestamp(fecha.getTime());
    }
```

Clase Utilitario que contiene métodos de uso común para las diferentes clases que conforman el Proyecto.

```java
public String transformaFechaFormatoDDMMYYYY(Date fecha) {
    SimpleDateFormat DATE_FORMAT = new SimpleDateFormat("dd/MM/yyyy");
    String datetext = DATE_FORMAT.format(fecha);
    return datetext;
}

public Date transformarStringFormatoDDMMYYYY(String fecha) {
    SimpleDateFormat formatoDelTexto = new SimpleDateFormat("dd/MM/yyyy");
    Date fech = null;
    try {
        fech = formatoDelTexto.parse(fecha);
    } catch (ParseException ex) {

    }
    return fech;
}

public Date transformarStringFormatoHHmmss(String fecha) {
    SimpleDateFormat formatoDelTexto = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    Date fech = null;
    try {
        fech = formatoDelTexto.parse(fecha);
    } catch (ParseException ex) {

    }
    return fech;
}
```

Métodos de la clase "Utilitario" para convertir fechas al formato adecuado para almacenar en la base de datos.

Diseño del reporte de Vehículos registrados. Con la extensión jrxml que nos proporciona el administrador de reportes "Ireport".

Proceso de Instalación

INSTALACION DEL JDK (JAVA) DEPENDIENDO LAS CARCATERISTICAS DEL COMPUTADOR EN ESTE CASO DE 32 BITS.

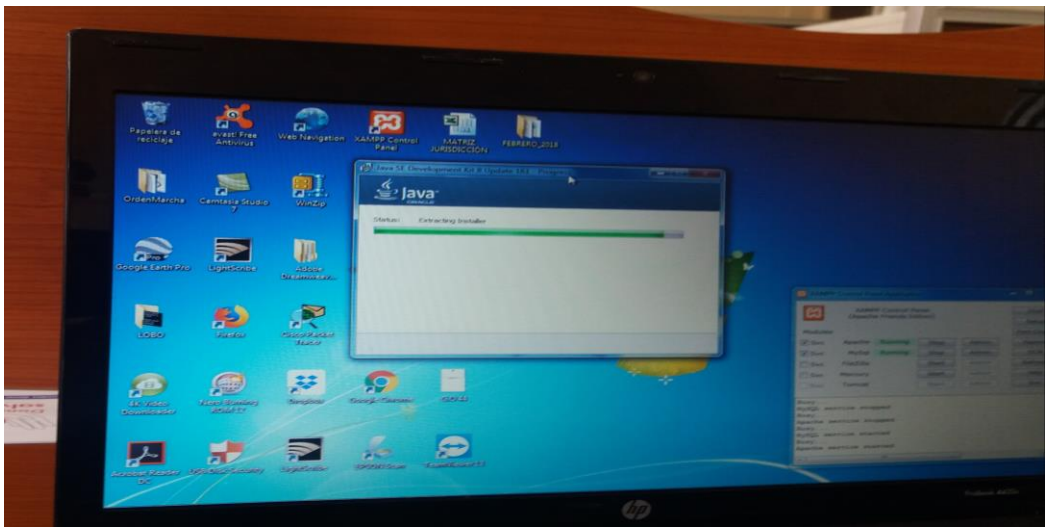


Figura 14 Instalación del JDK (JAVA)

Fuente: Propia
Elaborado por: Luis Lovato

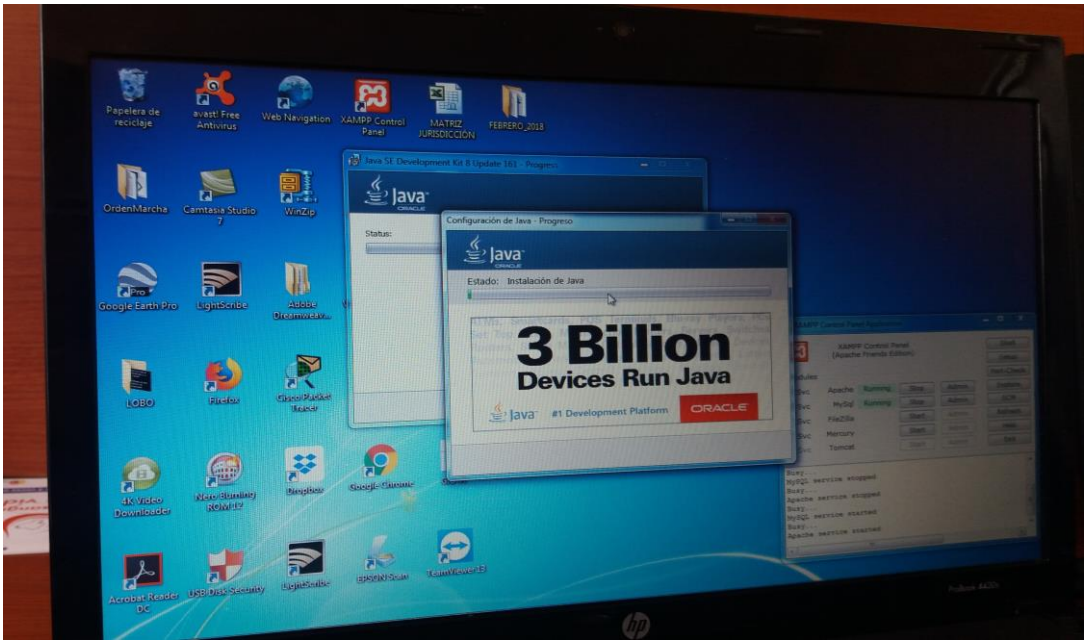


Figura 15 JDK(JAVA) correctamente instalado

Fuente: Propia
Elaborado por: Luis Lovato

INSTALACION DE LA CARPETA QUE CONTIENE LOS ARCHIVOS COMO REPORTES, LIBRERIAS Y EJECUTABLE DEL SISTEMA EN EL SITIO QUE CONSIDERE EL USUARIO.

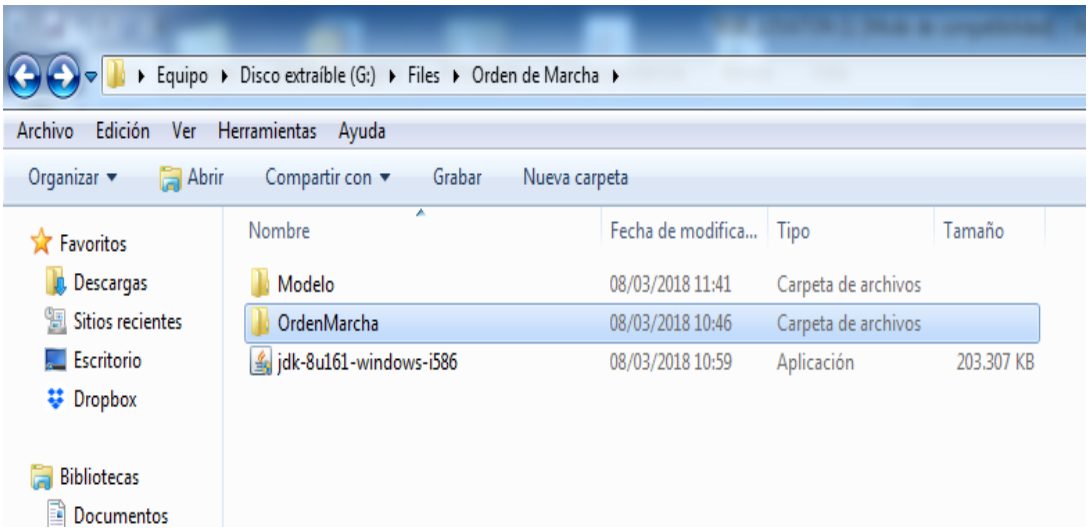


Figura 16 Carpeta con los módulos de instalación del sistema de control

Fuente: Propia
Elaborado por: Luis Lovato

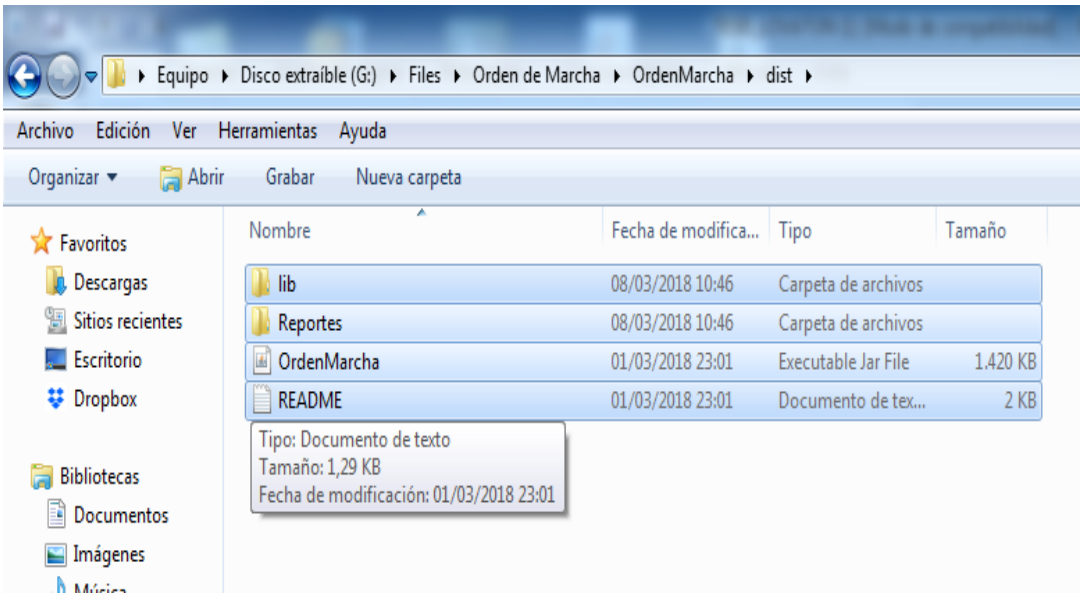


Figura 17 Módulos y ejecutable del sistema de control

Fuente: Propia
Elaborado por: Luis Lovato

INSTALACION DEL PAQUETE XAMPP QUE CONTIENE LA BASE DE DATOS DEL SISTEMA DE CONTROL Y CORRERLO.

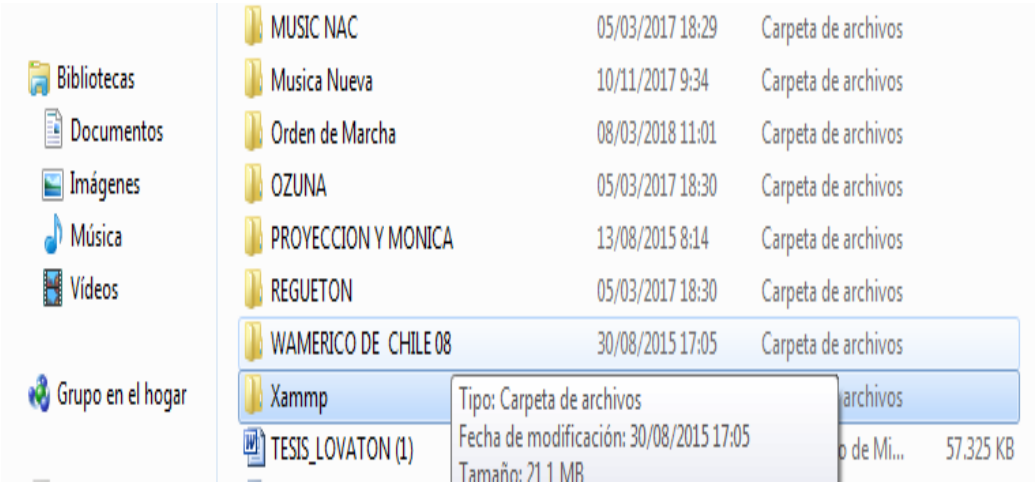


Figura 18 Paquete Xampp para la base de datos

Fuente: Propia
Elaborado por: Luis Lovato

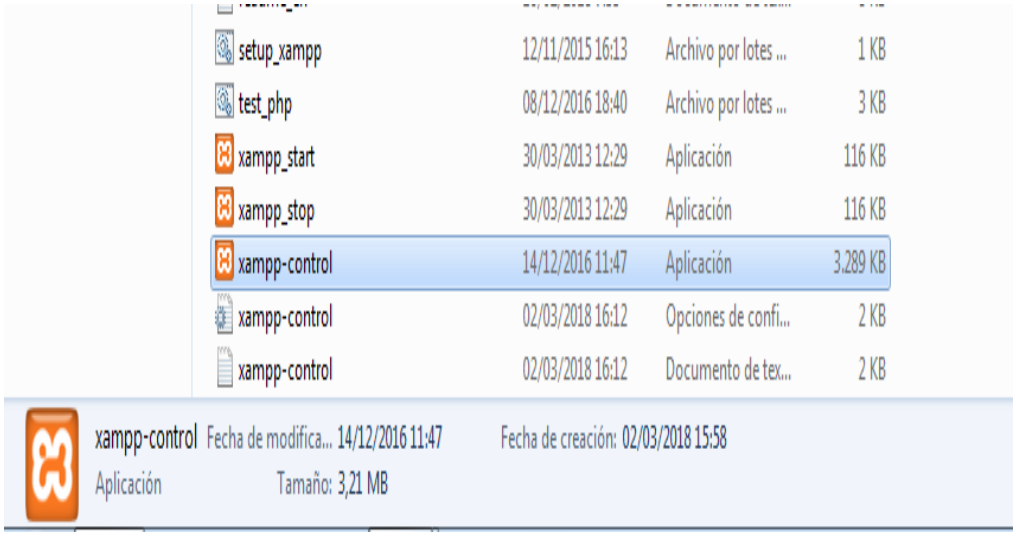


Figura 19 Xampp control necesario para la conexión de la base de datos

Fuente: Propia
Elaborado por: Luis Lovato

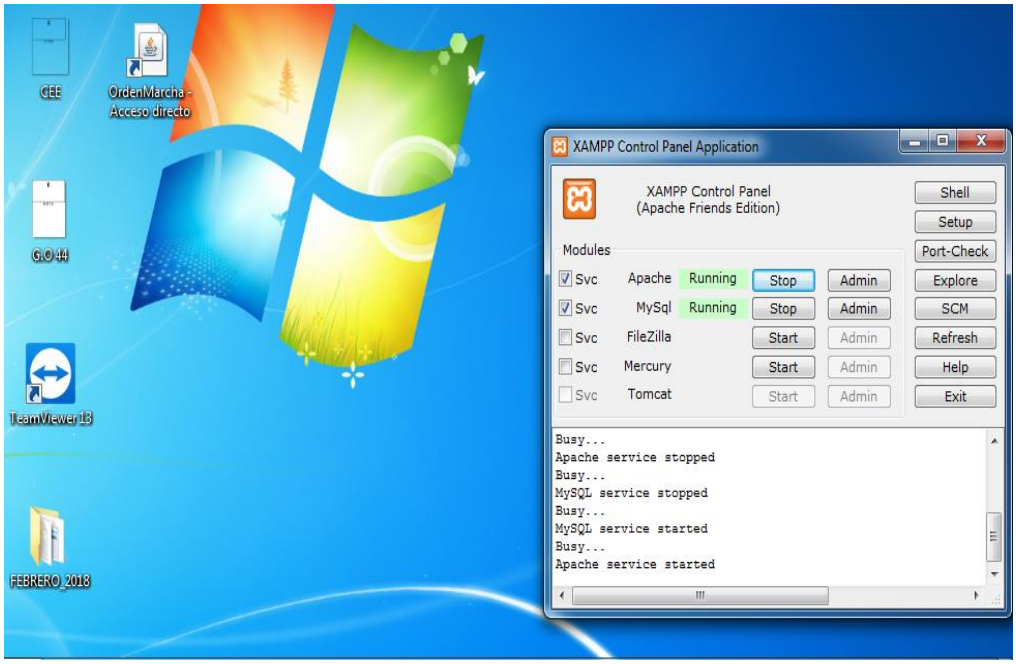


Figura 20 Conexión exitosa con la base de datos

Fuente: Propia
Elaborado por: Luis Lovato

CREAMOS UN ACCESO DIRECTO EN EL ESCRITORIO PARA MEJOR BUSQUEDA.

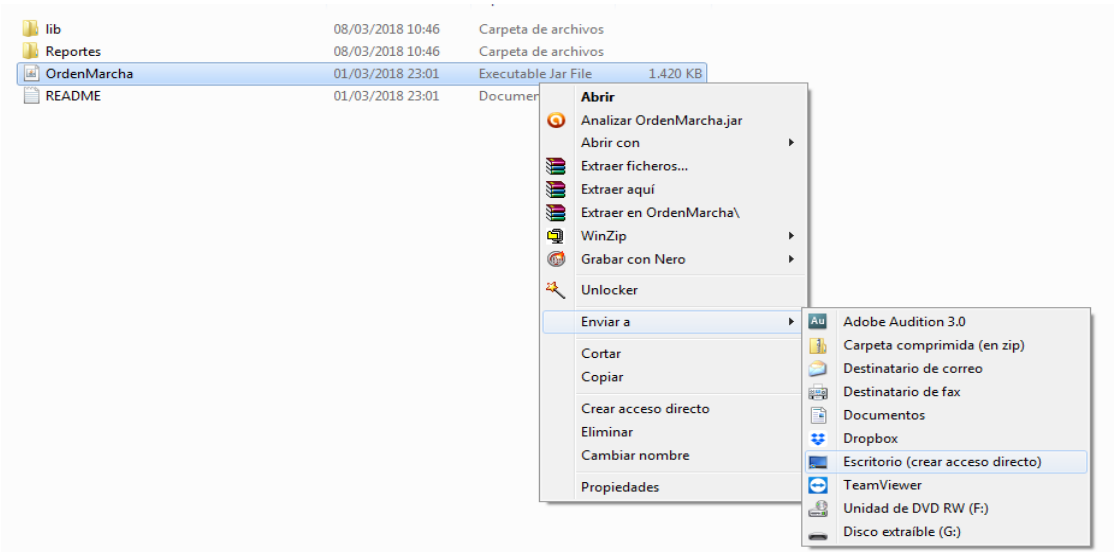


Figura 21 Proceso de creación del acceso directo del sistema de control

Fuente: Propia
Elaborado por: Luis Lovato

Figura 22 Acceso directo creado exitosamente

Fuente: Propia
Elaborado por: Luis Lovato

SISTEMA DE CONTROL INSTALADO EXITOSAMENTE.

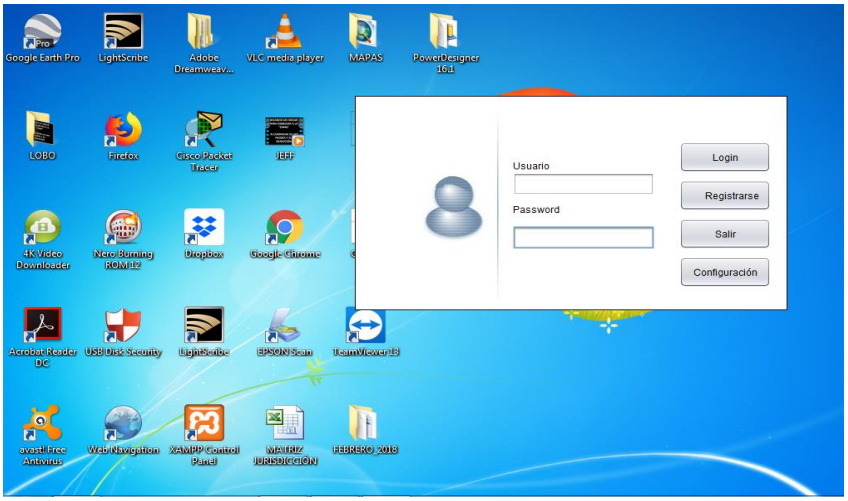


Figura 23 Pantalla de inicio del sistema de control

Fuente: Propia
Elaborado por: Luis Lovato

OBSERVACIONES:

- En caso de que la computadora ya tenga instalado el paquete xampp unicamente copiamos la base de datos en el mismo, ya que puede ocasionar un conflicto al momento de instalar el otro xampp.
- Si el ejecutable de la orden de marcha nos muestra el icono de RAR, es porque no esta instalado el JDK(JAVA), pues en ese instante hay que descargarse uno de acuerdo a las caracteristicas de su computador e instalarlo normalmente.

3.- Fotos de la Institución

Notas