UNIVERSIDAD CATÓLICA DE SANTA MARÍA

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

SISTEMA DE MONITOREO SATELITAL PARA DETERMINAR REGIONES CRÍTICAS EN UNA RUTA DE TRANSPORTE TERRESTRE - 2016

Tesis presentada por el bachiller:

MIGUEL ANGEL MAMANI ZEBALLOS

Para optar el título profesional de:

INGENIERO DE SISTEMAS

Asesora de tesis:

Dra. KARIM GUEVARA PUENTE DE LA VEGA

AREQUIPA - PERÚ

2017

PRESENTACIÓN

Sra. Directora de la Escuela Profesional de Ingeniería de Sistemas.

Sres. Miembros del Jurado Examinador de Tesis.

De conformidad con las disposiciones del Reglamento de Grados y Títulos de la Escuela Profesional de Ingeniería de Sistemas, pongo a vuestra consideración el presente trabajo de investigación titulado: **"SISTEMA DE MONITOREO SATELITAL PARA DETERMINAR REGIONES CRÍTICAS EN UNA RUTA DE TRANSPORTE TERRESTRE"**, el mismo que de ser aprobado me permitirá optar el Título Profesional de Ingeniero de Sistemas.

Arequipa, Diciembre de 2016

MAMANI ZEBALLOS MIGUEL ANGEL

Bachiller en Ingeniería de Sistemas

DEDICATORIA

A Dios, por el regalo de la vida y concederme la oportunidad de lograr una más de mis metas.

A la santísima Virgen María, por ser la luz y guía espiritual en el camino hacia mis objetivos.

A todo el amor, tiempo, dedicación, consejos y apoyo incondicional de mis padres: Salomé y Luis.

A toda mi familia, que me alentó con su interés y preocupación desde los inicios de mi vida universitaria.

AGRADECIMIENTOS

Un especial agradecimiento para la Dra. Karim Guevara Puente de la Vega y el Dr. Guillermo Calderón Ruiz, por su muy estimada y admirable asesoría, experiencia y orientación brindada en el desarrollo del presente proyecto.

A todos mis maestros, por su vocación y conocimientos transmitidos en estos últimos años.

A los consejos y el aliento de mis padres, familiares, amigos y a todas las personas que me brindaron parte de su tiempo y experiencia que contribuyeron en la realización de este proyecto.

ÍNDICE

Índice de Tablas

Índice de Figuras

RESUMEN

Muchos sistemas están sujetos a ser mejorados; una de las causas, es la existencia de nuevas necesidades del usuario final. La retroalimentación es una forma de aplicar un control a un sistema.

La inteligencia de negocios es cada vez más requerida y aplicada en la mayoría de empresas en el Perú, y una de las principales razones, es la necesidad de disponer de la mayor cantidad de información que pueda respaldar la toma de decisiones.

En las empresas de transporte con muchos viajes realizados, es muy necesario contar con la evaluación de los incidentes en el desplazamiento vehicular, para evaluar el nivel de seguridad de los servicios de transporte prestados. Para esto, es necesario contar y/o hacer la detección de todos los incidentes o riesgos existentes.

En esta tesis, se utiliza el proceso de descubrimiento de conocimiento para lograr obtener información oculta y generar un valor agregado a los almacenes de datos de los sistemas de monitoreo satelital de vehículos utilizados en los terminales terrestres actualmente, para ello es necesario utilizar técnicas de minería de datos correspondientes a la clasificación supervisada.

En un almacén de datos, según a un adecuado procesamiento, podemos encontrar información relevante; esta nueva información, está asociada a una serie de datos en

particular. Con el propósito de obtener un conocimiento útil a partir de un almacén de datos con las transmisiones de los vehículos monitoreados, es necesario hallar y vincular el geopunto más representativo para cada una de las transmisiones.

Finalmente, podemos clasificar mejor la correspondencia de los distintos registros de desplazamiento vehicular, según corresponda a una región crítica o no crítica. Esta fase de clasificación depende de los conjuntos de entrenamiento junto con las reglas de decisión preestablecidas.

Palabras Clave: Descubrimiento de conocimiento en bases de datos, Minería de datos, Algoritmo NN, Reglas de Decisión.

ABSTRACT

Many systems are subject to be improved; one cause is the existence of new end-user needs. Feedback is a way to apply a control to a system.

Business intelligence is becoming more required and applied in most companies in Peru, and one of the main reasons is the need to have for as much information to support decision-making.

In transport companies with many trips made, it is very necessary to have the evaluation of incidents in vehicular displacement, to evaluate the security level of transport services provided. For this is necessary to have and / or do the detection of all incidents or existing risks.

In this thesis, the process of knowledge discovery is being used to obtain hidden information and generate added value to data warehouse of the vehicles monitoring systems by satellite currently used in ground terminals, for it is necessary to use techniques of data mining corresponding to the supervised classification.

In a data warehouse, according to proper processing, we can find relevant information; this new information is associated with a number of particular data. With the purpose to obtain an useful knowledge from a data warehouse with transmissions of monitored

vehicles, it is necessary to find and link the most representative geopoint for each of the transmissions.

Finally, we can better classify the correspondence of the various registers of vehicular displacement, as appropriate to a critical or not-critical region. This phase to qualifying depends on the training sets along with the preestablished decision rules.

Keywords: Knowledge Discovery in Databases, Data Mining, Nearest Neighbors Algorithm, Decision Rules.

INTRODUCCIÓN

En la actualidad, los sistemas de monitoreo satelital de vehículos, implican muchos más que el almacenamiento de datos históricos, debido a que con un adecuado procesamiento, estos grandes volúmenes de datos pueden proveer información nueva, relevante, útil y de gran importancia para las empresas. El procedimiento de monitoreo vigente, exige que los registros de desplazamiento deben contener una serie de atributos básicos de referencia para cada una de las distintas transmisiones del estado vehicular en un tiempo y espacio determinado.

La mejora continua de una empresa involucra procesos de identificación de los problemas, oportunidades y el cumplimiento de una serie de objetivos propuestos.

La influencia de incorporar las tecnologías de información puede crecer exponencialmente ante la exigencia de competitividad en el mercado; para resolver problemas específicos previamente identificados teniendo en cuenta que un novato o un experto pueden encontrar nuevas necesidades y/o soluciones.

Dentro de los riesgos existentes en la actividad de transporte terrestre de personas, los conductores constituyen o representan uno de riesgos más crítico a controlar o reducir. Para ello, el sistema propuesto en esta tesis, describe un flujo de trabajo más completo para permitir evaluar y controlar el desempeño de los conductores.

La finalidad de esta tesis, es representar el proceso de descubrimiento de conocimiento en bases de datos a partir del análisis de reportes históricos estandarizados; estos reportes corresponden al seguimiento de unidades vehiculares y fueron generados por un sistema vigente de consulta y monitoreo en tiempo real; este y otros sistemas pertenecen a empresas autorizadas y encargadas de brindar el servicio tercerizado de monitoreo satelital de vehículos a las empresas de transporte terrestre; por razones de confidencialidad se omiten los nombres.

Esta tesis está estructurada en los siguientes capítulos:

En el Capítulo I se desarrolla la descripción del planteamiento teórico en torno al problema, en el cual se detallan los objetivos, preguntas de investigación, justificación e importancia de la investigación. En el Capítulo II se presenta el marco teórico, en los que se destaca la metodología a emplear y los algoritmos de clasificación. En el Capítulo III se describe el desarrollo aplicativo del sistema basado en las fases de la metodología CRISP-DM. En el Capítulo IV se presenta el análisis y discusión de los resultados obtenidos mediante un caso de prueba propuesto, el análisis y requerimientos del sistema propuesto están respaldados por una evaluación de expertos. Finalmente se mencionan las conclusiones y recomendaciones.

CAPÍTULO 1

PLANTEAMIENTO TEÓRICO

1.1. DESCRIPCIÓN DEL PROBLEMA

En el Perú, el servicio de transporte terrestre interprovincial para personas, es controlado y monitoreado mediante los sistemas de localización automática de vehículos en línea; este procedimiento, establecido desde hace ya varios años atrás mediante el Decreto Supremo N° 017-2009-MTC (2009), tiene como finalidad, brindar mayor seguridad a los usuarios del servicio de transporte. A pesar de esta medida de control adicional, el porcentaje de accidentes de tránsito registrados anualmente, Accidentes Declarados en las Unidades de la PNP (2015), no ha mostrado una disminución considerable, inclusive si se toman en cuenta como referencia las estadísticas de años anteriores a la implementación de dicha medida.

El desarrollo adecuado de las funciones del Centro de Control y Monitoreo de Flotas (2015), está respaldado por los sistemas de las empresas de monitoreo satelital de vehículos autorizadas por el MTC. Las distintas transmisiones del desplazamiento vehicular monitoreado, pueden ser exportadas mediante un tipo de Reporte de Tracking de Unidades (2016), mediante estos reportes podemos apreciar que dichos sistemas poseen una funcionalidad limitada debido a que no es factible tener la certeza del número total de servicios de transporte realizados por un

vehículo en un rango de fechas determinado (e.g., días, semanas, meses); en estas condiciones, los distintos tramos monitoreados sólo representan un historial de transmisiones, y en la actualidad resulta una necesidad fundamental contar con información íntegra y confiable a la hora de realizar o evaluar el nivel de seguridad del servicio de transporte prestado por una empresa, independientemente de los tipos de calidad de servicios brindados por las distintas empresas.

1.2. OBJETIVOS

1.2.1. Objetivo General

Desarrollar un sistema de monitoreo vehicular, que permita determinar las regiones críticas en una ruta de transporte terrestre, a partir de un almacén de datos correspondiente al monitoreo satelital de vehículos requerido por la SUTRAN.

1.2.2. Objetivos Específicos

1. Diseñar un modelo de clasificación de transmisiones, que permita integrar y actualizar la información en las transmisiones del desplazamiento vehicular registrado en un viaje monitoreado.
2. Demostrar la utilidad de los datos de salida obtenidos por el sistema propuesto mediante el análisis de la evaluación de expertos.

3. Proponer una mejora para el flujo de trabajo del sistema de monitoreo vigente empleado en los terminales terrestres, mediante la cual se permita gestionar las transmisiones almacenadas a nivel de viajes realizados.

1.3. PREGUNTAS DE INVESTIGACIÓN

1. ¿Las regiones críticas detectadas en una ruta de transporte pueden servir como un indicador para evaluar el procedimiento actual del control realizado en el servicio de transporte terrestre?
2. ¿El modelo de clasificación permitirá reevaluar las transmisiones, para integrar y actualizar la información contenida en las transmisiones del desplazamiento vehicular registrado en un viaje?
3. ¿La interpretación de la evaluación de expertos proporcionará la retroalimentación necesaria para medir la efectividad y utilidad del sistema propuesto para obtener nueva información?
4. ¿Es posible obtener y/o proponer un flujo de trabajo orientado a desarrollar un mejor control de los viajes registrados?

1.4. ÁREA Y LÍNEA DE INVESTIGACIÓN

- **Área de Investigación:** Sistemas de Información y Bases de datos.
- **Línea de Investigación:** Data Mining - Data Warehouse.
- **Sub-líneas:**
  - ❖ Sistemas de Información para la Toma de Decisiones.
  - ❖ Computer-Supported Cooperative Work.
- **Tipo de Investigación:** Aplicada.
- **Nivel de Investigación:** Experimental.

1.5. JUSTIFICACIÓN E IMPORTANCIA DE LA INVESTIGACIÓN

En el Perú, el servicio de transporte terrestre interprovincial es normado, supervisado, fiscalizado y sancionado por la Superintendencia de Transporte Terrestre de Personas, Carga y Mercancías (SUTRAN); entidad creada mediante la Ley N° 29380 (2009), adscrita al Ministerio de Transportes y Comunicaciones (MTC). La SUTRAN cuenta con un "Centro de Control y Monitoreo de Flotas" (2015), para velar por el respeto y cumplimiento de las normas sobre transporte terrestre, dichas funciones son respaldadas por el Decreto Supremo N° 017-2009-MTC (2009), en el cual, se estipula que las unidades vehiculares de las empresas destinadas al transporte terrestre, deben contar con un sistema de control y monitoreo inalámbrico que transmita mediante un dispositivo GPS a la autoridad en forma permanente la información del vehículo en ruta.

Por lo tanto, todas las transmisiones están en un almacén de datos y pueden ser visualizadas mediante un reporte con filtros de intervalo de fechas y las placas vehiculares registradas; a pesar de ello, todos los servicios de transporte operados por una determinada empresa no pueden ser visualizados de manera distintiva, debido al flujo de trabajo limitado en la mayoría de sistemas empleados en los terminales terrestres, y siendo, una de las necesidades primordiales de las empresas de transporte realizar el registro de los viajes de sus unidades vehiculares. El nivel de interacción con los sistemas de monitoreo disponibles es muy bajo, y una de sus funcionalidades principales actualmente es la de permitir visualizar datos históricos de los distintos estados de la geolocalización en el intervalo de tiempo de los vehículos consultados.

Hoy en día, el tema de calidad de servicio no es ajeno al rubro de transportes, para ello se debe evaluar el desplazamiento de un vehículo necesariamente a nivel de viajes; en la actualidad, las distintas transmisiones son evaluadas para verificar si un vehículo cometió una infracción por exceso de velocidad y un viaje es tema de investigación cuando se pretende esclarecer las causas de un accidente de tránsito.

En consecuencia, resulta muy necesario tener en cuenta que los sistemas de monitoreo satelital de vehículos vigentes en los terminales terrestres deben ser mejorados o reemplazados.

CAPÍTULO 2

MARCO TEÓRICO

2.1. ESTADO DEL ARTE

En esta sección, se describen una serie de propuestas y medidas de control formuladas en documentos y publicaciones, que evidencian la preocupación ante la problemática persistente referente a la seguridad vial y reducción de accidentes en el tránsito vehicular.

En el artículo de (García, F., Escobar, D. & Vásquez L., 2010) se muestran las diversas posibilidades de aplicación a casos prácticos referentes a los GPS, hoy en día muchas empresas ya disponen de información almacenada de varios meses e incluso años; en este artículo se presenta un enfoque para determinar las velocidades para cada uno de los segmentos de vía que componen la red vial de una ciudad a partir de las velocidades registradas se realiza un análisis para el desplazamiento promedio estimado de vehículos en un determinado día.

En el trabajo de (Sarmiento, 2012) se desarrolla un prototipo de sistema de información geográfico en entorno Web que permite detectar los eventos ocurridos en ruta de vehículos con GPS de una forma interactiva y funcional para el usuario, mediante la integración de las tecnologías de Bases de Datos, Cache de Mapas y

AJAX; el diseño del sistema corresponde al patrón de diseño MVC para permitir manipular y visualizar los datos geográficos procesados con Bases de Datos Espaciales.

En el trabajo de (Basulto, 2013) se describe como obtener o descubrir conocimiento acerca de los accidentes de tránsito que afectan a una empresa de transporte en un determinado periodo de años mediante la aplicación de técnicas de minería de datos, este trabajo fue posible mediante el acceso a una base de datos de las afectaciones ocurridas en dicha empresa con datos dispersos y sin normalizar; por medio de una metodología se analizan los accidentes de tránsito a fin de facilitar la extracción de la información implícita en los datos de una manera organizada de lo general a lo particular, para ayudar a que se adopten medidas encaminadas a reducir el nivel de accidentalidad.

El sistema propuesto, en esta tesis, está basado en torno a las pautas detalladas en el artículo de García, debido a que las reglas de decisión están consideradas en base a los límites de velocidad permisibles para una determinada zona de desplazamiento vehicular. Sarmiento (2012) detalla cómo funciona el dispositivo GPS y las tecnologías que emplea para integrar un sistema que registra el traslado vehicular, en esta propuesta se realiza un control de los distintos eventos ocurridos en geozonas registradas y delimitadas en forma de polígonos para evaluar el ingreso o salida a las zonas permitidas (Geocercas) pertenecientes a la ruta asignada a un vehículo; la presente tesis, se apoya en el uso de geocercas enfocado desde un punto de vista diferente, para ello no será necesario considerar emplear bases de

datos espaciales debido a que las geocercas poligonales no formarán parte del sistema propuesto, por lo tanto, se utilizará un gestor de base de datos normal que permita contener el flujo propuesto y así almacene los datos de salida esperados; dicha información puede servir como apoyo en la toma de decisiones a nivel estratégico y será exclusivamente de uso corporativo; para ello, el sistema propuesto contiene la funcionalidad de administrar en forma más ordenada las transmisiones del desplazamiento vehicular, organizándolas según la placa del vehículo, fecha y hora correspondiente a cada uno de los viajes realizados. El trabajo de Basulto (2013), es el antecedente que muestra, que sí es viable analizar desde otro punto de vista las posibles causas de accidentes en los datos históricos; en la presente tesis, se realizará un análisis de lo particular a lo general; debido a que un registro de los datos de monitoreo de un vehículo en movimiento corresponde a un instante de tiempo único con atributos de velocidad y geolocalización muy exactos, para referenciar con más exactitud el tipo de geozona a la cual pertenece la señal de GPS registrada forma parte de uno de los desafíos a superar en el desarrollo del trabajo de investigación presente; por lo que, es de vital importancia contar con un conjunto de entrenamiento de geopuntos lo bien definidos que abarque un gran porcentaje de la ruta a evaluar.

Tomando en cuenta los detalles mencionados anteriormente y considerando las especificaciones y recomendaciones establecidas en el “Texto Único Ordenado del Reglamento Nacional de Tránsito”, detallado en el Decreto Supremo N° 016-2009-MTC (2009), para los límites de velocidad permisibles en cada uno de los distintos geopuntos (conjunto de entrenamiento) asignados a una ruta, únicamente resta

organizar, planificar, evaluar y desarrollar los algoritmos necesarios que permitan la realización de la clasificación de correspondencia de los datos históricos; permitiendo de esta manera la realización de minería de datos además de alcanzar el objetivo principal propuesto en el presente tema de investigación.

2.2. MARCO CONCEPTUAL

En esta sección, se describirán los conceptos involucrados en la propuesta a desarrollarse en el presente tema de tesis.

2.2.1. El proceso de descubrimiento de conocimiento en bases de datos

En (Lara, 2014), se describe que, el conocimiento extraído por el proceso de KDD (en inglés, Knowledge Discovery in Databases) ha de poseer las siguientes cuatro características: no trivial, implícito, previamente desconocido y útil.

En (Pérez & Santín, 2007), podemos encontrar que, el proceso de KDD es un conjunto de tareas o fases, entre las que destacan:

- Establecimiento de un problema relevante.
- Selección de los datos adecuados para dar solución al problema.
- Exploración y limpieza de los datos.

- Procesamiento y modificación de los mismos.
- Aplicación de técnicas de modelado.
- Obtención e interpretación de los modelos obtenidos.
- Utilización del conocimiento obtenido.
- Generación de nuevos datos a partir de su aplicación en el mundo real.

Para (Lara, 2014), el proceso de KDD se compone de diferentes fases que, tal y como muestra la figura 2.1, son las siguientes:

- **Recopilación de datos.** En esta fase, los datos, procedentes de diferentes fuentes, se integran en un mismo y único repositorio de datos, denominado almacén de datos, más conocido como data warehouse.
- **Selección, limpieza y transformación de datos.** Sobre los datos recopilados en el almacén de datos no es posible realizar aún data mining, debido a que dichos datos pueden no estar limpios, pueden contener atributos irrelevantes, etc. Precisamente, en esta fase del proceso de KDD, se realiza una selección de los datos integrados en el data warehouse. Dichos datos, además, se limpian y transforman de cara a fases posteriores. El resultado de esta fase es la denominada vista minable.
- **Data mining.** Una vez se cuenta con una vista minable, el siguiente paso consiste en aplicar técnicas concretas de minería de datos para obtener modelos.

- **Interpretación y evaluación de modelos.** Los modelos obtenidos en la fase de data mining han de ser evaluados. Una vez comprobada la calidad de los mismos, estos son interpretados y, a partir de ellos, se obtiene el conocimiento.

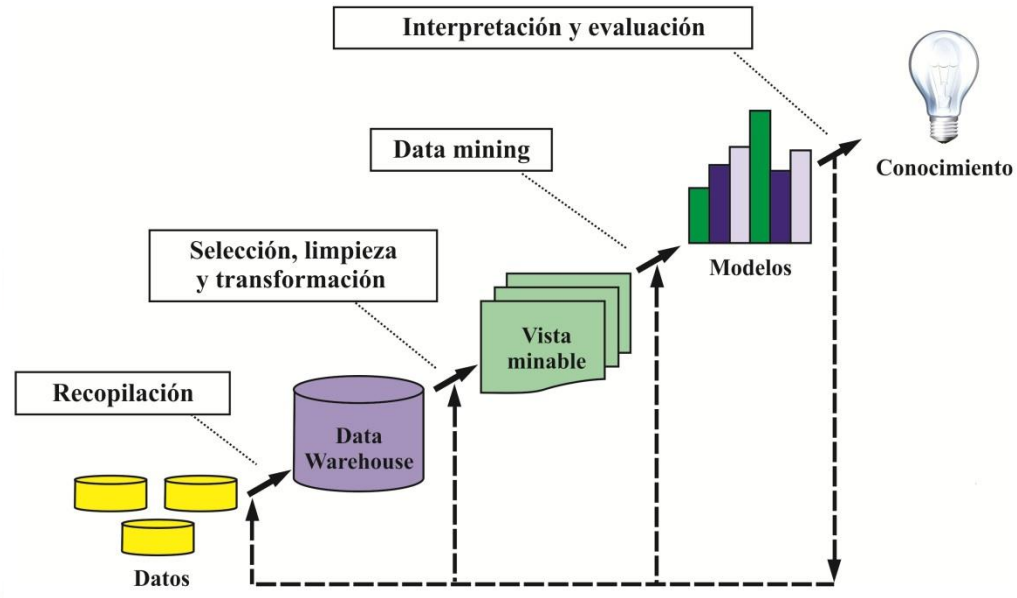


Figura 2.1. Proceso de KDD.
Fuente: (Lara, 2014).

En el libro de (Silberschatz, Korth & Sudarshan, 2006), se propone que, algunos tipos de conocimiento descubiertos a partir de una base de datos pueden representarse por un conjunto de reglas. Por supuesto, estas reglas no son verdaderas de modo universal, y tienen grados de soporte y de confianza.

2.2.2. Minería de datos

La minería de datos, es una fase del proceso de KDD en la que se obtienen modelos representativos de los datos por medio de la ejecución de diferentes técnicas y algoritmos, utilizados para resolver alguna tarea de data mining. (Lara, 2014).

El libro de (Silberschatz, Korth & Sudarshan, 2006), detalla que, el término minería de datos (data mining) hace referencia al proceso de análisis semiautomático de bases de datos de gran tamaño para hallar estructuras útiles. Al igual que la búsqueda de conocimiento en la inteligencia artificial, o el análisis estadístico, la minería de datos intenta descubrir reglas y estructuras a partir de los datos.

Suele haber una parte manual en la minería de datos, que consiste en el preprocesamiento de los datos hasta una forma aceptable para los algoritmos, y en el post procesamiento de las estructuras descubiertas para hallar otras nuevas que puedan resultar útiles, y puede que se necesite la interacción manual para escoger los tipos de estructuras útiles.

Los datos pueden clasificarse en: cuantitativos y cualitativos. A los diferentes tipos de problemas que se pueden resolver mediante el uso de técnicas de data mining se les conoce con el nombre de tareas.

En (Lara, 2014), podemos encontrar, que las tareas de data mining se suelen clasificar dependiendo del tipo de modelo que son capaces de generar. En función de ello, las tareas de data mining se clasifican en: tareas predictivas y tareas descriptivas.

Dentro de las aplicaciones de la minería de datos, la predicción es uno de los tipos más importantes de minería de datos, y una de las principales tareas predictivas de data mining es la clasificación.

En el libro de (Elmasri & Navathe, 2007), se establece que uno de los tipos de conocimiento adquirido durante la minería de datos, corresponde al de "Jerarquías de clasificación", la cual, propone trabajar a partir de un conjunto existente de eventos o transacciones para crear una jerarquía de clases.

A. Clasificación

> La clasificación es una tarea predictiva de data mining cuyo objetivo es predecir el valor, desconocido, de un atributo para un determinado ejemplo. Para dicha predicción, se utiliza datos históricos de otros ejemplos en los que sí se conoce el valor de la clase. (Lara, 2014).

> Dentro de los principales tipos de técnicas y algoritmos de clasificación existentes se encuentran las siguientes:

- **Basados en árboles de decisión.** Son técnicas y algoritmos que construyen estructuras arborescentes que se pueden aplicar para decidir la clase de un ejemplo sin clasificar.
- **Técnicas basadas en casos.** La clasificación se realiza comparando el nuevo ejemplo a clasificar con los ejemplos existentes de los que se conoce su clase, buscando aquellos ejemplos más similares al ejemplo a clasificar. El algoritmo de los *k* vecinos más próximos (k-nearest-neighbors) es un ejemplo de este tipo de técnicas.

La clasificación se puede llevar a cabo hallando reglas que dividan los datos en grupos disjuntos. El proceso de creación de clasificadores comienza con una muestra de los datos, denominada conjunto de entrenamiento, para cada tupla del conjunto de entrenamiento ya se conoce la clase a la que pertenece. (Silberschatz, Korth & Sudarshan, 2006).

2.2.3. Algoritmos de clasificación por vecindad

En los algoritmos basados en vecindad, la decisión de clasificar un caso $\boldsymbol{x}$ en la categoría $\boldsymbol{\theta}$ depende de una colección de N casos previamente clasificados $(x_1, \theta_1), (x_2, \theta_2), \dots, (x_N, \theta_N)$. Este tipo de problemas se enmarca en el dominio de la clasificación supervisada, y no existe un clasificador óptimo que resuelva satisfactoriamente todos los problemas que se pueden plantear. (Sierra, 2006).

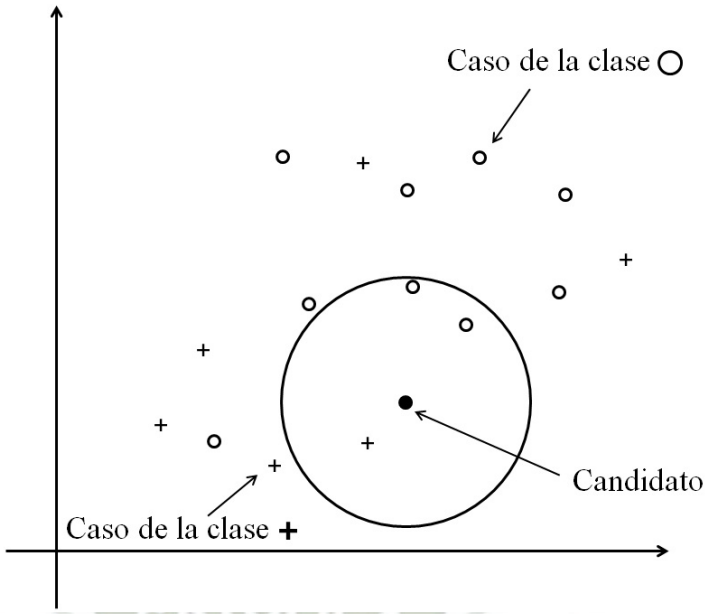


Figura 2.2. La regla NN frente a la regla K-NN (para K=3).
Fuente: (Sierra, 2006).

En el libro de (Sierra, 2006), se define que, para solucionar un problema de clasificación abordado con un enfoque basado en criterios de vecindad se puede caracterizar del siguiente modo:

1. Se dispone de un conjunto de N prototipos (o muestras ya clasificadas) llamado "conjunto de entrenamiento".
2. Tenemos que clasificar un nuevo caso, x, no perteneciente al conjunto de entrenamiento.
3. Existe una métrica entre los diferentes objetos del espacio de representación.

4. No se utiliza ninguna otra información acerca de la distribución de los parámetros estadísticos asociados al conjunto de entrenamiento.

En este caso, el clasificador asociará al caso ***x***, la clase verdadera del objeto que se encuentra más próximo a ***x***, dentro del espacio de representación.

- **Distancia euclídea:**

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$entre\ dos\ puntos\ P_1\ y\ P_2,$
$de\ coordenadas\ cartesianas\ (x_1, y_1)\ y\ (x_2, y_2)$

En la figura 2.3 se muestra el pseudocódigo del algoritmo de clasificación por la distancia mínima. La dificultad radica en la elección del único representante de cada clase; en el peor escenario se tendrá que comparar el nuevo elemento a clasificar con cada uno de los elementos del conjunto de entrenamiento, por lo que resultaría muy conveniente contar con una estrategia para obtener únicamente los elementos candidatos más cercanos al nuevo elemento a clasificar.

COMIENZO

Como entrada disponemos de:

El modelo, contenido M casos, uno por cada clase (x_i, θ_i), i =1, … ,M, y un nuevo caso (x, θ) que se desea clasificar

PARA cada caso del modelo (x_i, θ_i), HACER

COMIENZO

Calcular la distancia a x del caso

Sea D_i dicha distancia

FIN

Como salida, dar la clase θ_j cuya distancia D_j es la mínima de entre todas las clases

FIN

Figura 2.3. Pseudocódigo del algoritmo clasificador de la distancia mínima. ***Fuente:*** (Sierra, 2006).

2.2.4. Aprendizaje de árboles y reglas de decisión

Un árbol de decisión es una representación gráfica de un procedimiento para clasificar o evaluar un concepto. Intuitivamente, un árbol de decisión es una colección de condiciones organizadas jerárquicamente. Formalmente, un árbol de decisión es un árbol donde cada nodo representa una condición o test sobre algún atributo y cada rama que parte de ese nodo corresponde a un posible valor para ese atributo. Finalmente las hojas son las clases. (Palma & Morales, 2008).

El esquema básico que siguen la mayoría de los algoritmos de inducción de árboles de decisión consiste en construir el árbol de forma top-down, desde la

raíz a las hojas (Top-Down Induction of Decision Trees, TDIDT) (Quinlan, 1986).

En el libro de (Pajares & Santos, 2006) se detalla que; en el aprendizaje inductivo, el algoritmo de aprendizaje trata de encontrar un modelo de clasificación sencillo y general. Uno de los modelos más simples es el que encuentra una regla de clasificación.

Los árboles de decisión (Decision Trees o Top-Down Inductive Decision Trees, TDIDT) se construyen a partir de los ejemplos de forma inductiva siguiendo un proceso iterativo que comienza en la raíz del árbol y termina en sus hojas. Los nodos que no son hojas representan atributos de los ejemplos de muestra, las ramas representan los valores de dichos atributos y las hojas son los valores de la clase. La clasificación de un objeto se determina preguntando en cada nodo del árbol, desde la raíz, por el atributo del nodo. La rama que coincide con la respuesta es la elegida hasta alcanzar los valores de la clase. (Pajares & Santos, 2006).

En los árboles de decisión, los nodos representan la verificación de una condición sobre un atributo, las ramas representan el valor de la condición comprobada en el nodo del cual derivan y los nodos hoja representan las etiquetas de clase. (Lara, 2014).

Tabla 2.1.

Características generales de métodos de construcción de árboles.

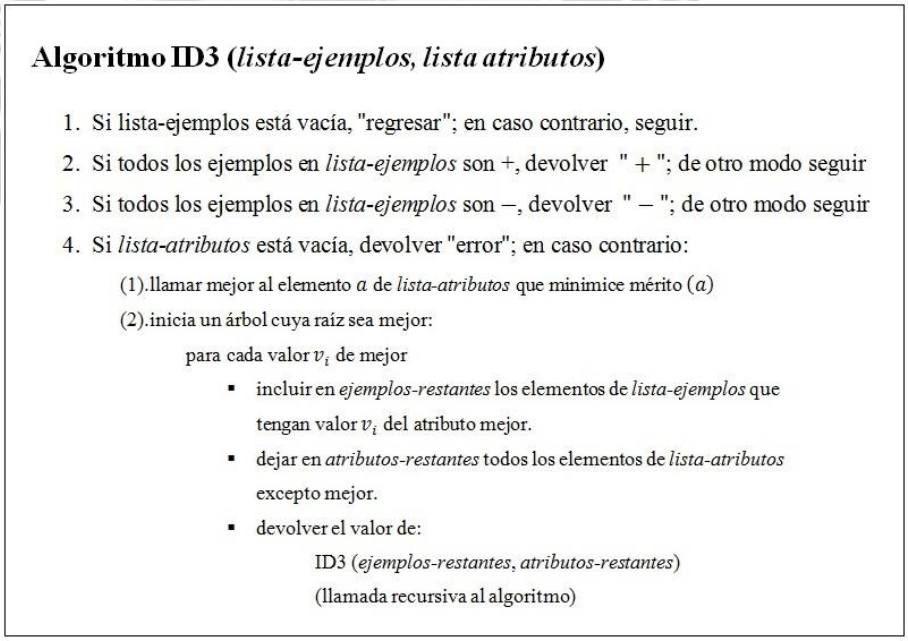
Algoritmo	Variables predictoras	Tipo de división	Criterio de División	Casos missing	Método de poda
CART (1984)	Continuas Discretas	Binaria	Ganancia (Gini index)	SI	Post-poda
ID3 (1979)	Discretas	n-aria	Ganancia en información (Entropía)	NO	-
C4.5 (1993)	Continuas Discretas	Binaria / n-aria	Gain ratio (Entropía)	SI	Post-poda
J4.8	Continuas Discretas	Binaria / n-aria	Gain ratio (Entropía)	SI	Post-poda
CHAID (1975)	Discretas	n-arias	x^2	SI	Pre-poda (nivel de significancia)

Fuente: (Sierra, 2006).

El árbol de decisiones necesario a emplear para dar solución al problema planteado en esta tesis, corresponde al tipo n-ario y los distintos registros no presentan datos incompletos; por lo que, ID3 resulta el algoritmo ideal.

A. Algoritmo ID3

El nombre viene de "Iterative Dichotomiser 3" y es el antecesor del C4.5. No realiza ningún proceso de poda y las divisiones se realizan sobre todos los posibles valores de la variable predictora seleccionada (solo es capaz de tratar variables discretas). Además, al utilizar el criterio de la ganancia en información como función de división, tiende a sesgarse hacia variables con un elevado número de categorías diferentes. Otras limitaciones de este algoritmo, además de las mencionadas, son que no contempla la posibilidad de casos con información incompleta (missing data) o con ruido (noise data). (Sierra, 2006).



Algoritmo ID3 (*lista-ejemplos, lista atributos*)

1. Si lista-ejemplos está vacía, "regresar"; en caso contrario, seguir.
2. Si todos los ejemplos en *lista-ejemplos* son +, devolver " + "; de otro modo seguir
3. Si todos los ejemplos en *lista-ejemplos* son –, devolver " – "; de otro modo seguir
4. Si *lista-atributos* está vacía, devolver "error"; en caso contrario:

   (1).llamar mejor al elemento a de *lista-atributos* que minimice mérito (a)

   (2).inicia un árbol cuya raíz sea mejor:

   para cada valor v_i de mejor

   - incluir en *ejemplos-restantes* los elementos de *lista-ejemplos* que tengan valor v_i del atributo mejor.
   - dejar en *atributos-restantes* todos los elementos de *lista-atributos* excepto mejor.
   - devolver el valor de:

     ID3 (*ejemplos-restantes*, *atributos-restantes*)

     (llamada recursiva al algoritmo)

Figura 2.4. Algoritmo ID3.

Fuente: (Pajares & Santos, 2006).

2.3. Global Position System (GPS)

GPS es un sistema que tiene como objetivo la determinación de las coordenadas espaciales de puntos respecto de un sistema de referencia mundial. Los puntos pueden estar ubicados en cualquier lugar del planeta, pueden permanecer estáticos o en movimiento y las observaciones pueden realizarse en cualquier momento del día. (Huerta, Mangiaterra & Noguera, 2005).

Para la obtención de coordenadas el sistema se basa en la determinación simultánea de las distancias a cuatro satélites (como mínimo) de coordenadas conocidas.

2.3.1. Sistema GPS - Constitución. Está constituido por tres segmentos fundamentales:

A. Segmento espacial. Se puede observar en la figura 2.5 la disposición aproximada que tienen los satélites de la constelación NAVSTAR, GPS que integran el segmento espacial.

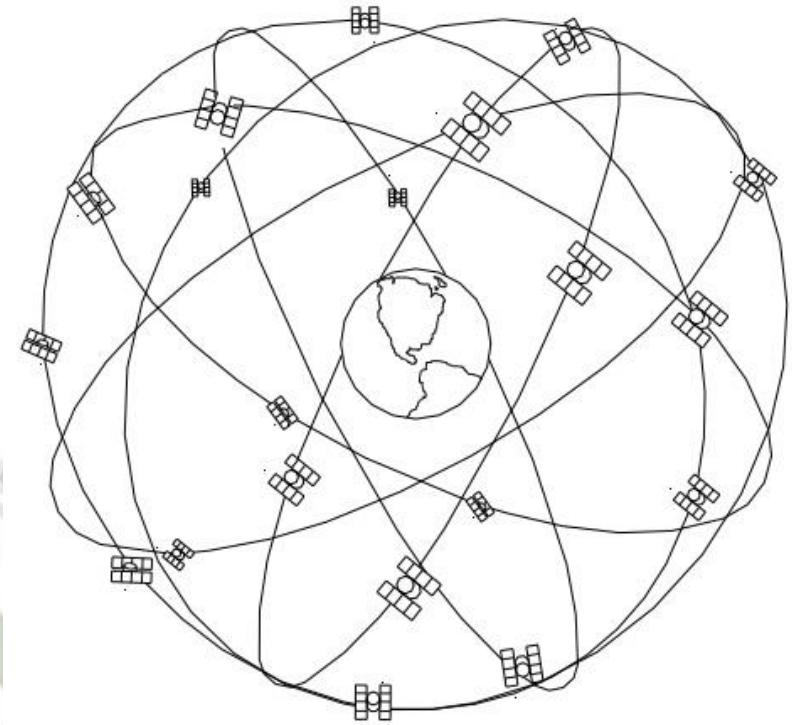


Figura 2.5. Constelación de satélites.

Fuente: (Huerta, Mangiaterra & Noguera, 2005).

B. Segmento de control. Sus funciones principales son:

- Monitoreo y control permanente de los satélites con el objeto de determinar y predecir las órbitas y los relojes de a bordo.
- Sincronización de los relojes de los satélites con el tiempo GPS.
- Transmisión, a cada satélite, de la información procesada.

C. Segmento de usuario. Está constituido por los instrumentos utilizados para recepcionar y procesar la señal emitida por los satélites. Estos instrumentos están integrados esencialmente, por una antena y un receptor.

La antena está conectada por cable al receptor o en otros casos forman una sola unidad. Las coordenadas que se calculan corresponden al centro radioeléctrico de la antena.

El receptor GPS es el dispositivo electrónico, que es instalado en los vehículos para el rastreo de su posición; consta de un mínimo de 4 canales que permiten recepcionar y procesar simultáneamente la señal de cada satélite. Posee además un oscilador de cuarzo que permite generar la frecuencia. Un microprocesador interno con el software correspondiente calcula las coordenadas de la antena y la velocidad.

Todo equipo adiciona una unidad de alimentación eléctrica que deberá brindar al receptor la autonomía necesaria. Estos dispositivos ofrecen diversas prestaciones según el fabricante.

2.4. Metodología CRISP-DM

CRISP-DM (Cross - Industry Standard Process for Data Mining) describe los enfoques de uso común que los expertos usan para enfrentar a los problemas de minería de datos (Chapman et al., 2000).

Este modelo de proceso para la minería de datos proporciona una visión general del ciclo de vida de un proyecto de minería de datos, dividido en seis fases, como se aprecia en la figura 2.6; sus tareas y relaciones entre ellas están representadas en la figura 2.7.

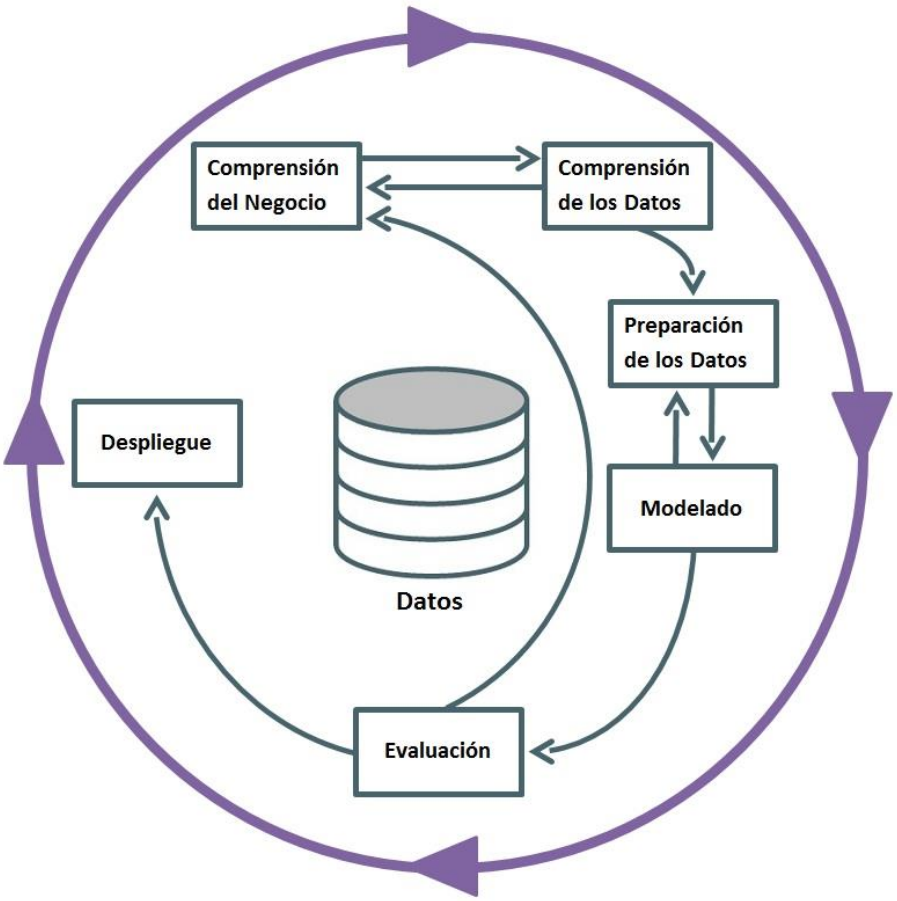


Figura 2.6. Fases de la metodología CRISP-DM.
Fuente: (Chapman et al., 2000).

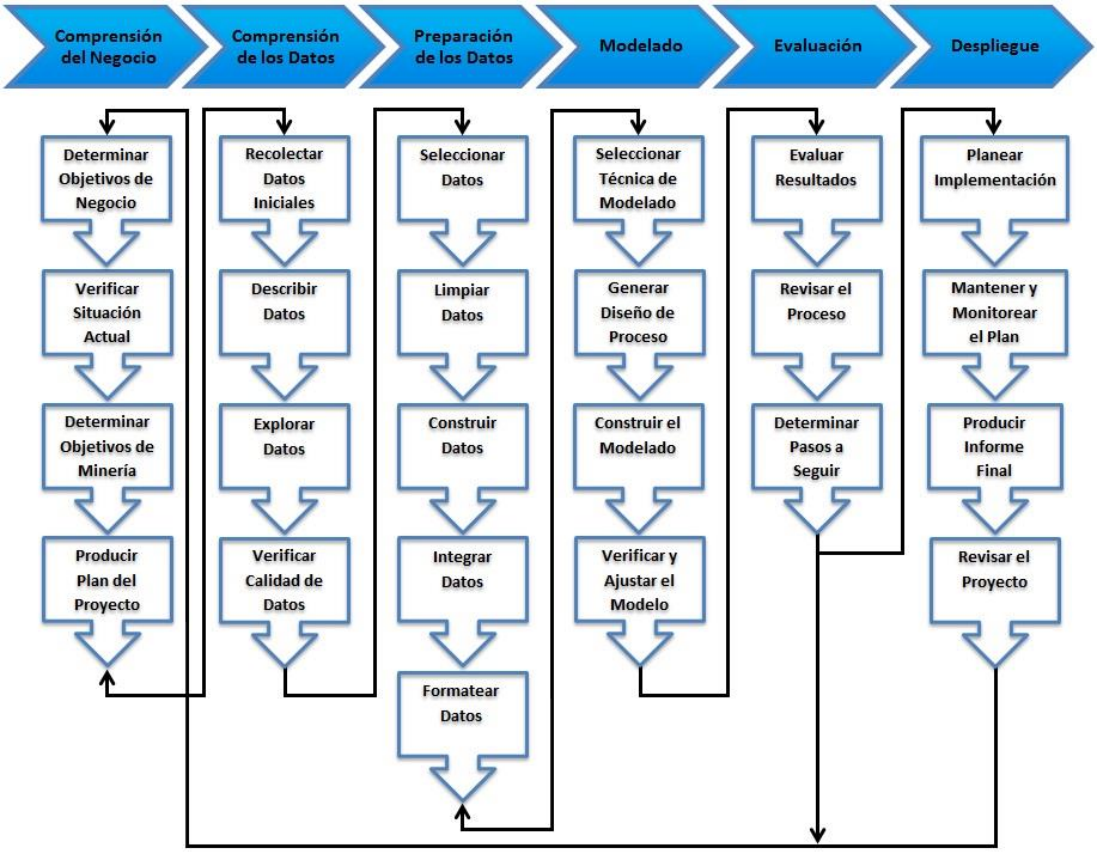


Figura 2.7. Descripción de un proyecto con la metodología CRISP-DM.
Fuente: (Chapman et al., 2007).

Esta metodología percibe diferentes objetivos entre los principales encontramos:

- Perseguir el cumplimiento de objetivos desde un punto de vista empresarial, por ende da preferencia a la comprensión del negocio.
- Desarrollar proyectos de minería de datos mediante un proceso estandarizado.
- Minimizar los costos que implica un proyecto de minería de datos en las empresas.

CAPÍTULO 3

DESARROLLO DE LA PROPUESTA

En este capítulo se aplicará la metodología CRISP-DM, desarrollando las distintas fases que propone para realizar la minería de datos, además, de exponer los detalles del módulo del sistema encargado de gestionar la obtención de reglas de decisión.

El sistema propuesto; nace como parte de una de las mejoras necesarias para los sistemas de monitoreo satelital de vehículos empleados por las empresas de transporte en los terminales terrestres en el Perú, las características y contribuciones que se desarrollarán, están detalladas en el contenido de esta sección.

3.1. Fase 1: Comprensión del negocio.

Las empresas de transporte terrestre, por lo general comparten la visión de "ser la empresa líder en el mercado". La finalidad de las empresas de transporte es obtener la mayor rentabilidad posible en los viajes programados y monitoreados, para esto, además de realizar una buena administración de los costos, en muy necesario no contar con gastos no programados como mantenimientos correctivos y reparaciones a causa de una conducción anormal o una colisión vehicular.

El objetivo principal del negocio es brindar un servicio de calidad que se destaque por puntualidad y seguridad, respaldado por la no existencia de accidentes de tránsito en los cuales recaiga la responsabilidad en la empresa de transporte. Para apoyar en el cumplimiento de este objetivo los operadores tienen que respetar y seguir la filosofía de "cero incidentes en ruta"; de esta manera, se puede contar con un ambiente más ideal para el desarrollo de las operaciones de transporte. Por lo tanto, se debe reducir o eliminar el número de incidentes registrados en un viaje, y para lograrlo, tienen que estar identificados todos o la mayoría de dichos incidentes. Teniendo en cuenta que el límite máximo permisible de velocidad no es realmente 90 km/h, debido al sustento contenido en el Informe N° 224-2013/GEL - INDECOPI (2013), en el cual a fin de no generar sanciones incorrectamente impuestas, se procedió a considerar un margen de error de 10 km/h en la velocidad detectada del vehículo que se encuentra en movimiento mediante el Decreto Supremo N° 009-2015-MTC (2015).

En el Perú, las empresas encargadas de brindar el servicio de transporte terrestre, están obligadas a adquirir el servicio de monitoreo satelital de vehículos para todas sus unidades de transporte. Según el Decreto Supremo N° 017-2009-MTC (2009); el cual, estipula que las unidades vehiculares de las empresas destinadas al transporte, deben contar con un sistema de control y monitoreo inalámbrico que transmita en forma permanente la información del vehículo en ruta al Centro de Control y Monitoreo de Flota de la SUTRAN.

Entre las funciones de la SUTRAN, una de las principales, es la de identificar y sancionar los excesos de velocidad de los vehículos infractores de las distintas empresas de transporte terrestre.

En el caso de los buses que brindan el servicio de transporte terrestre, los excesos continuos del límite de velocidad establecido registrados en un intervalo de tiempo mayor de 5 minutos corresponden a una sanción; aparentemente esta regla, se aplica a cualquiera fuese la zona o tramo involucrado debido a que no se indica lo contrario en dicho reglamento; y considerando que, hasta la fecha existe un elevado nivel de indeterminación respecto a la correspondencia exacta para muchos de los puntos transmitidos según el procedimiento de monitoreo satelital actual, siendo esta, una de las principales razones por las cuales no se puede tener la certeza respecto a qué tan eficiente fue realizado el transporte de una ciudad hacia otra, y teniendo en cuenta, que los módulos de reportes de los sistemas de monitoreo satelital vigentes no cuentan con los recursos necesarios para gestionar una interpretación ideal de los distintos eventos registrados en el desarrollo de uno o varios viajes monitoreados por el dispositivo GPS asignado a una unidad de transporte. La función principal de dicho dispositivo simplemente está limitada a transmitir las coordenadas de geolocalización (latitud y longitud) con un margen de error de 5 a 15 metros, velocidad instantánea de desplazamiento (Km/h), la fecha y la hora forman parte del vector de datos, que genera un registro histórico de la ruta cubierta por el vehículo monitoreado.

Según el reporte estadístico realizado y actualizado hasta el año 2015 en base a los accidentes de tránsito fatales y no fatales declarados a nivel nacional en las unidades de la PNP (Policía Nacional del Perú), Accidentes Declarados en las Unidades de la PNP (2015), la causa probable de un accidente de tránsito corresponde a uno de los factores detallados en la siguiente tabla.

Tabla 3.1.

Accidentes de Tránsito Fatales y No Fatales por año, según causa: 2006-2015.

ACCIDENTES DE TRÁNSITO FATALES Y NO FATALES POR AÑO, SEGÚN CAUSA: 2006-2015
(Unidades)

CAUSA	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
TOTAL	**77 840**	**79 972**	**85 337**	**86 026**	**83 653**	**84 495**	**95 692**	**102 762**	**101 307**	**95 532**
%	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Exceso de Velocidad	24 764	24 923	25 543	24 981	26 164	27 129	31 371	33 202	32 919	30 672
%	31.8	31.2	29.9	29.0	31.3	32.1	32.8	32.3	32.5	32.1
Ebriedad del Conductor	7 324	7 555	8 536	9 112	7 303	8 929	10 586	12 021	10 083	7 754
%	9.4	9.4	10.0	10.6	8.7	10.6	11.1	11.7	10.0	8.1
Imprudencia del Conduct	19 776	20 654	22 165	23 390	23 361	23 132	25 533	28 545	28 429	27 552
%	25.4	25.8	26.0	27.2	27.9	27.4	26.7	27.8	28.1	28.8
Imprudencia del Peatón	7 043	7 796	7 332	6 961	7 042	6 407	7 501	8 533	9 515	8 637
%	9.0	9.7	8.6	8.1	8.4	7.6	7.8	8.3	9.4	9.0
Desacato de señales	2 277	1 898	1 602	1 903	2 147	1 747	1 976	2 129	2 498	2 541
%	2.9	2.4	1.9	2.2	2.6	2.1	2.1	2.1	2.5	2.7
Falla Mecánica	2 306	2 297	2 547	2 343	2 077	2 322	2 389	2 380	2 098	1 905
%	3.0	2.9	3.0	2.7	2.5	2.7	2.5	2.3	2.1	2.0
Mal Estado de la Pista	0 976	1 082	1 505	1 287	1 101	1 225	1 662	1 781	1 795	1 660
%	1.3	1.4	1.8	1.5	1.3	1.4	1.7	1.7	1.8	1.7
Señalización Defectuosa	646	740	921	833	700	856	837	796	739	637
%	0.8	0.9	1.1	1.0	0.8	1.0	0.9	0.8	0.7	0.7
Otros	12 728	13 027	15 186	15 216	13 758	12 748	13 837	13 375	13 231	14 174
%	16.4	16.3	17.8	17.7	16.4	15.1	14.5	13.0	13.1	14.8

Cobertura: Nacional

Fuente: (Accidentes Declarados en las Unidades de la PNP, 2015).

En la presente tesis, no se analizará todas las causas de los accidentes de tránsito, debido a que, el propósito de esta investigación se enfoca en descubrir los distintos incidentes ocultos registrados en un determinado viaje realizado por una unidad de transporte; los factores, exceso de velocidad y desacato de señales son determinantes en la detección de incidentes tal y como se muestra en la figura 3.1.

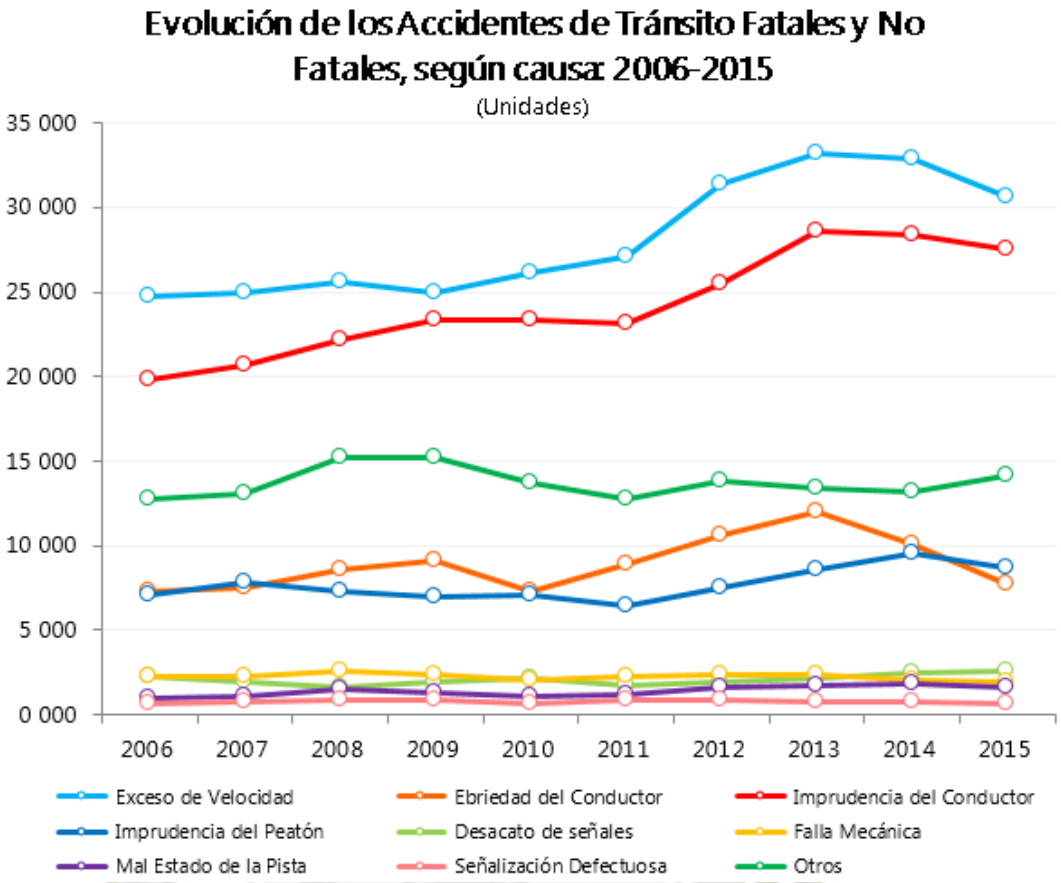


Figura 3.1. Evolución de los Accidentes de Tránsito Fatales y No Fatales, según causa: 2006-2015.

Fuente: (Accidentes Declarados en las Unidades de la PNP, 2015).

Una de las necesidades primordiales de una empresa de transporte, es brindar un excelente servicio a los distintos usuarios a nivel de seguridad; para ello se tiene que conocer el verdadero nivel de desempeño de sus conductores, esta acción va más allá de evaluar a un conductor en base al cumplimiento de los horarios de salida y llegada en los viajes que realizó. En caso de un reclamo o queja por parte de un consumidor de dicho servicio, este cuenta con el derecho de proceder a solicitar y llenar el libro de reclamaciones de la empresa en mención y esta acción representa una forma clara de evidenciar un hecho ocurrido o incidente

registrado en un determinado viaje, esta medida en conjunto con un buzón de sugerencias representa una posibilidad de adquirir información útil para el nivel estratégico de una empresa. Los sistemas de monitoreo utilizados en los terminales terrestres se muestran como una herramienta de apoyo para realizar un mejor control del desplazamiento vehicular a nivel de transmisiones GPS; lamentablemente dichos sistemas, no cuentan con la funcionalidad para registrar viajes realizados y asignar dichas transmisiones según corresponda a un viaje determinado, permitiendo evaluar o distinguir el desplazamiento a nivel de viajes registrados.

Actualmente, la mayoría de sistemas utilizados por las empresas de transporte terrestre a nivel nacional, más allá de visualizar la última ubicación de todas sus unidades monitoreadas, por lo general, emplean dichos sistemas para realizar la consulta del estado actual de una determinada unidad de transporte; y solo en calidad de investigación o control, realizan un post-monitoreo, para confrontar dicho desplazamiento con la hoja de ruta predeterminada y personalizada según la política de la empresa. Sin embargo, al pretender realizar la distinción del recorrido vehicular registrado entre una serie de *n* viajes entre dos ciudades o más, podemos encontrarnos con una situación cuestionable, incomoda y/o engorrosa; muchas veces, la empresa de transportes necesita distinguir y evaluar todos los viajes realizados correspondientes a un intervalo de fechas, y para realizar esta tarea, hace uso del archivo exportado mediante el sistema de monitoreo contratado por requerimiento de SUTRAN y se utilizan las funciones matemáticas y lógicas que brinda el software de hojas de cálculo denominado Excel, mediante

esta aplicación, son tratadas las distintas transmisiones para obtener la velocidad promedio registrada, excesos continuos de velocidad entre transmisiones y otros indicadores para finalmente guardar los datos representantes de cada viaje realizado en una hoja o archivo.

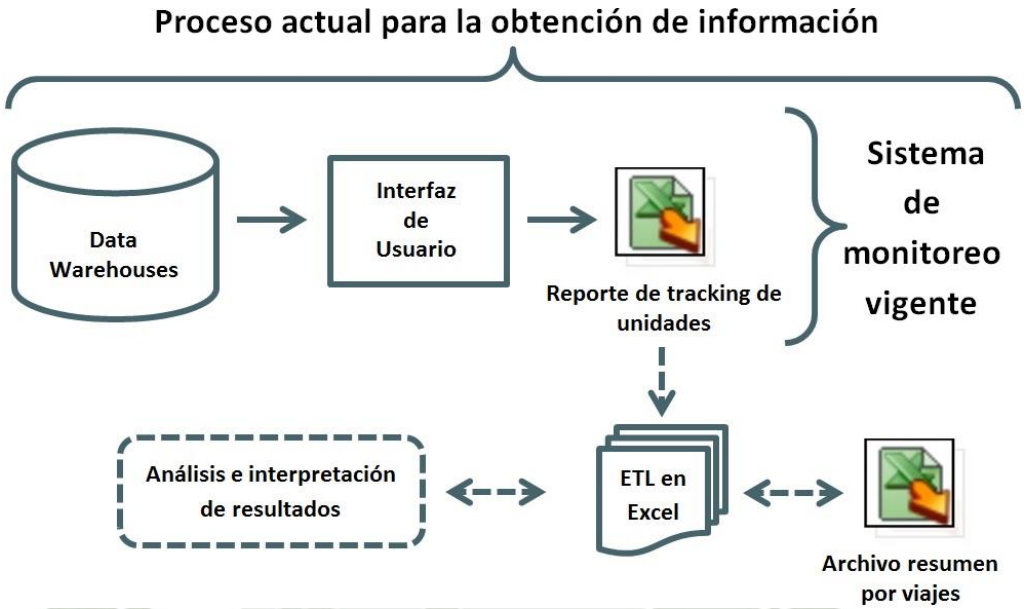


Figura 3.2. Proceso actual para la obtención de información.
Fuente: Elaboración propia.

La solución a este problema está basada en el proceso de descubrimiento de conocimiento en bases de datos o KDD, y consiste, en aplicar el algoritmo de clasificación por vecindad en conjunto con las reglas de decisión obtenidas por el algoritmo de aprendizaje de árboles. La obtención de resultados ideales o aceptables dependerá del nivel de preparación e integridad de los datos a procesar con la aplicación de los algoritmos mencionados anteriormente en el capítulo 2, los cuales, en conjunto con las muestras de entrenamiento a emplear, permitirán

obtener el vector final a evaluar mediante las reglas de decisión que pueden permitir la obtención de información oculta, predecible e importante.

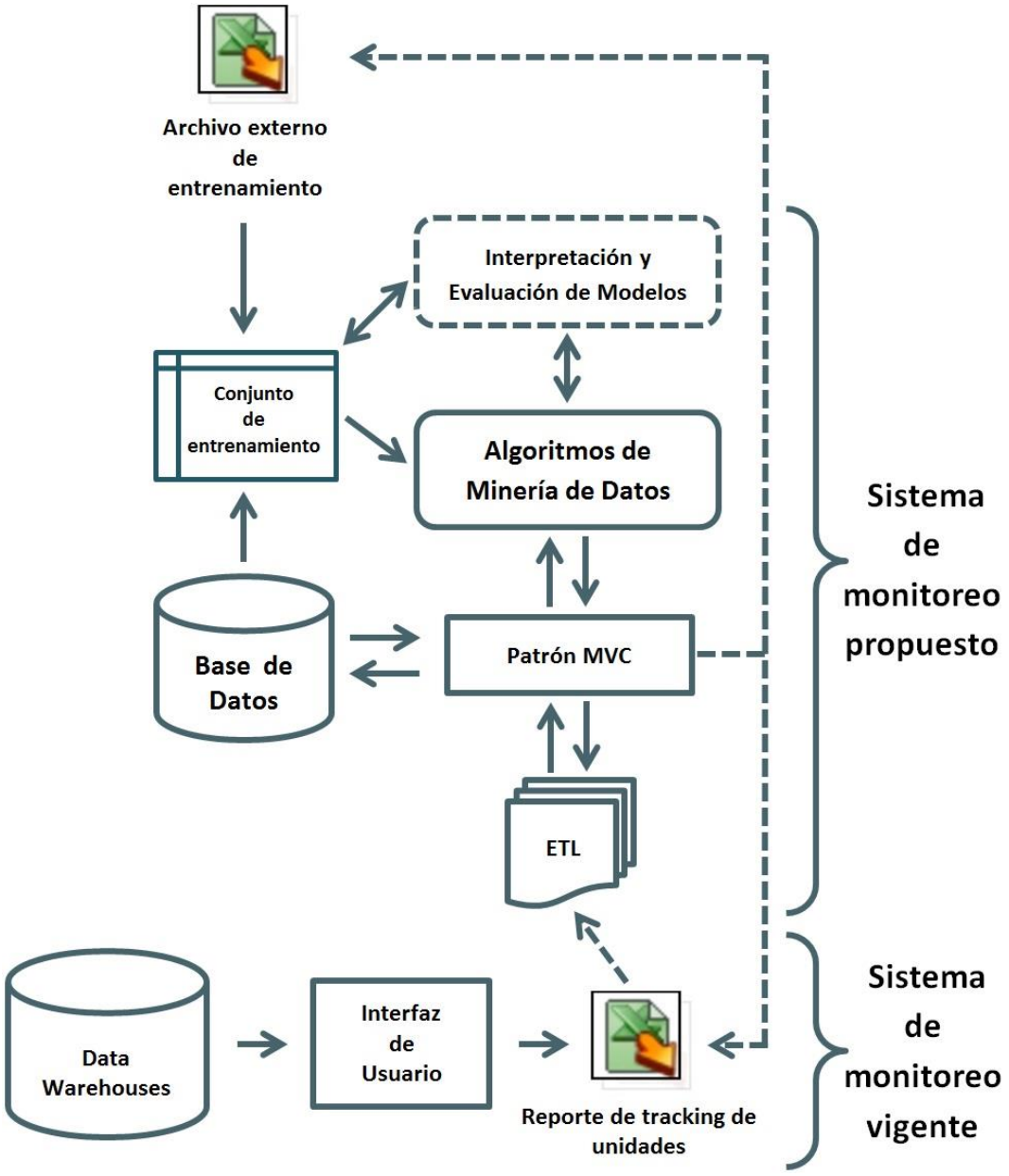


Figura 3.3. Proceso del sistema de monitoreo propuesto para la obtención de nueva información.

Fuente: Elaboración propia.

3.2. Fase 2: Comprensión de los datos.

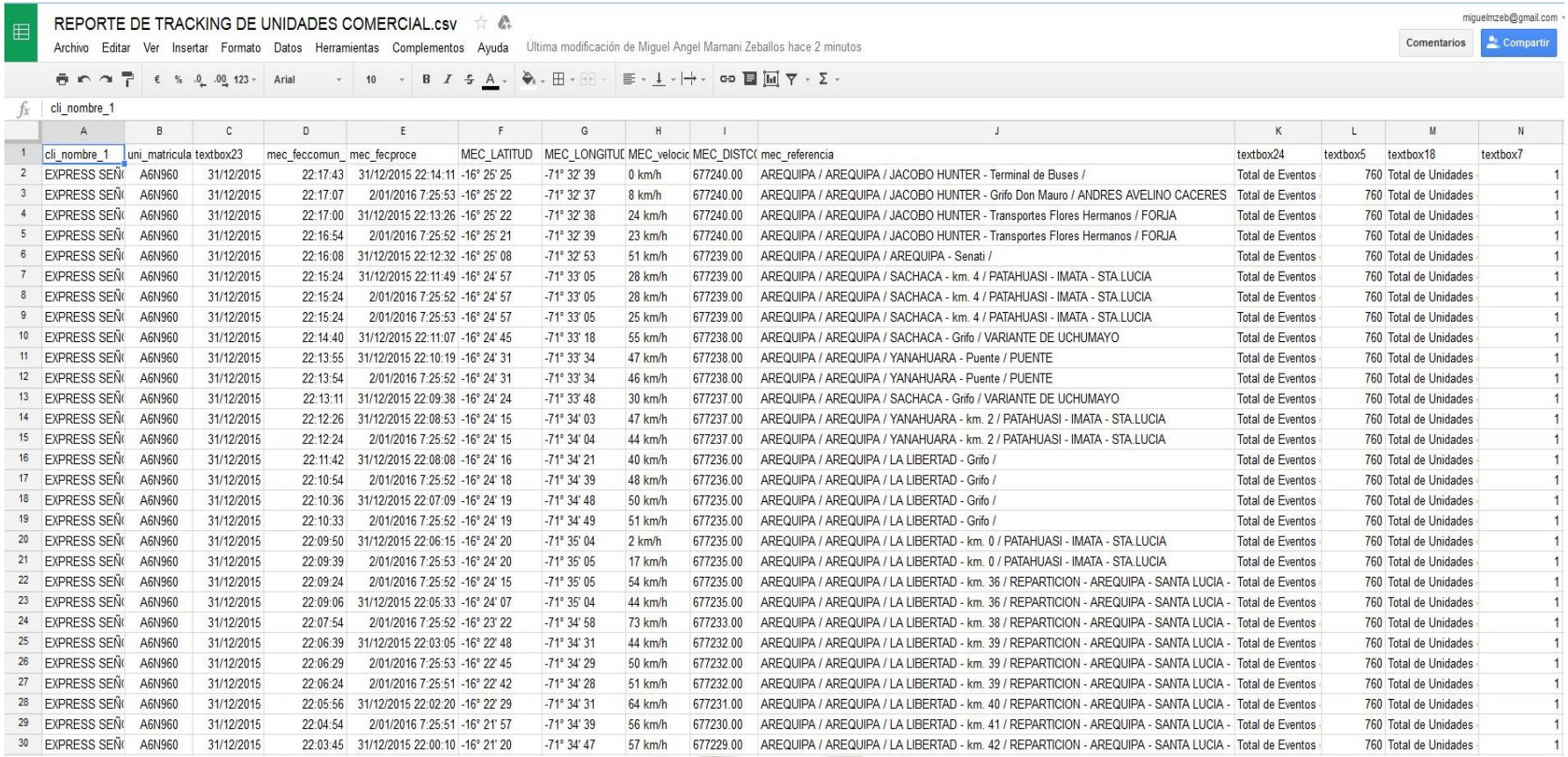
La recolección de los datos iniciales a clasificar, es posible mediante el módulo de reportes del sistema de monitoreo vigente de la empresa contratada para brindar dicho servicio; este módulo, denominado Reporte de Tracking de Unidades (2016), según a un determinado rango de fechas asignado para una o varias unidades de transporte establecido por el usuario, genera un archivo exportado en el formato de CSV como se muestra en la figura 3.4.

Un reporte, contiene las distintas transmisiones existentes pertenecientes a los intervalos de tiempo en que fueron activados los dispositivos de monitoreo y no corresponden necesariamente a la cobertura de una ruta y poseen campos tales como: fecha, hora, latitud, longitud, velocidad (Km/h), odómetro (km) y referencia del lugar en que fue procesada cada una de las transmisiones registradas en intervalos discontinuos de tiempo según la programación, el modelo o el nivel de señal del dispositivo GPS.

La estructura de los datos iniciales está distribuida como se muestra en la figura 3.4 y los campos corresponden como sigue:

- cli_nombre_1 (nombre del cliente).
- uni_matricula_1 (placa de la unidad de transporte).
- textbox23 (fecha de la transmisión).
- mec_feccomun_1 (hora de la transmisión).

- mec_fecproce (fecha y hora de envío de la transmisión).
- MEC_LATITUD (latitud de la transmisión).
- MEC_LONGITUD (longitud de la transmisión).
- MEC_velocidad (velocidad instantánea enviada en la transmisión).
- MEC_DISTCOSTA (odómetro).
- mec_referencia (lugar de referencia de la transmisión).
- textbox5 (total de transmisiones existentes en el archivo origen).



REPORTE DE TRACKING DE UNIDADES COMERCIAL.csv

Archivo Editar Ver Insertar Formato Datos Herramientas Complementos Ayuda Última modificación de Miguel Angel Mamani Zeballos hace 2 minutos

miguelmzeb@gmail.com

Comentarios Compartir

Arial 10

cli_nombre_1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	cli_nombre_1	uni_matricula	textbox23	mec_feccomun_	mec_fecproce	MEC_LATITUD	MEC_LONGITUD	MEC_velocic	MEC_DISTC(	mec_referencia	textbox24	textbox5	textbox18	textbox7
2	EXPRESS SEÑ(	A6N960	31/12/2015	22:17:43	31/12/2015 22:14:11	-16° 25' 25	-71° 32' 39	0 km/h	677240.00	AREQUIPA / AREQUIPA / JACOBO HUNTER - Terminal de Buses /	Total de Eventos	760	Total de Unidades	1
3	EXPRESS SEÑ(	A6N960	31/12/2015	22:17:07	2/01/2016 7:25:53	-16° 25' 22	-71° 32' 37	8 km/h	677240.00	AREQUIPA / AREQUIPA / JACOBO HUNTER - Grifo Don Mauro / ANDRES AVELINO CACERES	Total de Eventos	760	Total de Unidades	1
4	EXPRESS SEÑ(	A6N960	31/12/2015	22:17:00	31/12/2015 22:13:26	-16° 25' 22	-71° 32' 38	24 km/h	677240.00	AREQUIPA / AREQUIPA / JACOBO HUNTER - Transportes Flores Hermanos / FORJA	Total de Eventos	760	Total de Unidades	1
5	EXPRESS SEÑ(	A6N960	31/12/2015	22:16:54	2/01/2016 7:25:52	-16° 25' 21	-71° 32' 39	23 km/h	677240.00	AREQUIPA / AREQUIPA / JACOBO HUNTER - Transportes Flores Hermanos / FORJA	Total de Eventos	760	Total de Unidades	1
6	EXPRESS SEÑ(	A6N960	31/12/2015	22:16:08	31/12/2015 22:12:32	-16° 25' 08	-71° 32' 53	51 km/h	677239.00	AREQUIPA / AREQUIPA / AREQUIPA - Senati /	Total de Eventos	760	Total de Unidades	1
7	EXPRESS SEÑ(	A6N960	31/12/2015	22:15:24	31/12/2015 22:11:49	-16° 24' 57	-71° 33' 05	28 km/h	677239.00	AREQUIPA / AREQUIPA / SACHACA - km. 4 / PATAHUASI - IMATA - STA.LUCIA	Total de Eventos	760	Total de Unidades	1
8	EXPRESS SEÑ(	A6N960	31/12/2015	22:15:24	2/01/2016 7:25:52	-16° 24' 57	-71° 33' 05	28 km/h	677239.00	AREQUIPA / AREQUIPA / SACHACA - km. 4 / PATAHUASI - IMATA - STA.LUCIA	Total de Eventos	760	Total de Unidades	1
9	EXPRESS SEÑ(	A6N960	31/12/2015	22:15:24	2/01/2016 7:25:53	-16° 24' 57	-71° 33' 05	25 km/h	677239.00	AREQUIPA / AREQUIPA / SACHACA - km. 4 / PATAHUASI - IMATA - STA.LUCIA	Total de Eventos	760	Total de Unidades	1
10	EXPRESS SEÑ(	A6N960	31/12/2015	22:14:40	31/12/2015 22:11:07	-16° 24' 45	-71° 33' 18	55 km/h	677238.00	AREQUIPA / AREQUIPA / SACHACA - Grifo / VARIANTE DE UCHUMAYO	Total de Eventos	760	Total de Unidades	1
11	EXPRESS SEÑ(	A6N960	31/12/2015	22:13:55	31/12/2015 22:10:19	-16° 24' 31	-71° 33' 34	47 km/h	677238.00	AREQUIPA / AREQUIPA / YANAHUARA - Puente / PUENTE	Total de Eventos	760	Total de Unidades	1
12	EXPRESS SEÑ(	A6N960	31/12/2015	22:13:54	2/01/2016 7:25:52	-16° 24' 31	-71° 33' 34	46 km/h	677238.00	AREQUIPA / AREQUIPA / YANAHUARA - Puente / PUENTE	Total de Eventos	760	Total de Unidades	1
13	EXPRESS SEÑ(	A6N960	31/12/2015	22:13:11	31/12/2015 22:09:38	-16° 24' 24	-71° 33' 48	30 km/h	677237.00	AREQUIPA / AREQUIPA / SACHACA - Grifo / VARIANTE DE UCHUMAYO	Total de Eventos	760	Total de Unidades	1
14	EXPRESS SEÑ(	A6N960	31/12/2015	22:12:26	31/12/2015 22:08:53	-16° 24' 15	-71° 34' 03	47 km/h	677237.00	AREQUIPA / AREQUIPA / YANAHUARA - km. 2 / PATAHUASI - IMATA - STA.LUCIA	Total de Eventos	760	Total de Unidades	1
15	EXPRESS SEÑ(	A6N960	31/12/2015	22:12:24	2/01/2016 7:25:52	-16° 24' 15	-71° 34' 04	44 km/h	677237.00	AREQUIPA / AREQUIPA / YANAHUARA - km. 2 / PATAHUASI - IMATA - STA.LUCIA	Total de Eventos	760	Total de Unidades	1
16	EXPRESS SEÑ(	A6N960	31/12/2015	22:11:42	31/12/2015 22:08:08	-16° 24' 16	-71° 34' 21	40 km/h	677236.00	AREQUIPA / AREQUIPA / LA LIBERTAD - Grifo /	Total de Eventos	760	Total de Unidades	1
17	EXPRESS SEÑ(	A6N960	31/12/2015	22:10:54	2/01/2016 7:25:52	-16° 24' 18	-71° 34' 39	48 km/h	677236.00	AREQUIPA / AREQUIPA / LA LIBERTAD - Grifo /	Total de Eventos	760	Total de Unidades	1
18	EXPRESS SEÑ(	A6N960	31/12/2015	22:10:36	31/12/2015 22:07:09	-16° 24' 19	-71° 34' 48	50 km/h	677235.00	AREQUIPA / AREQUIPA / LA LIBERTAD - Grifo /	Total de Eventos	760	Total de Unidades	1
19	EXPRESS SEÑ(	A6N960	31/12/2015	22:10:33	2/01/2016 7:25:52	-16° 24' 19	-71° 34' 49	51 km/h	677235.00	AREQUIPA / AREQUIPA / LA LIBERTAD - Grifo /	Total de Eventos	760	Total de Unidades	1
20	EXPRESS SEÑ(	A6N960	31/12/2015	22:09:50	31/12/2015 22:06:15	-16° 24' 20	-71° 35' 04	2 km/h	677235.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 0 / PATAHUASI - IMATA - STA.LUCIA	Total de Eventos	760	Total de Unidades	1
21	EXPRESS SEÑ(	A6N960	31/12/2015	22:09:39	2/01/2016 7:25:53	-16° 24' 20	-71° 35' 05	17 km/h	677235.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 0 / PATAHUASI - IMATA - STA.LUCIA	Total de Eventos	760	Total de Unidades	1
22	EXPRESS SEÑ(	A6N960	31/12/2015	22:09:24	2/01/2016 7:25:52	-16° 24' 15	-71° 35' 05	54 km/h	677235.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 36 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1
23	EXPRESS SEÑ(	A6N960	31/12/2015	22:09:06	31/12/2015 22:05:33	-16° 24' 07	-71° 35' 04	44 km/h	677235.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 36 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1
24	EXPRESS SEÑ(	A6N960	31/12/2015	22:07:54	2/01/2016 7:25:52	-16° 23' 22	-71° 34' 58	73 km/h	677233.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 38 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1
25	EXPRESS SEÑ(	A6N960	31/12/2015	22:06:39	31/12/2015 22:03:05	-16° 22' 48	-71° 34' 31	44 km/h	677232.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 39 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1
26	EXPRESS SEÑ(	A6N960	31/12/2015	22:06:29	2/01/2016 7:25:53	-16° 22' 45	-71° 34' 29	50 km/h	677232.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 39 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1
27	EXPRESS SEÑ(	A6N960	31/12/2015	22:06:24	2/01/2016 7:25:51	-16° 22' 42	-71° 34' 28	51 km/h	677232.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 39 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1
28	EXPRESS SEÑ(	A6N960	31/12/2015	22:05:56	31/12/2015 22:02:20	-16° 22' 29	-71° 34' 31	64 km/h	677231.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 40 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1
29	EXPRESS SEÑ(	A6N960	31/12/2015	22:04:54	2/01/2016 7:25:51	-16° 21' 57	-71° 34' 39	56 km/h	677230.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 41 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1
30	EXPRESS SEÑ(	A6N960	31/12/2015	22:03:45	31/12/2015 22:00:10	-16° 21' 20	-71° 34' 47	57 km/h	677229.00	AREQUIPA / AREQUIPA / LA LIBERTAD - km. 42 / REPARTICION - AREQUIPA - SANTA LUCIA -	Total de Eventos	760	Total de Unidades	1

Figura 3.4. Estructura de los datos de origen.

Fuente: (Reporte de Tracking de Unidades, 2016).

Cada uno de los datos registrados en las "*n* transmisiones" del dispositivo GPS conforma un vector de datos 100% completo (i.e., que ningún campo de cualquiera de los *n* registros o filas en el archivo de datos de origen está en blanco o nulo); esto se debe a que los distintos dispositivos están homologados y estandarizados; además de contar con una precisión muy exacta en cuanto a la ubicación de geoposicionamiento, por lo tanto, la presencia de datos outlier es nula.

La calidad de los datos es muy buena debido a que no existen valores perdidos y el formato de fechas está unificado para todos los registros.

Al explorar más detenidamente los datos iniciales, se aprecia que existen campos que son irrelevantes (e.g., el campo que indica el número total de transmisiones existentes en el archivo origen en forma repetitiva para todos los registros); otros campos necesitan una conversión de nominal a numérico, debido a que en su estado original no se pueden procesar numéricamente (las coordenadas de posicionamiento deben estar en números decimales); por lo señalado, es realmente necesario realizar un proceso de transformación de datos, el mismo que permitirá la generación de nuevos campos necesarios como transmision_tipo_velocidad (varchar), transmision_exceso_velocidad (boolean), transmision_incidente (boolean), entre otros campos que en la actualidad no son considerados en el sistema de monitoreo satelital vigente; considerando que ya son más de 6 años a la actualidad que se espera más de una mejora en dicho servicio ofertado en el Perú.

3.3. Fase 3: Preparación de los datos.

Debido a que la preparación de los datos es fundamental para realizar una óptima minería de datos, se deben considerar los siguientes pasos:

- Definir el conjunto de vecinos más cercanos.
- Definir el conjunto de entrenamiento de reglas de decisión.
- Definir los reportes recopilados correspondientes a los viajes a evaluar, para habilitar el almacén de datos.
- Realizar el proceso de ETL para importar los registros de monitoreo necesarios.
- Integrar con la ayuda del algoritmo NN el vector de datos necesario para aplicar las reglas de decisión.

La definición de los conjuntos de entrenamiento es fundamental para realizar una minería de datos ideal en la siguiente fase, por lo cual, dichos conjuntos están sujetos a ser reevaluados en más de una fase de la metodología CRISP-DM.

Los datos correspondientes a los distintos desplazamientos de un vehículo se encuentran en archivos de reportes estandarizados y distribuidos en forma de una sola tabla. Todos los registros o transmisiones en los archivos tracking de unidades, corresponden al seguimiento de una unidad identificada por su número de

matrícula, fecha, hora, geolocalización y velocidad instantánea almacenada en el formato de archivo CSV.

Debido a que no es factible contar con el acceso a la DB del sistema de monitoreo vigente, por temas relacionados con la seguridad; resulta necesario crear el almacén de datos a partir de los reportes de tracking de unidades exportados por el sistema de monitoreo satelital vigente para simular la recepción de transmisiones GPS. Para que la importación de datos sea exitosa, el sistema propuesto cuenta con un módulo encargado del proceso ETL tal y como se detalla en la figura 3.5.

El proceso de ETL a emplear realiza la migración de los datos a clasificar previamente transformados con un algoritmo de integración de datos, el mismo que, se encargará principalmente de eliminar campos irrelevantes y crear o modificar los valores o campos para una adaptación ideal de los algoritmos complementarios; previo al almacenamiento en la tabla transmisiones, cada registro del tracking de unidades es validado por el campo uni_matricula_1 (varchar) que posee una relación con la tabla vehículos a través del campo vehiculo_placa (varchar), mediante consultas SQL el sistema obtiene el valor correspondiente al campo vehiculo_id (integer) que referencia numéricamente al valor contenido semánticamente en el campo uni_matricula_1 (varchar). De la estructuración del proceso ETL, dependerá la modificación y creación de los campos fundamentales como: transmision_latitud (decimal), transmision_longitud (decimal).

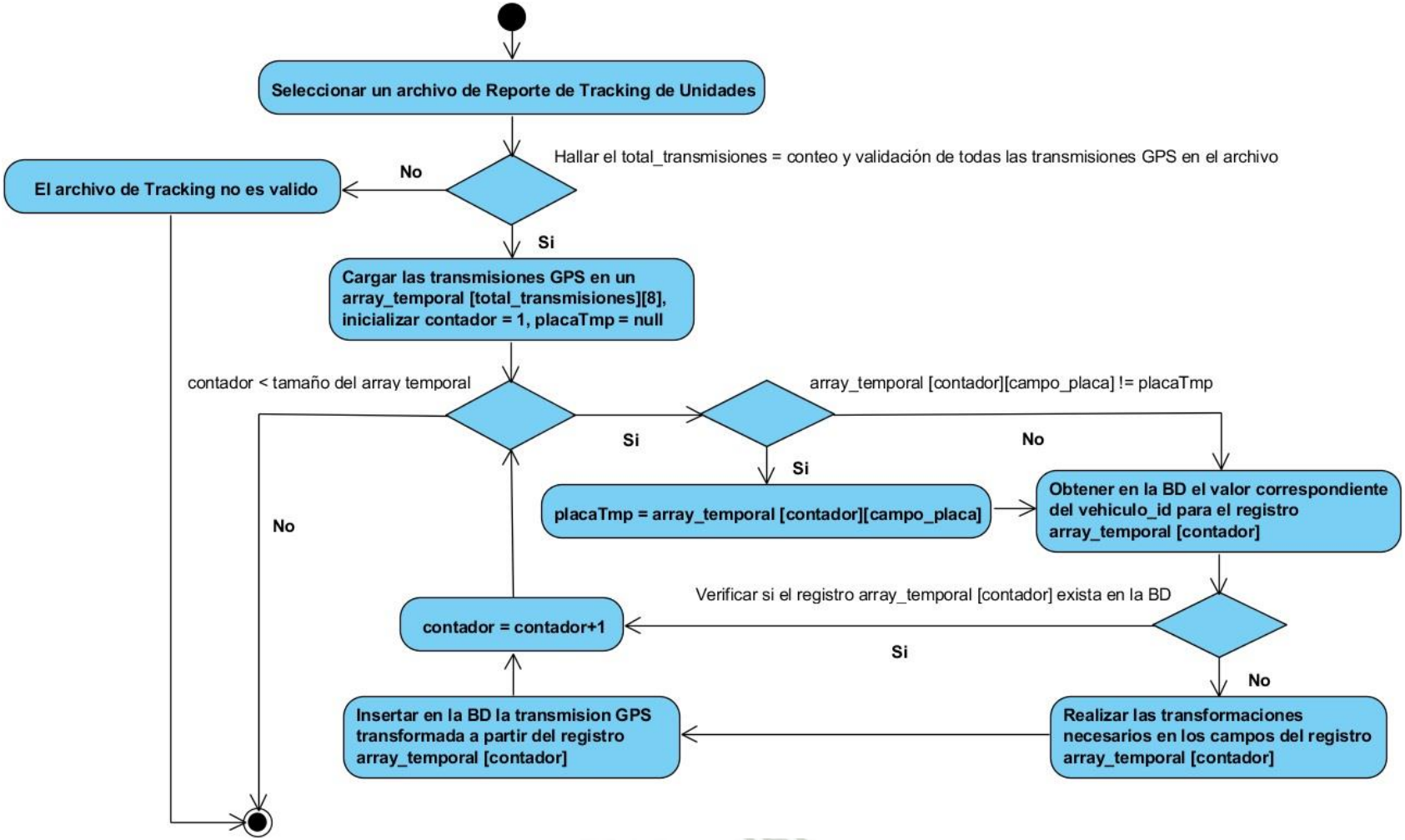


Figura 3.5. Pseudocódigo del proceso de ETL para la migración de transmisiones.

Fuente: Elaboración propia.

El diseño de la base de datos local MySQL, tal y como muestra la figura 3.6 posee restricciones para permitir el almacenamiento de los m nuevos registros de transmisiones refinados y sin duplicados a partir de las *n* transmisiones existentes en el archivo de origen. El diseño necesario de la DB que soporte el flujo de trabajo para el sistema propuesto, requiere contar con las siguientes tablas, las mismas que, permiten organizar los datos necesarios para realizar un flujo ideal en el sistema propuesto en el presente proyecto: transmisiones, vehículos, geopuntos, rutas, viajes, trabajadores, cargos y usuarios.

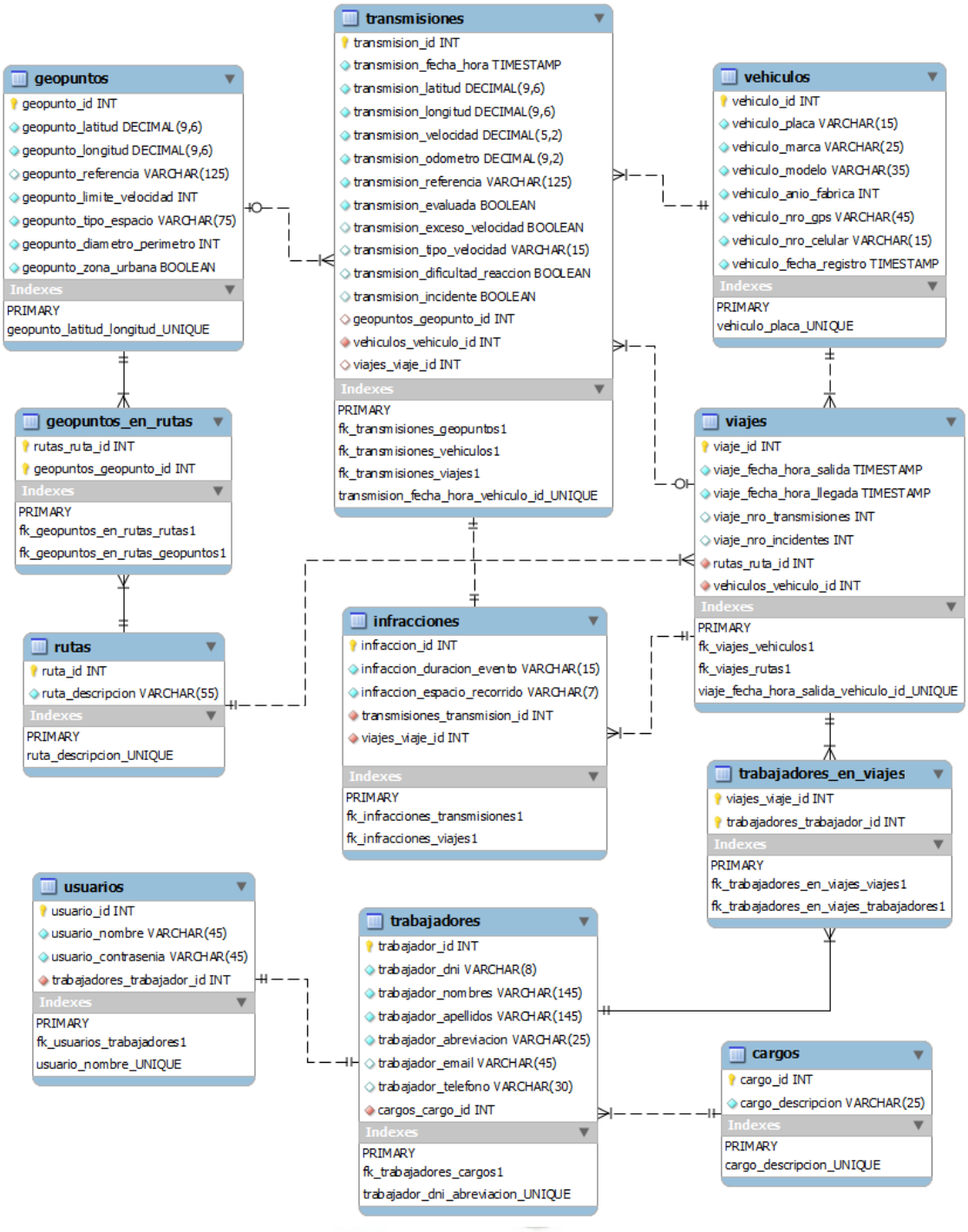


Figura 3.6. Diagrama entidad relación de la base de datos “monitoreo”.

Fuente: Elaboración propia.

El sistema hace uso del algoritmo NN en base al conjunto de entrenamiento perteneciente a la tabla geopuntos, para lograr la obtención el vector de datos necesario para aplicar las reglas de decisión; este algoritmo será detallado en la siguiente fase.

3.4. Fase 4: Modelado.

Para la realización de esta fase se emplean las técnicas de minería de datos predictivas que especifican el modelo para los datos en base a un conocimiento teórico previo. El modelo supuesto para los datos debe contrastarse después del proceso de minería de datos antes de aceptarlo como válido.

El modelo de clasificación empleado en esta tesis, requiere del uso en conjunto de dos algoritmos, el algoritmo NN que realizará un análisis discriminante que permite asignar o clasificar nuevos individuos dentro de grupos previamente definidos, y el algoritmo ID3 que permitirá realizar la codificación de las reglas de decisiones a través de la "estrategia de aprendizaje con el algoritmo ID3", mediante las mismas será posible clasificar los datos en grupos basados en los valores de las variables.

Para (Pérez & Santín, 2007), todo modelo debe superar la fase de identificación objetiva (a partir de los datos se aplican reglas que permitan identificar el mejor modelo posible que ajuste los datos), estimación (proceso de cálculo de los parámetros del modelo elegido en la fase de identificación), diagnosis (proceso de

contraste de la validez del modelo estimado) y predicción (proceso de utilización del modelo identificado, estimado y validado para predecir valores futuros de las variables dependientes). En algunos casos el modelo se obtiene como mezcla del conocimiento obtenido antes y después del Data Mining y también debe contrastarse antes de aceptarse como válido. Tanto los árboles de decisión como las redes neuronales y el análisis discriminante son a su vez técnicas de clasificación que pueden extraer perfiles de comportamiento o clases.

Para que el algoritmo NN realice la clasificación más acertada de correspondencia, esta depende y es directamente proporcional al tamaño del universo del conjunto de entrenamiento o geopuntos registrados y asociados a una o varias rutas; sería ideal utilizar un conjunto de entrenamiento de similar tamaño para generar las reglas de decisión a partir de datos relacionados a los accidentes de tránsito registrados en el Perú, pero actualmente esa es una de las debilidades a afrontar según el Plan Nacional de Seguridad Vial 2015-2024 desarrollado por el Consejo Nacional de Seguridad Vial (2015), el mismo que detalla que existe una deficiente recopilación de datos de accidentes de tránsito, así como un déficit en la calidad de datos recopilados de accidentes de tránsito. Ante esta situación, en este plan de tesis se utiliza un conjunto de entrenamiento relativamente pequeño para obtener las reglas de decisión, a pesar de ello, ese conjunto considera todas las combinaciones lógicas posibles en las cuales pueda existir riesgo.

Hay ciertas técnicas y algoritmos que no generan un modelo como tal. En (Lara, 2014), podemos encontrar, que las técnicas de clasificación basadas en árboles de

decisión sí que construyen un modelo. Dicho modelo es, precisamente, el árbol de decisión.

Finalmente, para diseñar el modelo de clasificación que más se ajuste, debemos guiarnos del objetivo principal de este proyecto, para determinar así las regiones críticas en una ruta de transporte terrestre; a continuación se muestran las técnicas de aprendizaje supervisado a utilizar y el objetivo que persigue cada una de ellas, considerando además el diseño de proceso necesario en la construcción del modelado del sistema en general:

3.4.1. Algoritmos

Estos algoritmos permiten completar el vector de datos o campos necesarios en los *m* nuevos registros de transmisiones almacenadas con resultado del proceso ETL, todas las transmisiones nuevas son únicas y deben ser asignadas a un viaje previamente registrado en el sistema. Para lograr integrar el vector final, se requiere realizar una clasificación, a partir del vector de datos necesario para aplicar las reglas de decisión para cada registro de transmisión obtenido en la fase de preparación de los datos con la ayuda del algoritmo NN. Con la creación de los campos necesarios para evaluar el desplazamiento de un vehículo en un determinado viaje, el sistema alcanza el objetivo principal propuesto en esta tesis.

A. Algoritmo NN (Distancia Mínima)

El objetivo principal al emplear este algoritmo es encontrar y asignar el geopunto más cercano a cada una de las transmisiones almacenadas por el proceso ETL; este algoritmo, a pesar de no poseer un nivel de complejidad muy alto, necesita un conjunto de entrenamiento muy grande para determinar y asignar de manera efectiva los valores discretos para los siguiente campos: transmision_exceso_velocidad (boolean), transmision_tipo_velocidad (varchar) y transmision_dificultad_reaccion (boolean) en base a los atributos del geopunto asignado para cada una de las transmisiones pertenecientes a un determinado desplazamiento vehicular; el campo geopunto_tipo_espacio {recta, curva, otro(curvas continuas, curva peligrosa, zona urbana)}, reduce la incertidumbre respecto al tipo de espacio o geolocalización instantánea correspondiente para cada transmisión. En la figura 3.7, podemos visualizar el pseudocódigo del algoritmo NN, necesario para obtener los campos mencionados, a partir del geopunto más cercano.

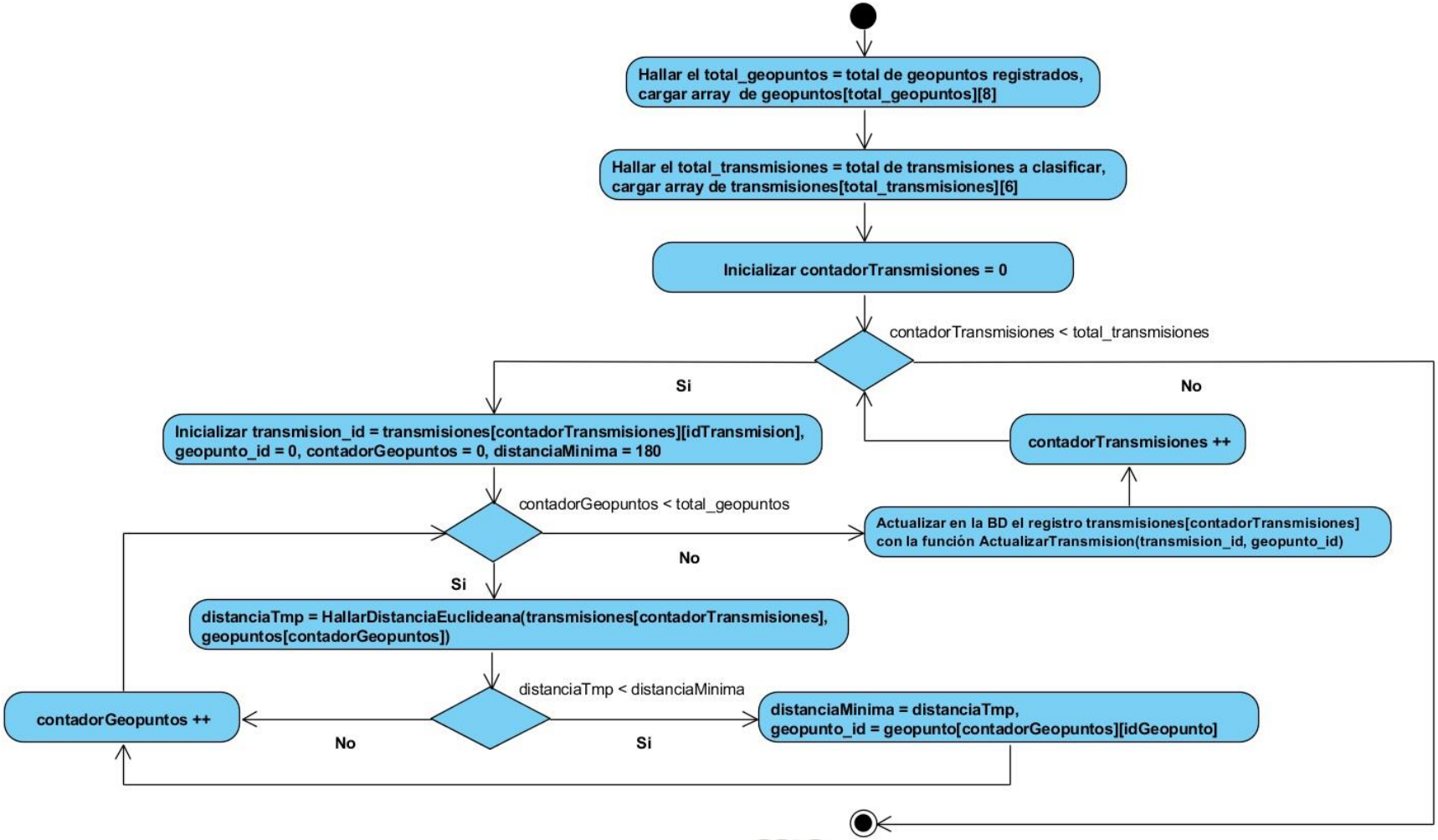


Figura 3.7. Pseudocódigo del algoritmo NN.

Fuente: Elaboración propia.

B. Algoritmo ID3 (Reglas de decisión)

La finalidad de este algoritmo es: clasificar, predecir o determinar las regiones críticas existentes (incidentes) asociadas al campo "riesgo" (+ ó -) que corresponde al nivel de riesgo existente para cada una de las transmisiones almacenadas y vinculadas a un determinado viaje registrado en el sistema propuesto.

Estrategia de aprendizaje con el algoritmo ID3

La estrategia en este algoritmo, es respaldada a partir de los ejemplos que se describen mediante un conjunto de atributos, que en un momento dado toman unos valores concretos. En la figura 3.8 se muestra un conjunto de atributos y sus valores según los tipos de riesgo para su clasificación como región crítica o región no crítica. Las líneas continuas indican que el valor del atributo se corresponde con la clase nivel de riesgo dada, a diferencia de las discontinuas que expresan no coincidencia.

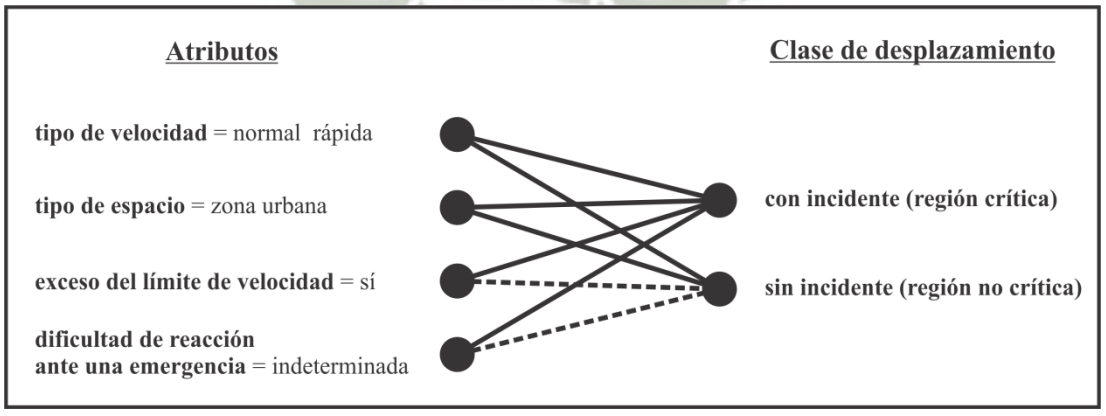


Figura 3.8. Atributos y clases.
Fuente: Elaboración propia.

Del ejemplo de la figura 3.8, podemos aprender la siguiente regla:

$$Si\ (a_1 = val_1) \wedge (a_2 = val_2) \wedge \ldots \wedge (a_n = val_n)\ entonces\ (C_k = val_k)$$

siendo a_1 = tipo de velocidad, a_2 = tipo de espacio, a_3 = exceso de velocidad; mientras que $C_1 = +$ (región crítica) y $C_2 = -$ (región no crítica).

Para el sistema propuesto, las reglas se generan a partir de los ejemplos o muestras de entrenamiento dadas en la tabla 3.2.

Tabla 3.2.

Muestras de entrenamiento.

Muestra	Tipo de velocidad	Tipo de espacio	Exceso del límite de velocidad	Dificultad de reacción ante una emergencia	Clase (región críticas)
1	defensiva	recta	no excede	determinada	NO (sin incidente)
2	defensiva	curva	no excede	determinada	NO (sin incidente)
3	defensiva	otro	no excede	determinada	NO (sin incidente)
4	normal rápida	recta	no excede	determinada	NO (sin incidente)
5	normal rápida	curva	no excede	determinada	NO (sin incidente)
6	normal rápida	otro	no excede	determinada	NO (sin incidente)
7	normal rápida	recta	sí excede	determinada	NO (sin incidente)
8	normal rápida	recta	no excede	indeterminada	NO (sin incidente)
9	normal rápida	curva	no excede	indeterminada	NO (sin incidente)
10	normal rápida	otro	no excede	indeterminada	NO (sin incidente)
11	normal rápida	recta	sí excede	indeterminada	NO (sin incidente)
12	normal rápida	curva	sí excede	indeterminada	SI (con incidente)
13	normal rápida	otro	sí excede	indeterminada	SI (con incidente)
14	excesiva	recta	sí excede	indeterminada	SI (con incidente)
15	excesiva	curva	sí excede	indeterminada	SI (con incidente)
16	excesiva	otro	sí excede	indeterminada	SI (con incidente)

Fuente: Elaboración propia.

En el libro de (Pajares & Santos, 2006) indica que, para construir el árbol el orden de los atributos se elige de mayor a menor capacidad discriminante. Una forma de realizarlo es atendiendo al contenido de información derivado de la teoría de la información. Para ello se definen los siguientes términos,

$$p\text{: } porcentaje\ de\ ejemplos + = \frac{|E+|}{|E+| + |E-|}$$

$$n\text{: } porcentaje\ de\ ejemplos - = \frac{|E-|}{|E+| + |E-|}$$

siendo $|E+|$ y $|E-|$ respectivamente el número de ejemplos positivos y negativos observado. El contenido de información de un conjunto de datos es:

$$infor\ (p,n) = -p\log_2(p) - n\log_2 n$$

siendo una medida de la entropía (grado de desorden) existente. Considerando ahora $0\log_2(0) = 0$. Este concepto está relacionado con la teoría de información en el sentido de que un evento seguro con $p = 1$ no contiene información (información nula). La información está relacionada con la entropía, a mayor entropía menor información. Por tanto, para construir el árbol hay que elegir los atributos con mayor información (menor entropía), que se mide a través de la función de mérito como sigue:

$$Mérito\ de\ un\ atributo\ a_m\text{: } mérito(a_m) = \sum_{i=1}^{n} r_i\ infor(p_i, n_i)$$

siendo:

$$p_i = \%\ de\ ejemplos + en\ la\ rama\ i;$$

$$n_i = \%\ de\ ejemplos - en\ la\ rama\ i;$$

$$r_i = \frac{p_i + n_i}{N}\ porcentaje\ de\ ejemplos\ en\ la\ rama\ i,$$

$$con\ N\ número\ de\ ejemplos,$$

mide la ponderación del grado de desorden de la rama correspondiente con respecto a los ejemplos que contiene. Seguidamente, se construyen las tablas de información,

Tabla 3.3.

Tablas del contenido de información

Exceso del límite de velocidad

no excede = 9	sí excede = 7
$p_1 = 0/9$ $n_1 = 9/9$	$p_2 = 5/7$ $n_2 = 2/7$

Tipo de velocidad

defensiva = 3	normal rápida = 10	excesiva = 3
$p_1 = 0/3$ $n_1 = 3/3$	$p_2 = 2/10$ $n_2 = 8/10$	$p_3 = 3/3$ $n_3 = 0/3$

Dificultad de reacción ante una emergencia

determinada = 7	indeterminada = 9
$p_1 = 0/7$ $n_1 = 7/7$	$p_2 = 5/9$ $n_2 = 4/9$

Tipo de espacio

recta = 6	curva = 5	otro = 5
$p_1 = 1/6$ $n_1 = 5/6$	$p_2 = 2/5$ $n_2 = 3/5$	$p_3 = 2/5$ $n_3 = 3/5$

Fuente: Elaboración propia.

$$mérito(exceso\ de\ velocidad) = 9/16\ \ infor(0/9\,,9/9) + 7/16\ \ infor(5/7\,,2/7)$$
$$= 0.37761$$
$$mérito(tipo\ de\ velocidad) = 3/16\ \ infor(0/3\,,3/3) + 10/16\ \ infor(2/10\,,8/10) + 3/16\ \ infor(3/3\,,0/3)$$
$$= 0.45121$$
$$mérito(dificultad\ de\ reacción) = 7/16\ \ infor(0/7\,,7/7) + 9/16\ \ infor(5/9\,,4/9)$$
$$= 0.55748$$
$$mérito(tipo\ de\ espacio) = 6/16\ \ infor(1/6\,,5/6) + 5/16\ \ infor(2/5\,,3/5) + 5/16\ \ infor(2/5\,,3/5)$$
$$= 0.85060$$

Figura 3.9. Orden de atributos por mérito.

Fuente: Elaboración propia.

En consecuencia, el orden de los atributos por su valor informativo es: exceso de velocidad, tipo de velocidad, estabilidad y finalmente tipo de espacio. En la siguiente figura 3.10 se muestra el árbol que representa los datos.

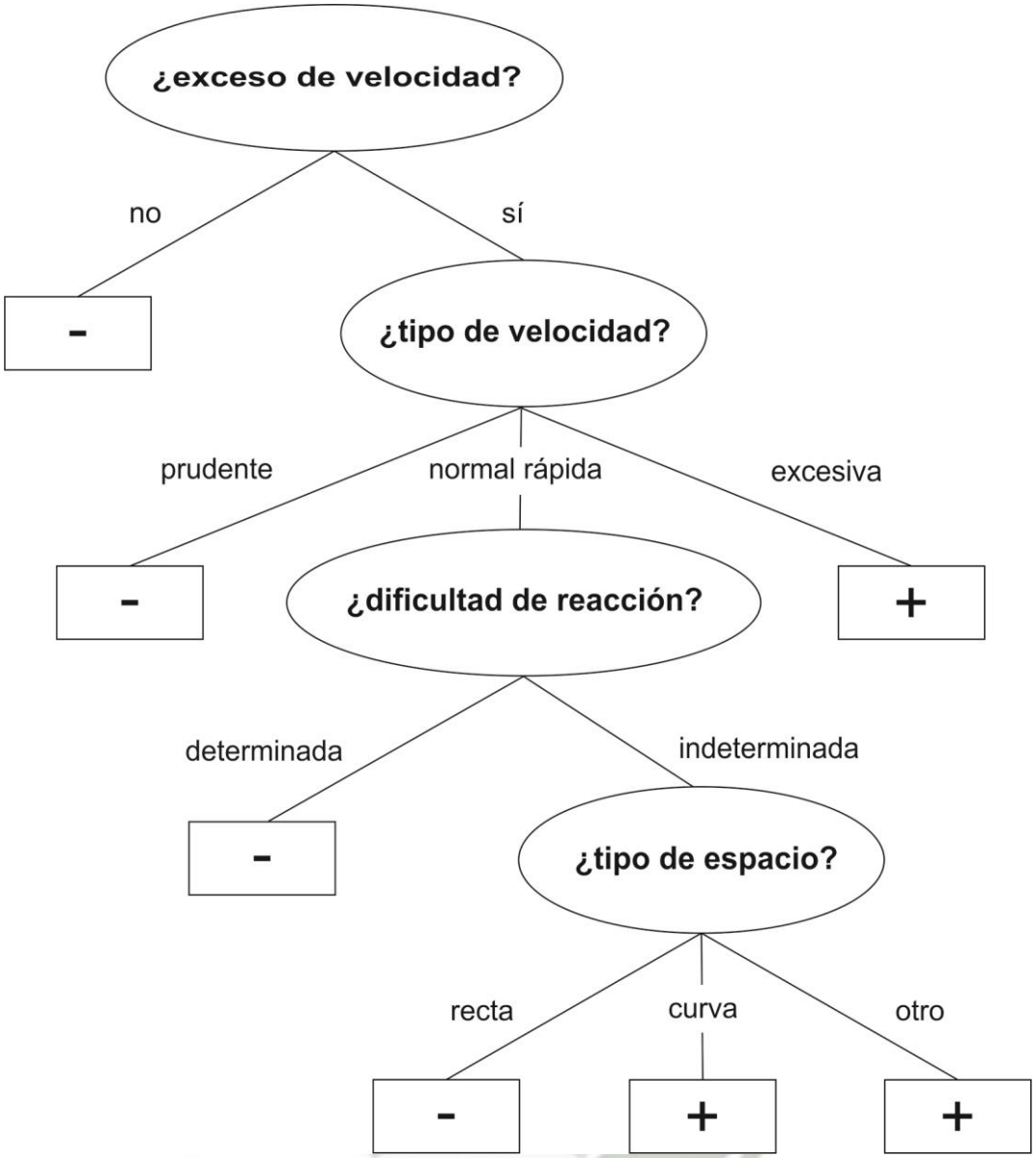


Figura 3.10. Árbol de decisión.
Fuente: Elaboración propia.

El siguiente paso consiste en reflejar ese conocimiento, mediante un conjunto de reglas. Básicamente, cada camino desde la raíz hacia una hoja corresponde a una regla, siendo las hojas las representantes de la una clase según corresponda la conjunción de las decisiones asociadas a una rama del árbol.

El conjunto de reglas correspondiente al árbol de la figura 3.10 es:

- $Si(exceso\ de\ velocidad = no) => (clase = -)$
- $Si(exceso\ de\ velocidad = sí)\ y\ (tipo\ de\ velocidad = prudente) => (clase = -)$
- $Si(exceso\ de\ velocidad = sí)\ y\ (tipo\ de\ velocidad = excesiva) => (clase = +)$
- $Si(exceso\ de\ velocidad = sí)\ y\ (tipo\ de\ velocidad = normal\ rápida)\ y\ (dificultad\ de\ reacción = sí) => (clase = -)$
- $Si(exceso\ de\ velocidad = sí)\ y\ (tipo\ de\ velocidad = normal\ rápida)\ y\ (dificultad\ de\ reacción = no)\ y\ (tipo\ de\ espacio = recta) => (clase = -)$
- $Si(exceso\ de\ velocidad = sí)\ y\ (tipo\ de\ velocidad = normal\ rápida)\ y\ (dificultad\ de\ reacción = no)\ y\ (tipo\ de\ espacio = curva) => (clase = +)$
- $Si(exceso\ de\ velocidad = sí)\ y\ (tipo\ de\ velocidad = normal\ rápida)\ y\ (dificultad\ de\ reacción = no)\ y\ (tipo\ de\ espacio = otro) => (clase = +)$

En el caso de las dos últimas reglas se podrían simplificar utilizando la disyunción.

3.4.2. Esquema del modelo de clasificación de transmisiones

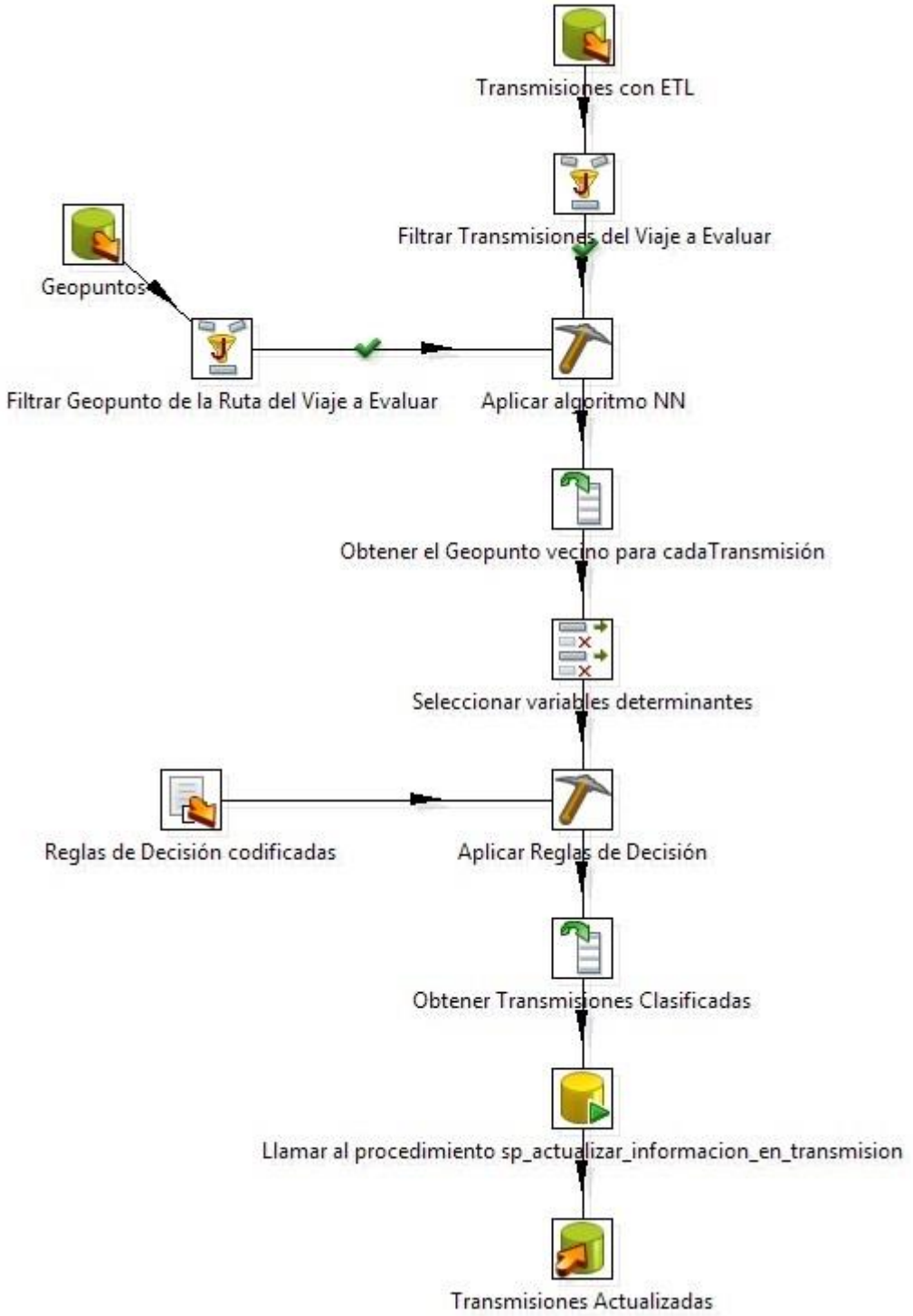


Figura 3.11. Esquema del modelo de clasificación de transmisiones.

Fuente: Elaboración propia.

3.4.3. Análisis, Diseño y Desarrollo del Sistema

Cuando se pretende construir un sistema, es importante considerar una serie de pasos que se deben seguir a fin de obtener un resultado de calidad y que en la Ingeniería de Software se conoce como proceso de desarrollo de software.

Para obtener un desarrollo exitoso, resulta necesario definir el marco de trabajo a utilizar, para este caso, se optó por elegir las fases principales del proceso de desarrollo RUP (Rational Unified Process). En RUP los cinco flujos de trabajo son: requerimientos, análisis, diseño, implementación y prueba, tienen lugar sobre las cuatro fases: concepción, elaboración, construcción y transición.

A. Análisis de requerimientos.

Antes de la etapa de diseño, se deben prever y considerar las características necesarias del sistema propuesto, teniendo en cuenta la experiencia previa del sistema vigente y la retroalimentación de los usuarios finales. Debido a que la especificación de requerimientos a tratar y las estrategias para alcanzarlos son cruciales.

Dentro de la definición del flujo de trabajo requerido para el sistema desarrollado, fue necesario evaluar una serie de técnicas de clasificación supervisada de minería de datos, para determinar cuáles serán las necesarias

a emplear como parte de la solución al problema a tratar en esta tesis; además, es necesario considerar todos los formularios de gestión y registro involucrados en el flujo de trabajo propuesto como una posible mejora para el sistema de monitoreo vigente.

Para obtener las funcionalidades a proponer como parte de una mejora en el procedimiento de monitoreo satelital de vehículos, el nuevo sistema cuenta con una base de datos diseñada para soportar un flujo de trabajo más completo.

El requerimiento principal del sistema, es el de permitir evaluar el nivel de seguridad de un servicio de transporte o viaje realizado a partir de la información de los posibles riesgos o incidentes registrados en dicho viaje. Esta funcionalidad, permitirá a la vez, contar con un resumen del desempeño de los conductores asignados en los viajes realizados.

El sistemas propuesto no cuenta con la funcionalidad de recepción de transmisiones de un equipo GPS; para simular esta acción, el proceso ETL integrado en el sistema desarrollado, carga las transmisiones previamente exportadas en reportes de tracking de unidades generados por el sistema de monitoreo utilizado en la actualidad en el Perú.

Finalmente se tienen que definir los algoritmos y establecer el orden en que serán ejecutados, para ello, el sistema cuenta con un patrón de

arquitectura de software y su propio almacén de datos actualizado por el proceso ETL; la rapidez del sistema respecto a las transacciones procesadas por segundo está respaldada por el uso de hilos.

Para el desarrollo del sistema propuesto, se optó por el uso de las tecnologías orientadas a objetos soportadas por el lenguaje de programación java, así como el gestor de base de datos MySQL y el entorno de desarrollo integrado NetBeans.

B. Diagrama de casos de uso y especificaciones

El diagrama de casos de uso del sistema propuesto está conformado como se detalla en la figura 3.12.

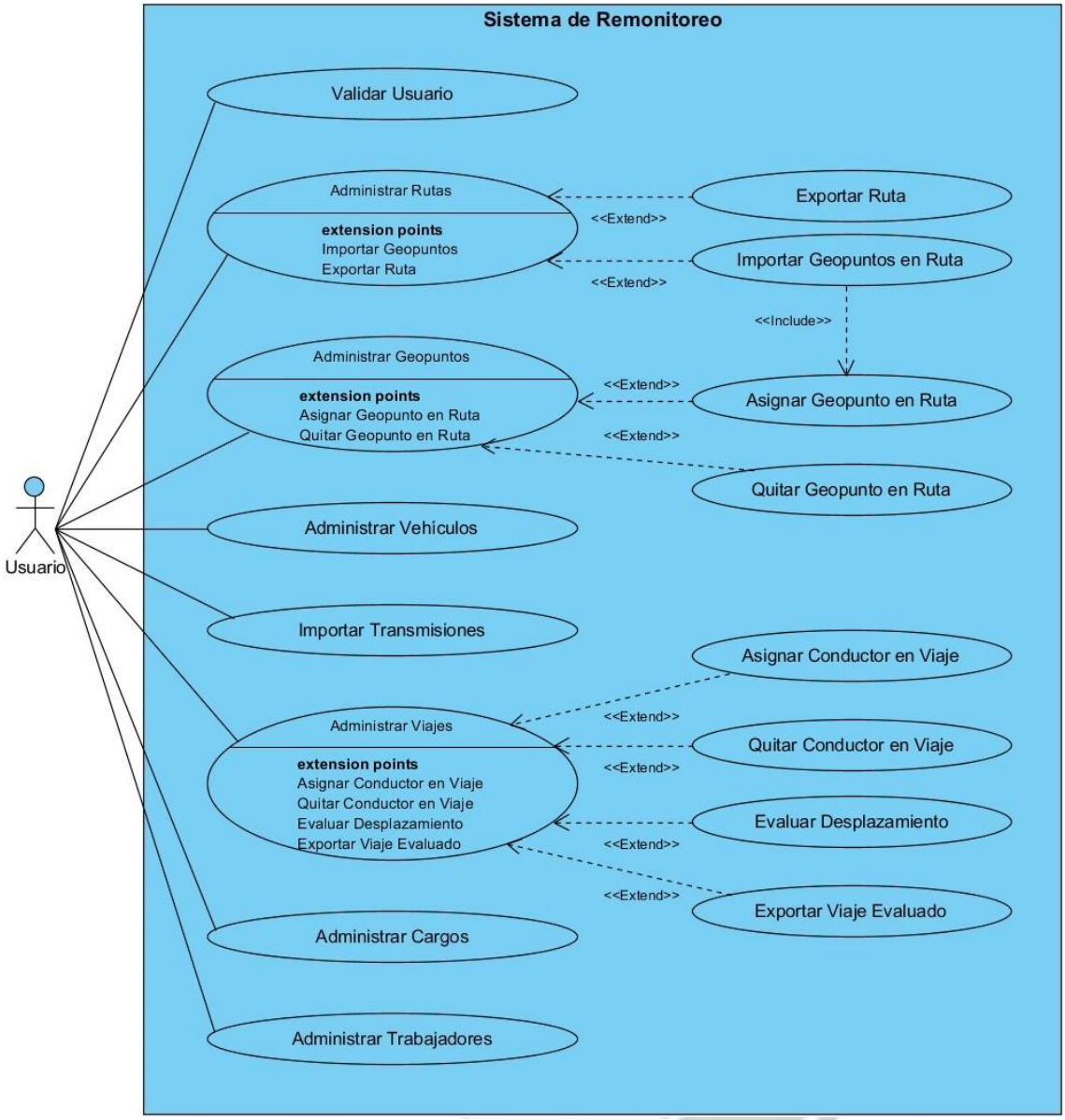


Figura 3.12. Diagrama de casos de uso del sistema propuesto.

Fuente: Elaboración propia.

Las respectivas especificaciones que se va a realizar para el sistema propuesto son:

Tabla 3.4.

Descripción de caso de uso: Validar usuario.

Caso de Uso	**Validar usuario**
Descripción:	Validar el acceso de un usuario al sistema mediante la verificación de las credenciales (nombre y contraseña de usuario).
Actores:	Usuario.
Precondición:	El usuario debe estar registrado en el sistema.
Flujo Principal:	1. El controlador principal muestra la pantalla o formulario de Login. 2. El usuario ingresa las credenciales para acceder al sistema, y pulsa en el botón "Aceptar" para validar el usuario. 3. El controlador del servicio de validación en el Login realiza una petición a la base de datos para confrontar si el usuario existe y además le pertenece dicha contraseña. 4. Si el nombre y la contraseña son válidos el controlador muestra la pantalla o formulario principal del sistema
Subflujos:	Ninguno.
Postcondición:	El usuario accede al sistema.
Excepciones:	4.1. Si el nombre y contraseña no es válido el controlador de validación de usuario muestra un mensaje indicando que el usuario o la contraseña no son correctos. 4.2. Continuar en el paso 2 del flujo normal.

Fuente: Elaboración Propia.

Tabla 3.5.

Descripción de caso de uso: Administrar Rutas.

Caso de Uso	**Administrar Rutas**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario para administrar las rutas.
Actores:	Usuario.
Precondición:	Validación de usuario Administrador en el formulario Login.
Flujo Principal:	1. Inicia cuando el usuario selecciona la opción "Rutas" en el menú desplegable "Formularios" del formulario Principal. 2. Se obtendrá y mostrará la lista de Rutas existentes.

	Punto de extensión: Exportar Ruta. **Punto de extensión: Importar Geopuntos en Ruta.**
Subflujos:	S-1 Nuevo registro de Ruta. Para crear una Ruta se presionará el botón "Nuevo" y se habilitarán los contenedores de los datos para una nueva ruta en la parte superior del formulario "Rutas", para almacenar estos datos se presionará el botón "Guardar". S-2 Editar registro de Ruta. Para actualizar los datos de una Ruta, previamente, se debe seleccionar una fila o registro en la tabla de Rutas y luego seleccionar el botón "Editar"; los contenedores de datos para una Ruta se habilitarán con los datos del registro seleccionado, para registrar los cambios realizados se presionará el botón "Guardar". S-3 Eliminar registro de Ruta. Para eliminar los datos de una Ruta, previamente, se debe seleccionar una fila o registro en la tabla de Rutas y posteriormente seleccionar el botón "Eliminar" y confirmación para realizar la eliminación.
Postcondición:	Crear, Obtener, Actualizar y Eliminar registros de Rutas.
Excepciones:	En S-1 y S-2 el campo Descripción posee una restricción que garantiza que no se registren valores duplicados en la base de datos.

Fuente: Elaboración Propia.

Tabla 3.6.

Descripción de caso de uso: Exportar Ruta.

Caso de Uso	**Exportar Ruta**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario "Rutas", para exportar los geopuntos pertenecientes a una ruta.
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir por lo menos un geopunto asignado a la ruta a exportar.
Flujo Principal:	1. Luego de elegir la ruta registrada, el usuario debe presionar el botón "Exportar Ruta" y luego confirmar la operación. 2. Se visualiza el explorador de archivos y el usuario elige una carpeta de destino e ingresa un nombre para guardar el archivo generado. 3. Se ejecuta el método para escribir un archivo con los geopuntos pertenecientes a una ruta.

Subflujos:	Ninguno.
Postcondición:	Exporta Ruta, obtiene un archivo con los geopuntos pertenecientes a una determinada Ruta.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

Tabla 3.7.

Descripción de caso de uso: Importar Geopuntos en Ruta.

Caso de Uso	**Importar Geopuntos en Ruta**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario "Rutas", para importar geopuntos y/o asociarlos a una ruta.
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir un archivo con geopuntos a importar.
Flujo Principal:	1. Luego de elegir la ruta registrada, el usuario debe presionar el botón "Importar Geopuntos en Ruta" y luego confirmar la operación. 2. El usuario busca y elige el archivo de geopuntos a importar. 3. Se ejecuta el método para leer un archivo con los geopuntos, el cual realiza la carga masiva de Geopuntos y/o asigna a los mismos la Ruta previamente seleccionada; para realizar esta acción con éxito, es necesario reconocer y validar el archivo que contiene los registros de Geopuntos a importar. **4. Include (Asignar Geopunto en Ruta).**
Subflujos:	Ninguno.
Postcondición:	Importar Geopuntos en Ruta, registra Geopuntos y/o asigna a los mismos la Ruta previamente seleccionada.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

Tabla 3.8.

Descripción de caso de uso: Administrar Geopuntos.

Caso de Uso	**Administrar Geopuntos**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario para administrar los geopuntos.
Actores:	Usuario.
Precondición:	Validación de usuario Administrador en el formulario Login.
Flujo Principal:	1. Inicia cuando el usuario selecciona la opción "Geopuntos" en el menú desplegable "Configuración" del formulario Principal. 2. Se obtendrá y mostrará la lista de Geopuntos existentes. **Punto de extensión: Asignar Geopunto en Ruta.** **Punto de extensión: Quitar Geopunto en Ruta.**
Subflujos:	S-1 Nuevo registro de Geopunto. Para crear un Geopunto se presionará el botón "Nuevo" y se habilitarán los contenedores de los datos para un nuevo Geopunto en la parte superior del formulario "Geopuntos", para almacenar estos datos se presionará el botón "Guardar". S-2 Editar registro de Geopunto. Para actualizar los datos de un Geopunto, previamente, se debe seleccionar una fila o registro en la tabla de Geopuntos y luego seleccionar el botón "Editar"; los contenedores de datos para un Geopunto se habilitarán con los datos del registro seleccionado, para registrar los cambios realizados se presionará el botón "Guardar". S-3 Eliminar registro de Geopunto. Para eliminar los datos de un Geopunto, previamente, se debe seleccionar una fila o registro en la tabla de Geopuntos y posteriormente seleccionar el botón "Eliminar" y confirmación para realizar la eliminación. S-4 Habilitar Administrar Ruta(s) del Geopunto. Habilita los contenedores para Asignar o Quitar una Ruta(s) a un Geopunto, mediante el botón "Administrar Ruta(s) del Geopunto".
Postcondición:	Crear, Obtener, Actualizar, Eliminar registros de Geopuntos y/o Habilitar Administrar Ruta(s) del Geopunto.
Excepciones:	En S-1 y S-2 la combinación de los campos Latitud y Longitud poseen una restricción que garantiza que no se registren valores duplicados en la base de datos.

Fuente: Elaboración Propia.

Tabla 3.9.

Descripción de caso de uso: Asignar Geopunto en Ruta.

Caso de Uso	**Asignar Geopunto en Ruta**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario "Geopuntos", para asignarle a un geopunto una ruta(s).
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir una ruta(s) para asignar al geopunto previamente seleccionado.
Flujo Principal:	1. SI el usuario no solicita la operación de Importar Geopuntos en Ruta desde el formulario "Rutas" y selecciona Habilitar Administrar Ruta(s) del Geopunto en el formulario "Geopuntos". 1.1. Luego de elegir el geopunto registrado, mediante el botón "Administrar Ruta(s) del Geopunto", se habilitan los contenedores para Asignar o Quitar una Ruta(s) a un Geopunto. 1.2. El usuario selecciona una ruta existente y presiona el botón "Asignar Ruta". 1.3. Se ejecuta sp_insertar_geopunto_en_ruta. 1.4. Se actualiza el listado de las rutas asignadas al Geopunto correspondiente. 2. SI NO 2.1. Se identifica el id del Geopunto y el id de la Ruta a insertar. 2.2. Se ejecuta sp_insertar_geopunto_en_ruta.
Subflujos:	S-1 Administrar Geopuntos. El botón "Administrar Geopuntos", permite regresar al flujo inicial del formulario "Geopuntos".
Postcondición:	Se le asigna a un Geopunto una Ruta.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

Tabla 3.10.

Descripción de caso de uso: Quitar Geopunto en Ruta.

Caso de Uso	**Quitar Geopunto en Ruta**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario "Geopuntos", para quitarle a un geopunto una ruta(s).
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir por lo menos una ruta asignada al geopunto previamente seleccionado.
Flujo Principal:	1. Luego de elegir el geopunto registrado, mediante el botón "Administrar Ruta(s) del Geopunto", se habilitan los contenedores para Asignar o Quitar una Ruta(s) a un Geopunto. 2. El usuario selecciona una ruta asignada y presiona el botón "Quitar Ruta". 3. Se ejecuta sp_eliminar_geopunto_en_ruta. 4. Se actualiza el listado de las rutas asignadas al Geopunto correspondiente.
Subflujos:	S-1 Administrar Geopuntos. El botón "Administrar Geopuntos", permite regresar al flujo inicial del formulario "Geopuntos".
Postcondición:	Se le quita a un Geopunto una Ruta.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

Tabla 3.11.

Descripción de caso de uso: Administrar Vehículos.

Caso de Uso	**Administrar Vehículos**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario para administrar los vehículos.
Actores:	Usuario.
Precondición:	Validación de usuario Administrador en el formulario Login.
Flujo Principal:	1. Inicia cuando el usuario selecciona la opción "Vehículos" en el menú desplegable "Formularios" del formulario Principal. 2. Se obtendrá y mostrará la lista de Vehículos existentes.
Subflujos:	S-1 Nuevo registro de Vehículo. Para crear un Vehículo se presionará el botón "Nuevo" y se

	habilitarán los contenedores de los datos para un nuevo vehículo en la parte superior del formulario "Vehículos", para almacenar estos datos se presionará el botón "Guardar". S-2 Editar registro de Vehículo. Para actualizar los datos de un Vehículo, previamente, se debe seleccionar una fila o registro en la tabla de Vehículos y luego seleccionar el botón "Editar"; los contenedores de datos para un Vehículo se habilitarán con los datos del registro seleccionado, para registrar los cambios realizados se presionará el botón "Guardar". S-3 Eliminar registro de Vehículo. Para eliminar los datos de un Vehículo, previamente, se debe seleccionar una fila o registro en la tabla de Vehículos y posteriormente seleccionar el botón "Eliminar" y confirmación para realizar la eliminación.
Postcondición:	Crear, Obtener, Actualizar y/o Eliminar registros de Vehículos.
Excepciones:	En S-1 y S-2 el campo Placa posee una restricción que garantiza que no se registren valores duplicados en la base de datos.

Fuente: Elaboración Propia.

Tabla 3.12.

Descripción de caso de uso: Importar Transmisiones.

Caso de Uso	**Importar Transmisiones**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario para importar transmisiones.
Actores:	Usuario.
Precondición:	Validación de usuario Administrador en el formulario Login.
Flujo Principal:	1. Inicia cuando el usuario selecciona la opción "Transmisiones" en el menú desplegable "Monitoreo" del formulario Principal. 2. Se obtendrá y mostrará la lista de Transmisiones existentes disponibles. 3. El botón "Importar Transmisiones", permite realizar la carga masiva de Transmisiones; para realizar esta acción con éxito, además de contar con la confirmación del usuario, es necesario reconocer y validar el archivo que contiene los registros de Transmisiones a importar a través del proceso ETL. 4. Se ejecuta sp_insertar_transmision.
Subflujos:	Ninguno.

Postcondición:	Se registran Transmisiones.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

Tabla 3.13.

Descripción de caso de uso: Administrar Cargos.

Caso de Uso	**Administrar Cargos**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario para administrar los cargos.
Actores:	Usuario.
Precondición:	Validación de usuario Administrador en el formulario Login.
Flujo Principal:	1. Inicia cuando el usuario selecciona la opción "Cargos" en el menú desplegable "Formularios" del formulario principal. 2. Se obtendrá y mostrará la lista de Cargos existentes.
Subflujos:	S-1 Nuevo registro de Cargo. Para crear un Cargo se presionará el botón "Nuevo" y se habilitarán los contenedores de los datos para un nuevo cargo en la parte superior del formulario "Cargos", para almacenar estos datos se presionará el botón "Guardar". S-2 Editar registro de Cargo. Para actualizar los datos de un Cargo, previamente, se debe seleccionar una fila o registro en la tabla de Cargos y luego seleccionar el botón "Editar"; los contenedores de datos para un Cargo se habilitarán con los datos del registro seleccionado, para registrar los cambios realizados se presionará el botón "Guardar". S-3 Eliminar registro de Cargo. Para eliminar los datos de un Cargo, previamente, se debe seleccionar una fila o registro en la tabla de Cargos y posteriormente seleccionar el botón "Eliminar" y confirmación para realizar la eliminación.
Postcondición:	Crear, Obtener, Actualizar y/o Eliminar registros de Cargos para un Trabajador.
Excepciones:	En S-1 y S-2 el campo Descripción posee una restricción que garantiza que no se registren valores duplicados en la base de datos.

Fuente: Elaboración Propia.

Tabla 3.14.

Descripción de caso de uso: Administrar Trabajadores.

Caso de Uso	**Administrar Trabajadores**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario para administrar los trabajadores.
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir al menos un Cargo.
Flujo Principal:	1. Inicia cuando el usuario selecciona la opción "Trabajadores" en el menú desplegable "Formularios" del formulario Principal. 2. Se obtendrá y mostrará la lista de Trabajadores existentes.
Subflujos:	S-1 Nuevo registro de Trabajador. Para crear un Trabajador se presionará el botón "Nuevo" y se habilitarán los contenedores de los datos para un nuevo trabajador en la parte superior del formulario "Trabajadores", para almacenar estos datos se presionará el botón "Guardar". S-2 Editar registro de Trabajador. Para actualizar los datos de un Trabajador, previamente, se debe seleccionar una fila o registro en la tabla de Trabajadores y luego seleccionar el botón "Editar"; los contenedores de datos para un Trabajador se habilitarán con los datos del registro seleccionado, para registrar los cambios realizados se presionará el botón "Guardar". S-3 Eliminar registro de Trabajador. Para eliminar los datos de un Trabajador, previamente, se debe seleccionar una fila o registro en la tabla de Trabajadores y posteriormente seleccionar el botón "Eliminar" y confirmación para realizar la eliminación.
Postcondición:	Crear, Obtener, Actualizar y/o Eliminar registros de Trabajadores.
Excepciones:	En S-1 y S-2 el campo DNI y el campo Abreviación poseen una restricción que garantiza que no se e registren valores duplicados en la base de datos.

Fuente: Elaboración Propia.

Tabla 3.15.

Descripción de caso de uso: Administrar Viajes.

Caso de Uso	**Administrar Viajes**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario para administrar los viajes.
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir al menos un Trabajador con el cargo de Conductor. 3. Deben existir transmisiones importadas por el proceso ETL con el campo de "Evaluación" = 0.
Flujo Principal:	1. Inicia cuando el usuario selecciona la opción "Viajes" en el menú desplegable "Monitoreo" del formulario Principal. 2. Se obtendrá y mostrará la lista de Viajes existentes. **Punto de extensión: Asignar Conductor en Viaje.** **Punto de extensión: Quitar Conductor en Viaje.** **Punto de extensión: Evaluar Desplazamiento.** **Punto de extensión: Exportar Viaje Evaluado.**
Subflujos:	S-1 Nuevo registro de Viaje. Para crear un Viaje se presionará el botón "Nuevo" y se habilitarán los contenedores de los datos para un nuevo Viaje en la parte superior del formulario "Viajes" en la sección "Datos del Viaje", para almacenar estos datos se presionará el botón "Guardar". S-2 Editar registro de Viaje. Para actualizar los datos de una Viaje, previamente, se debe seleccionar una fila o registro en la tabla de Viajes y luego seleccionar el botón "Editar"; los contenedores de datos para un Viaje se habilitarán con los datos del registro seleccionado, para registrar los cambios realizados se presionará el botón "Guardar". S-3 Eliminar registro de Viaje. Para eliminar los datos de un Viaje, previamente, se debe seleccionar una fila o registro en la tabla de Viajes y posteriormente seleccionar el botón "Eliminar" y confirmación para realizar la eliminación. S-4 Habilitar Administrar Conductor(es) del Viaje. Habilita los contenedores para Asignar o Quitar un Conductor(es) a un Viaje.
Postcondición:	Crear, Obtener, Actualizar, Eliminar registros de Viajes y/o Habilitar Administrar Conductor(es) del Viaje.
Excepciones:	En S-1 y S-2 la combinación de los campos Fecha de salida y el id vehículo poseen una restricción que garantiza que no se registren valores duplicados en la base de datos.

Fuente: Elaboración Propia.

Tabla 3.16.

Descripción de caso de uso: Asignar Conductor en Viaje.

Caso de Uso	**Asignar Conductor en Viaje**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario "Viajes", para asignarle a un viaje un conductor(es).
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir por lo menos un conductor registrado en el formulario Trabajadores.
Flujo Principal:	1. Luego de elegir el viaje registrado, mediante el botón "Administrar Conductor(es) del Viaje", se habilitan los contenedores para Asignar o Quitar un Conductor(es) a un Viaje. 2. El usuario selecciona un conductor existente y presiona el botón "Asignar Conductor". 3. Se ejecuta sp_insertar_trabajador_en_viaje. 4. Se actualiza el listado de los conductores asignados al Viaje correspondiente.
Subflujos:	S-1 Administrar Viajes. El botón "Administrar Viajes", permite regresar al flujo inicial del formulario "Viajes".
Postcondición:	Se le asigna a un Viaje un conductor.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

Tabla 3.17.

Descripción de caso de uso: Quitar Conductor en Viaje.

Caso de Uso	**Quitar Conductor en Viaje**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario "Viajes", para quitarle a un viaje un conductor(es).
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir por lo menos una ruta asignada al geopunto previamente seleccionado.
Flujo Principal:	1. Luego de elegir el viaje registrado, mediante el botón "Administrar Conductor(es) del Viaje", se habilitan los contenedores para Asignar o Quitar un Conductor(es) a un

	Viaje. 2. El usuario selecciona un conductor asignado y presiona el botón “Quitar Conductor”. 3. Se ejecuta sp_eliminar_trabajador_en_viaje. 4. Se actualiza el listado de los conductores asignados al Viaje correspondiente.
Subflujos:	S-1 Administrar Viajes. El botón “Administrar Viajes”, permite regresar al flujo inicial del formulario “Viajes”.
Postcondición:	Se le quita a un Viaje un conductor.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

Tabla 3.18.

Descripción de caso de uso: Evaluar Desplazamiento.

Caso de Uso	**Evaluar Desplazamiento**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario “Viajes”, para evaluar en desplazamiento registrado en cada una de las transmisiones que correspondan a un viaje, se emplea en conjunto el algoritmo NN y las reglas de decisión.
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir geopuntos registrados y asignados a la ruta de dicho Viaje. 3. Debe existir transmisiones recientemente importadas o tener ya registradas transmisiones con el campo de evaluación igual a “0”, y en ambos casos, que estén en el intervalo de fechas del Viaje a evaluar.
Flujo Principal:	1. Luego de elegir el viaje previamente evaluado, el usuario debe presionar el botón “Evaluar Desplazamiento” y luego confirmar la operación. 2. Se ejecuta el método para cargar los geopuntos pertenecientes a la ruta del Viaje. 3. Se ejecuta el método para cargar las transmisiones que corresponden al Viaje a evaluar, según el id del vehículo y rango de fechas del mismo. 4. Se hace un recorrido de todas las transmisiones cargadas, permitiendo hallar el geopunto cercano para cada una de ellas. 5. Se aplica las reglas de decisión y se ejecuta

	sp_actualizar_informacion_en_transmision. 6. Se ejecuta sp_actualizar_informacion_en_viaje para actualizar el número de transmisiones y el número de incidentes correspondientes al Viaje al desplazamiento evaluado.
Subflujos:	Ninguno.
Postcondición:	Se actualiza la información de todas las transmisiones correspondientes al desplazamiento del Viaje evaluado; clasificando dichas transmisiones para hallar el número de incidentes registrados.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

Tabla 3.19.

Descripción de caso de uso: Exportar Viaje Evaluado.

Caso de Uso	**Exportar Viaje Evaluado**
Descripción:	El propósito de este caso de uso es describir el flujo que se realiza en el formulario "Viajes", para exportar las transmisiones evaluadas y clasificadas pertenecientes a un viaje.
Actores:	Usuario.
Precondición:	1. Validación de usuario Administrador en el formulario Login. 2. Debe existir por lo menos una transmisión asignada al viaje a exportar, para ello el viaje seleccionado debe estar previamente evaluado.
Flujo Principal:	1. Luego de elegir el viaje previamente evaluado, el usuario debe presionar el botón "Exportar Viaje Evaluado" y luego confirmar la operación. 2. Se ejecuta el método para escribir un archivo con las transmisiones pertenecientes a dicho viaje. 3. El usuario ingresa un nombre y elige una carpeta de destino para guardar el archivo generado.
Subflujos:	Ninguno.
Postcondición:	Se obtiene un archivo con las transmisiones evaluadas y clasificadas pertenecientes a un Viaje.
Excepciones:	Ninguno.

Fuente: Elaboración Propia.

C. Diagramas de secuencias

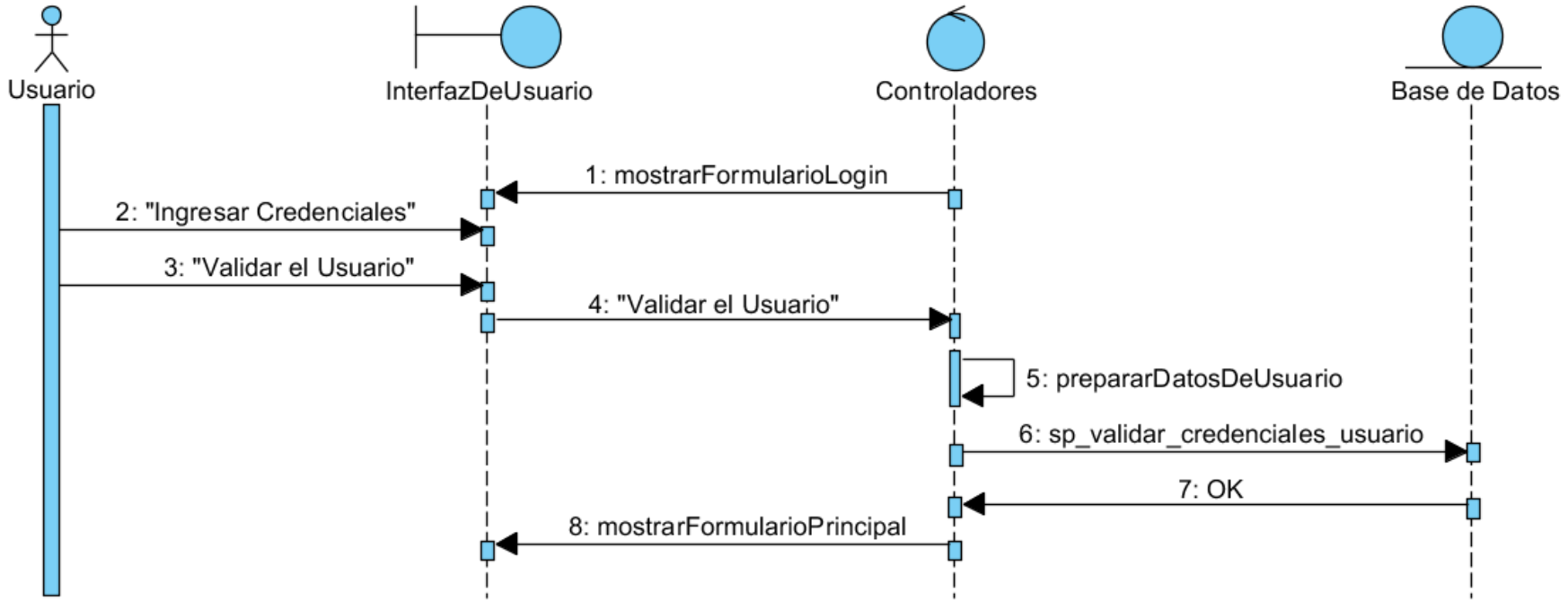


Figura 3.13. Diagrama de secuencia: Validar usuario.

Fuente: Elaboración propia.

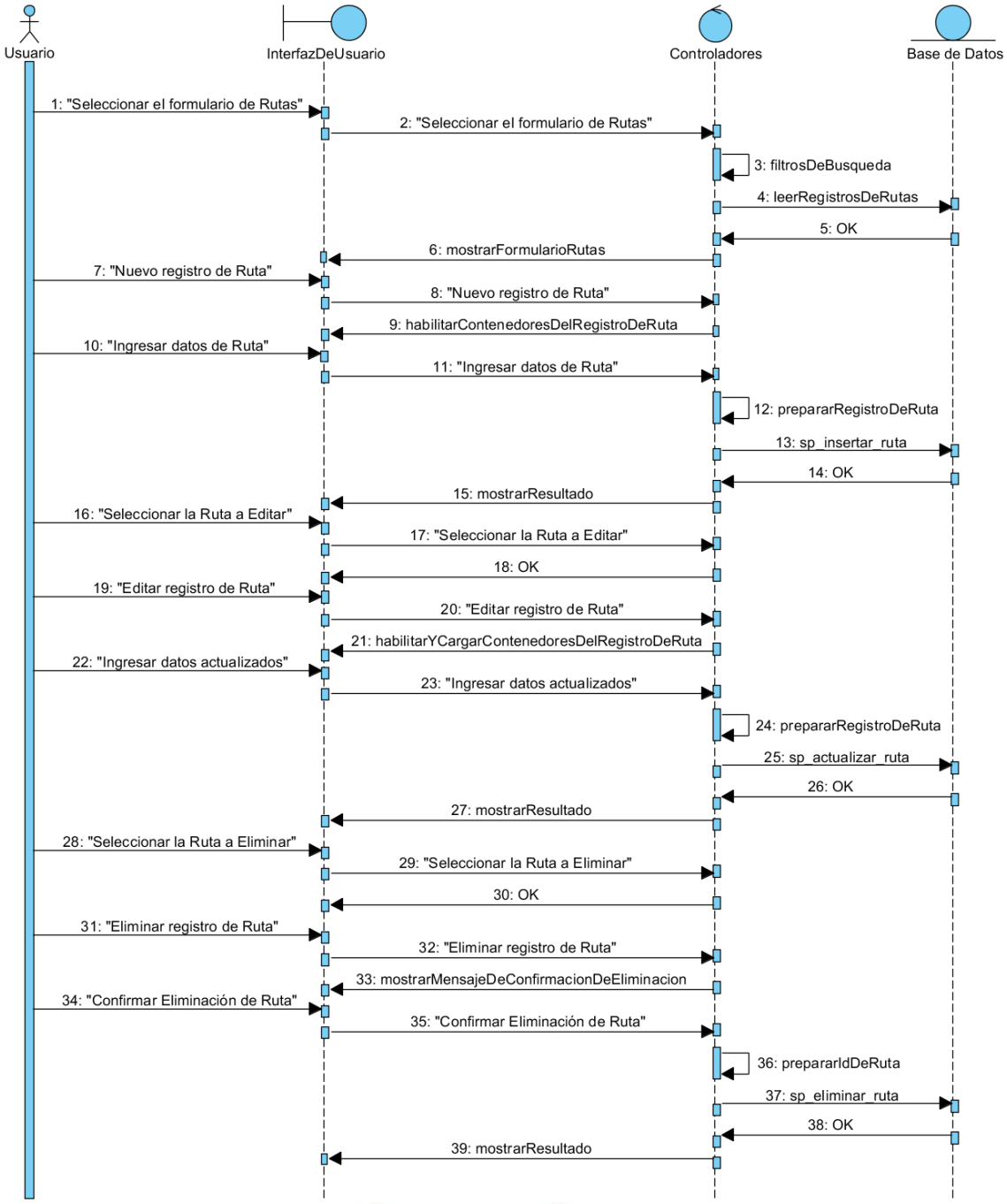


Figura 3.14. Diagrama de secuencia: Administrar Rutas.

Fuente: Elaboración propia.

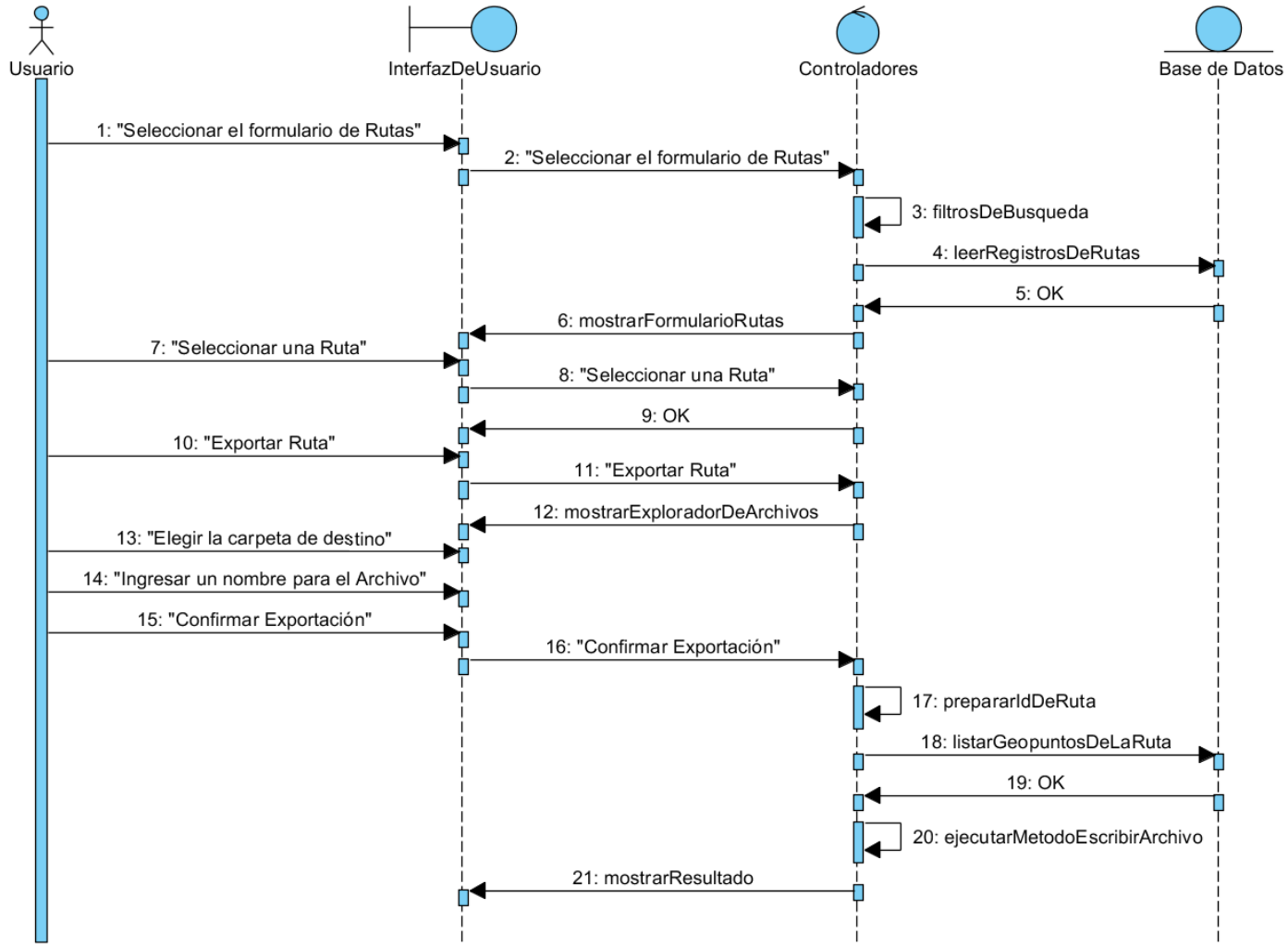


Figura 3.15. Diagrama de secuencia: Exportar Ruta.

Fuente: Elaboración propia.

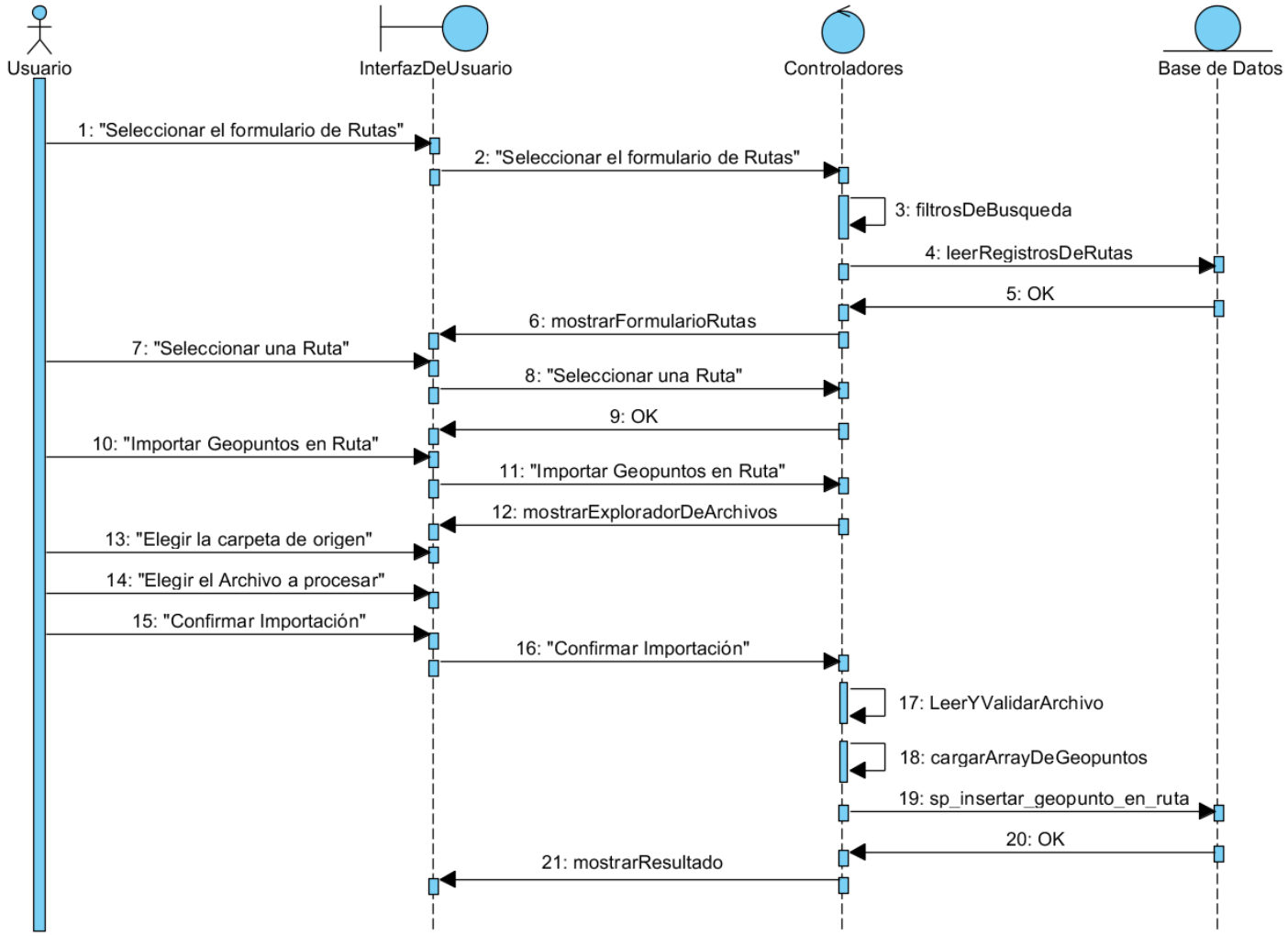


Figura 3.16. Diagrama de secuencia: Importar Geopuntos en Ruta.

Fuente: Elaboración propia.

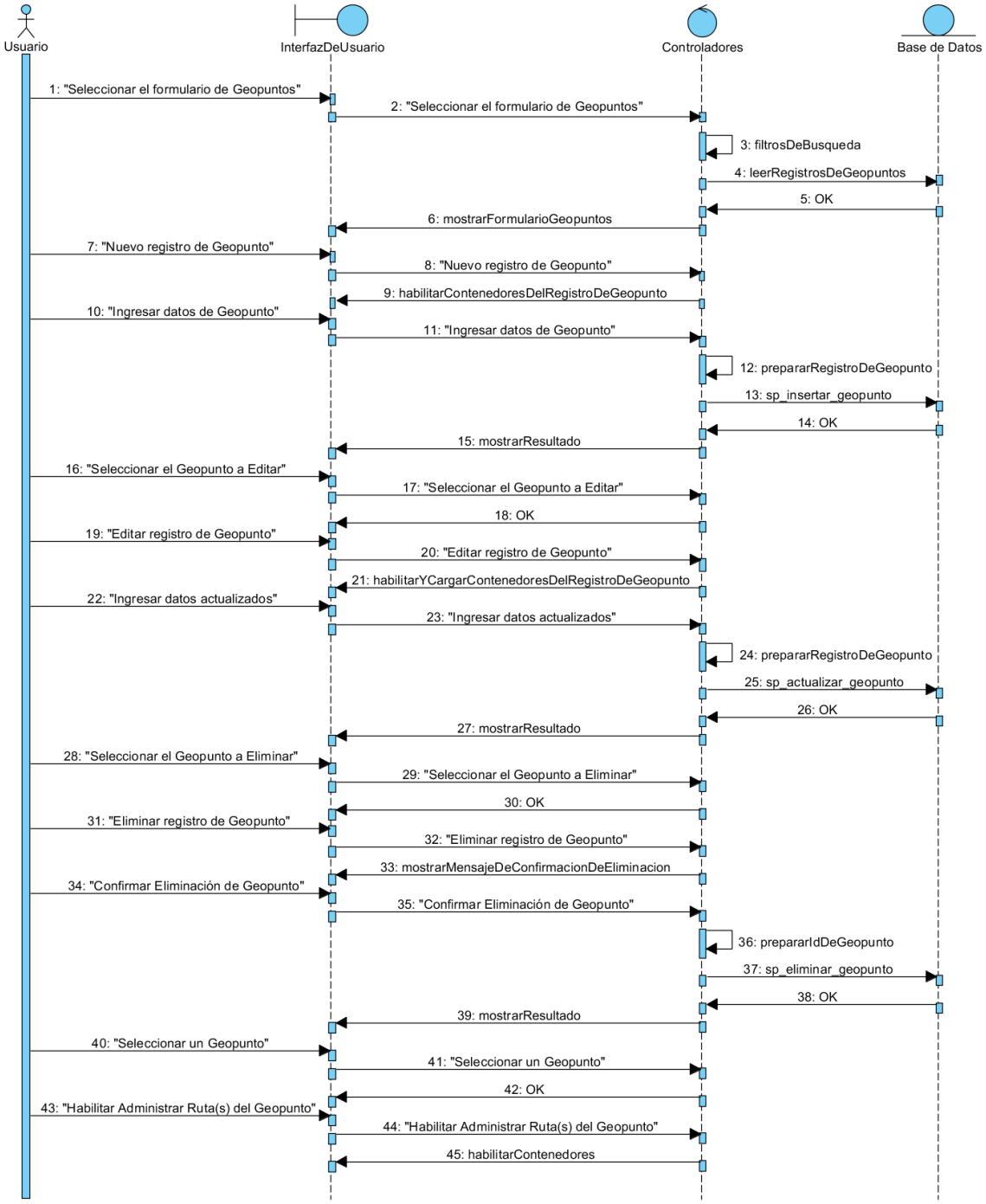


Figura 3.17. Diagrama de secuencia: Administrar Geopuntos.

Fuente: Elaboración propia.

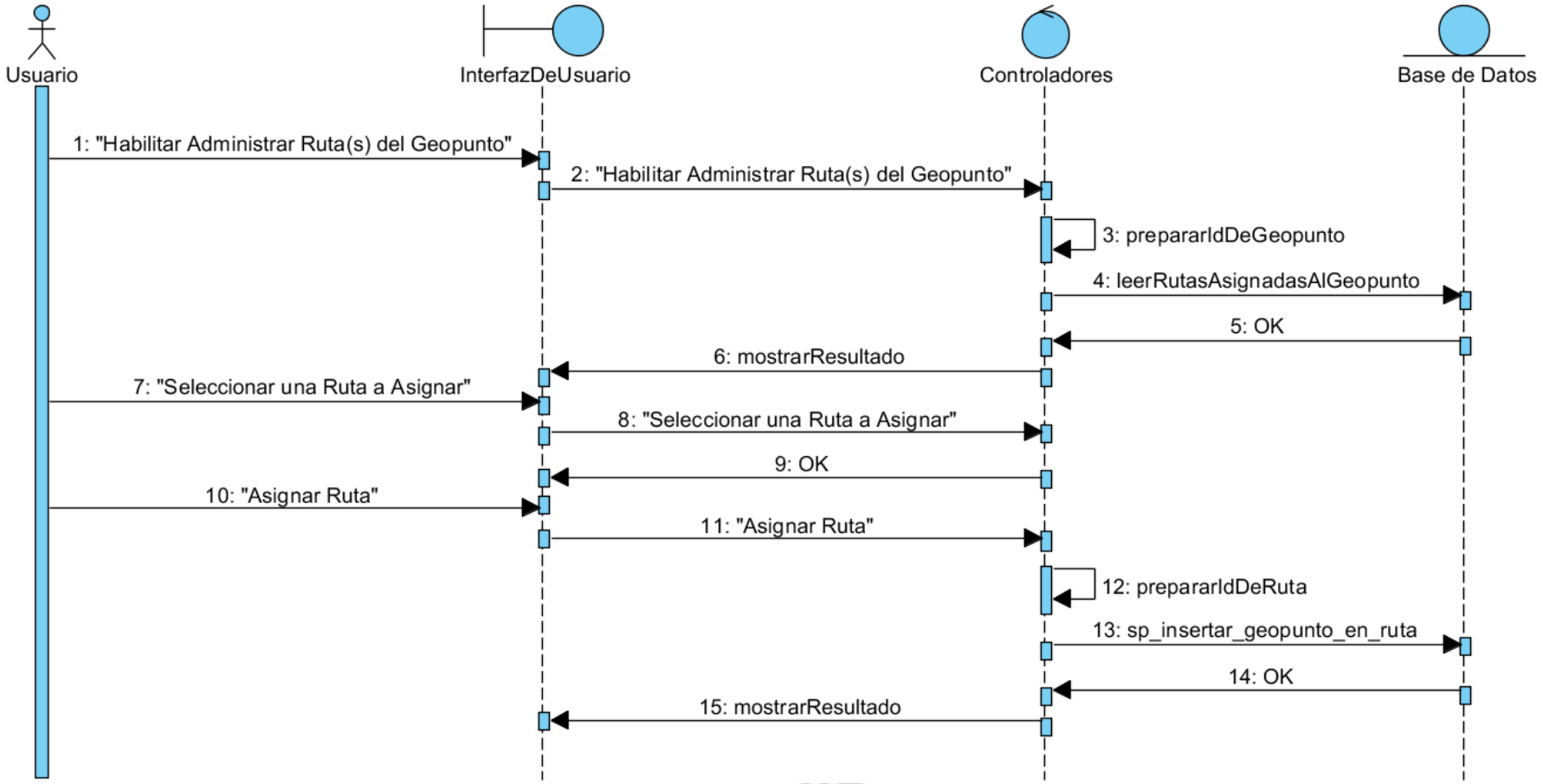


Figura 3.18. Diagrama de secuencia: Asignar Geopunto en Ruta.

Fuente: Elaboración propia.

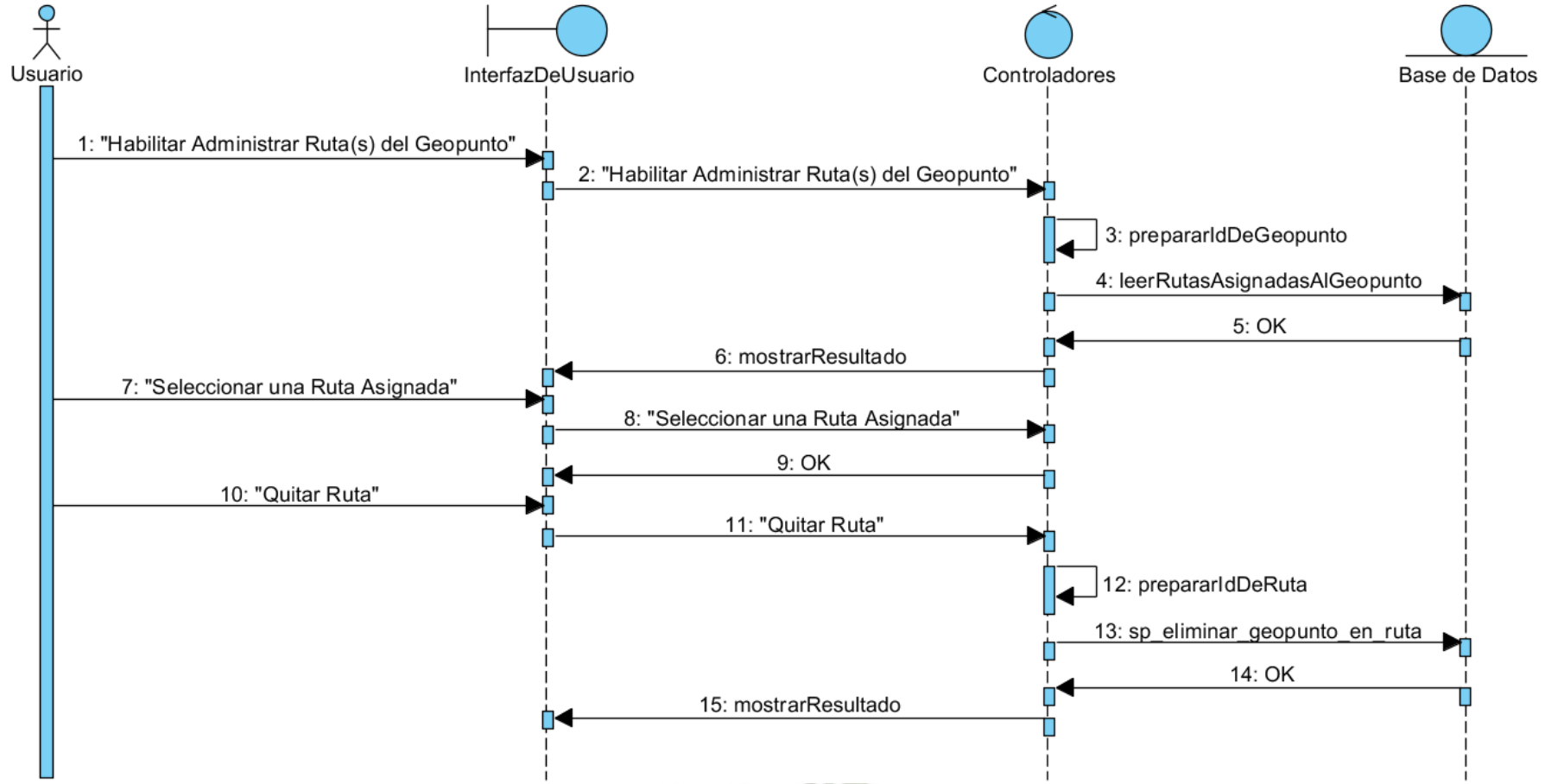


Figura 3.19. Diagrama de secuencia: Quitar Geopunto en Ruta.

Fuente: Elaboración propia.

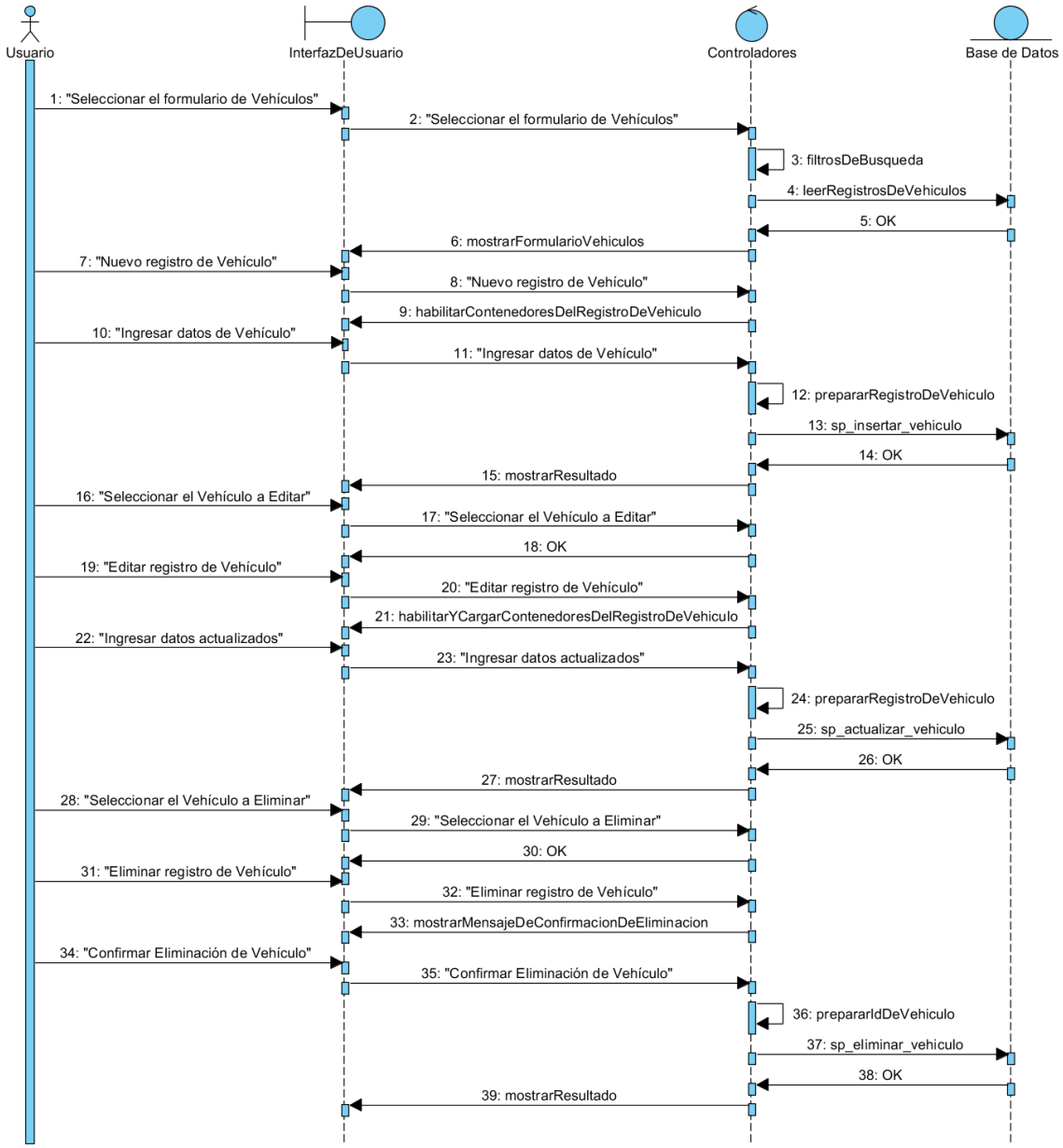


Figura 3.20. Diagrama de secuencia: Administrar Vehículos.

Fuente: Elaboración propia.

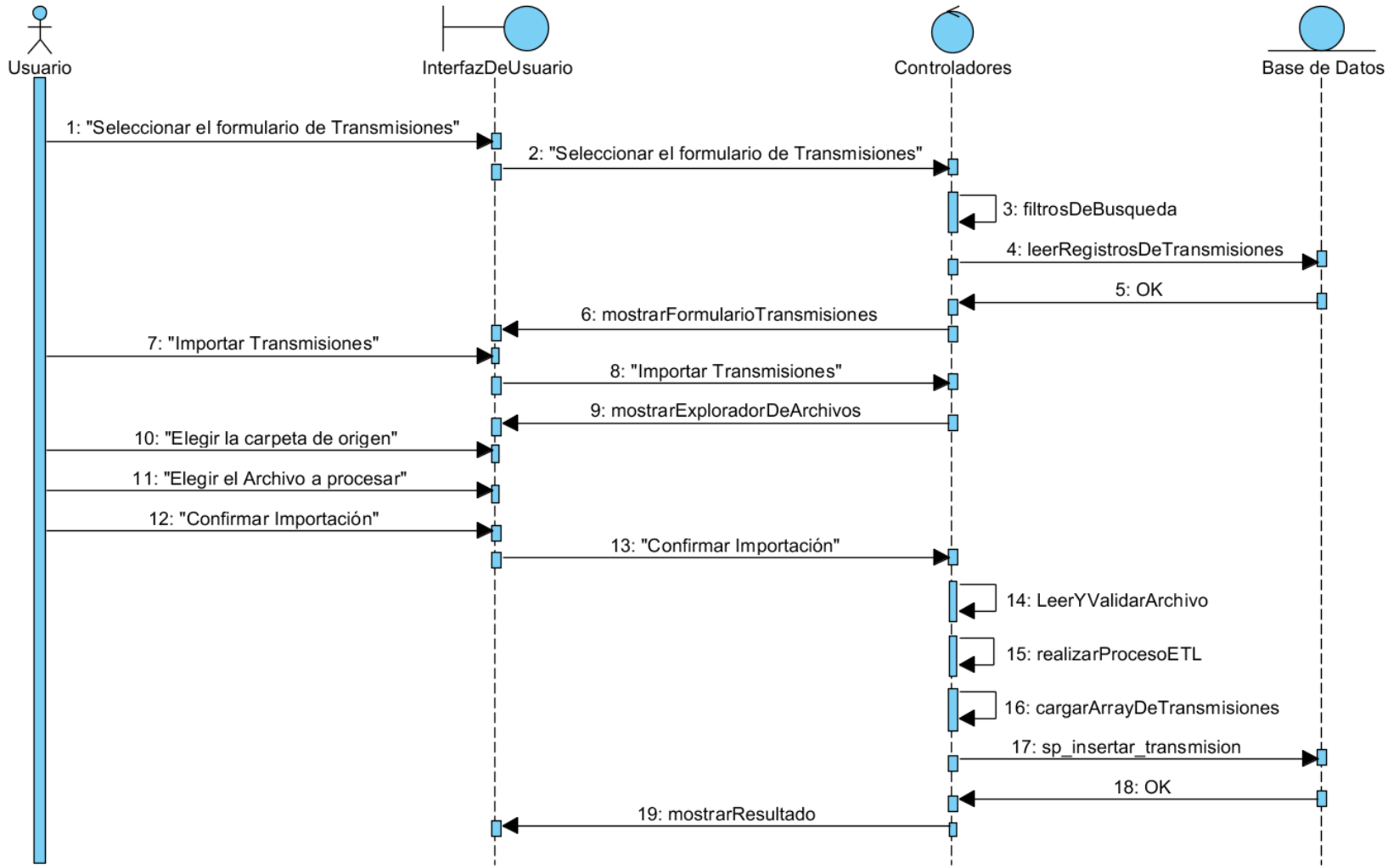


Figura 3.21. Diagrama de secuencia: Importar Transmisiones.

Fuente: Elaboración propia.

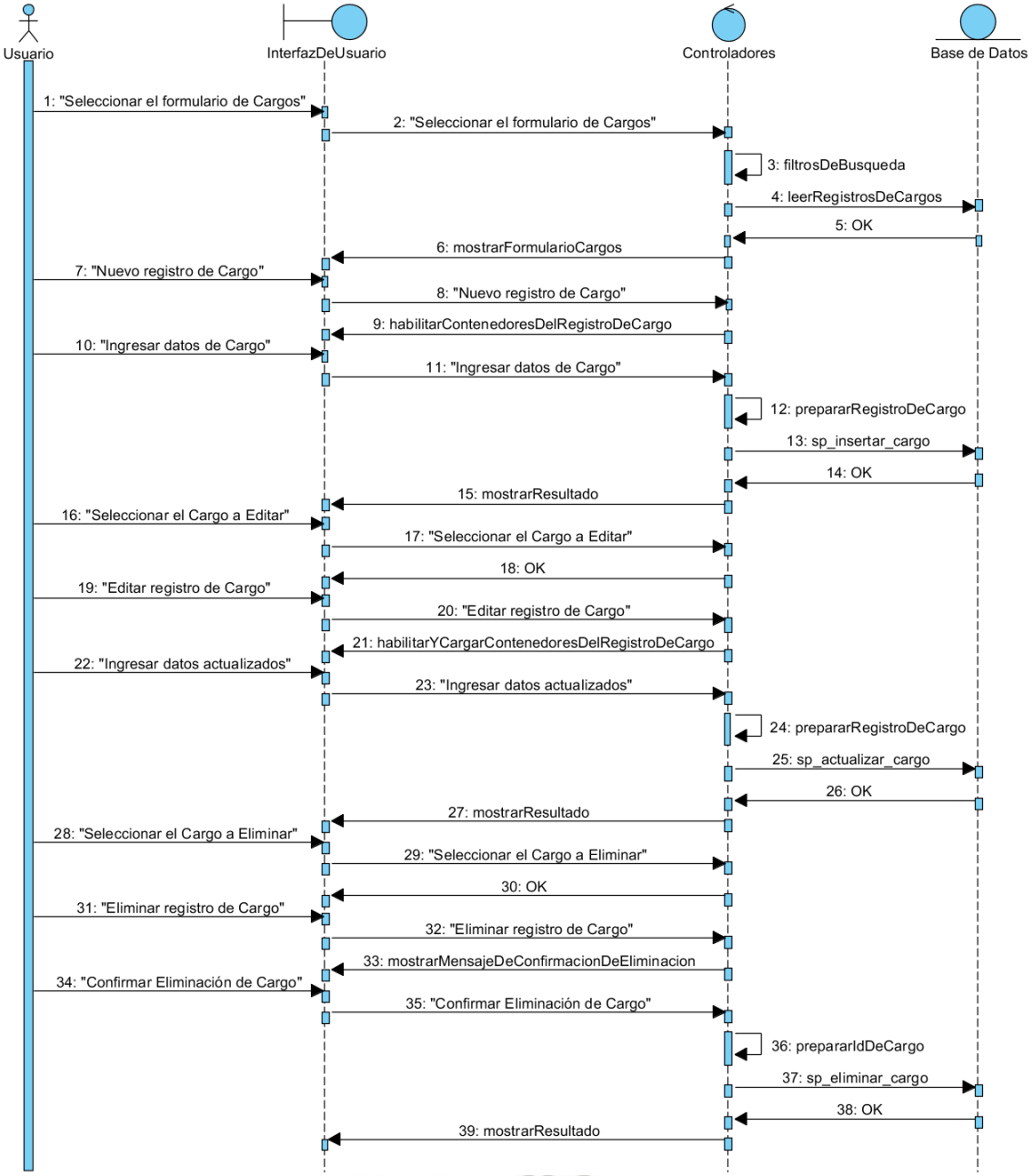


Figura 3.22. Diagrama de secuencia: Administrar Cargos.

Fuente: Elaboración propia.

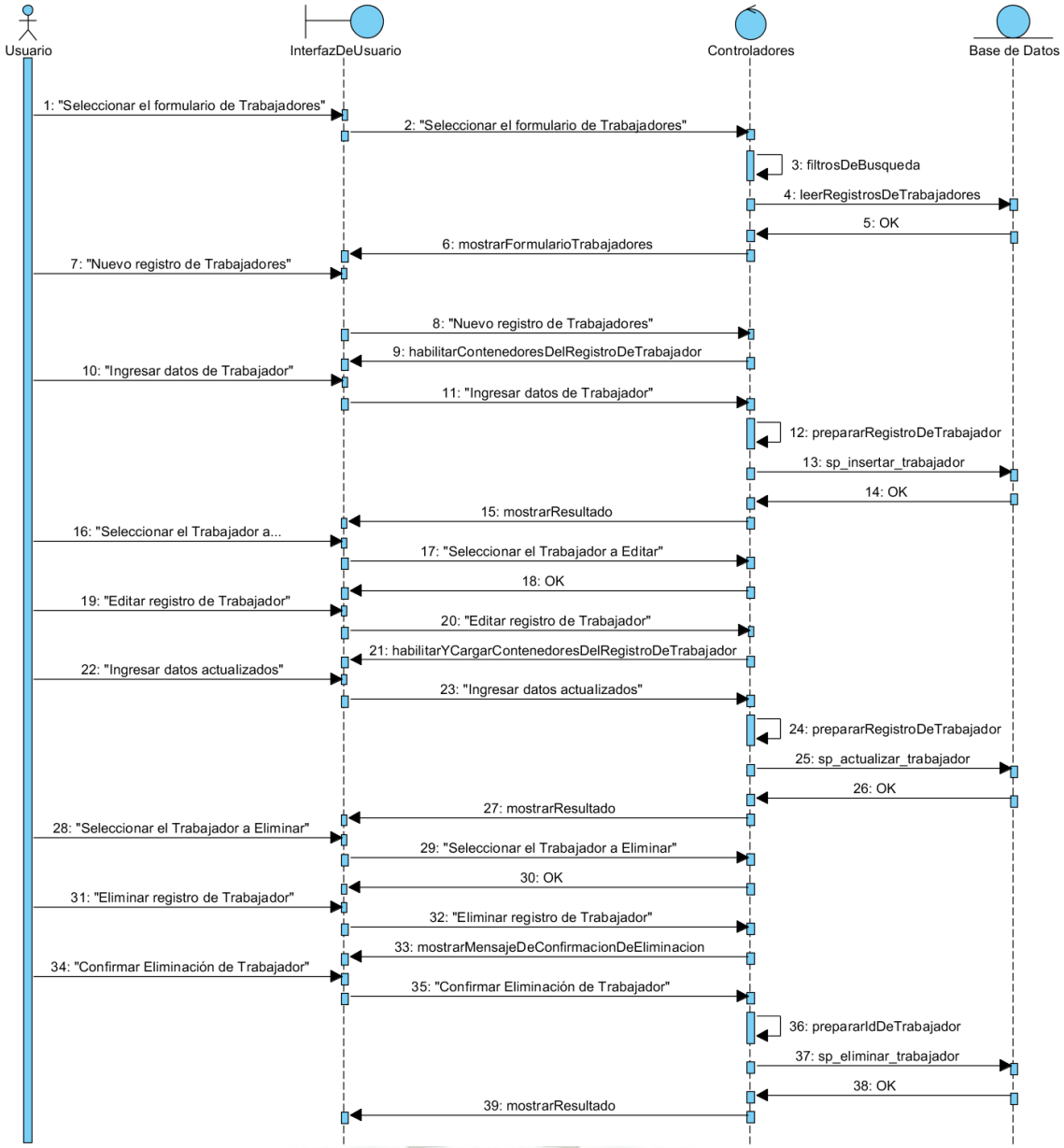


Figura 3.23. Diagrama de secuencia: Administrar Trabajadores.

Fuente: Elaboración propia.

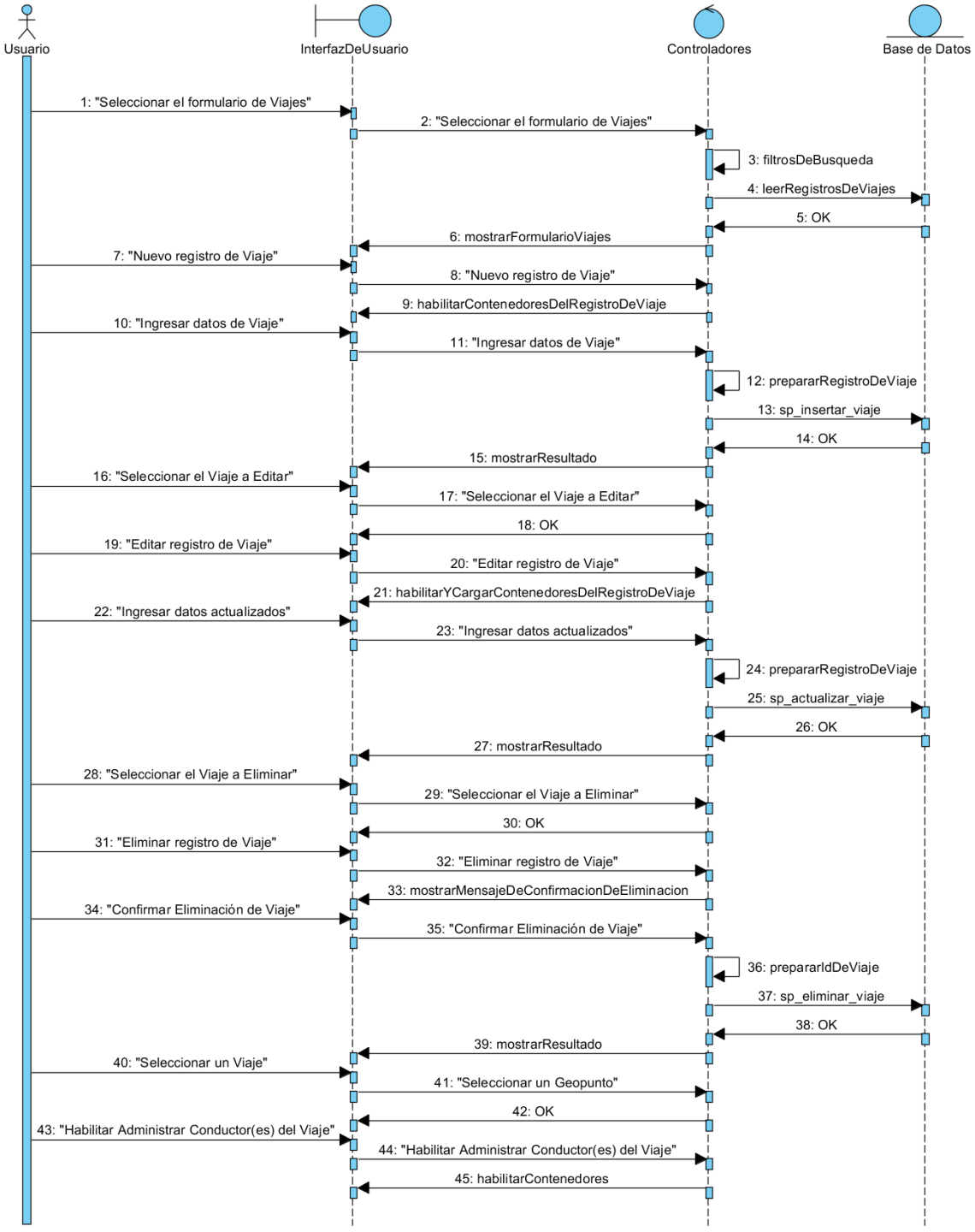


Figura 3.24. Diagrama de secuencia: Administrar Viajes.

Fuente: Elaboración propia.

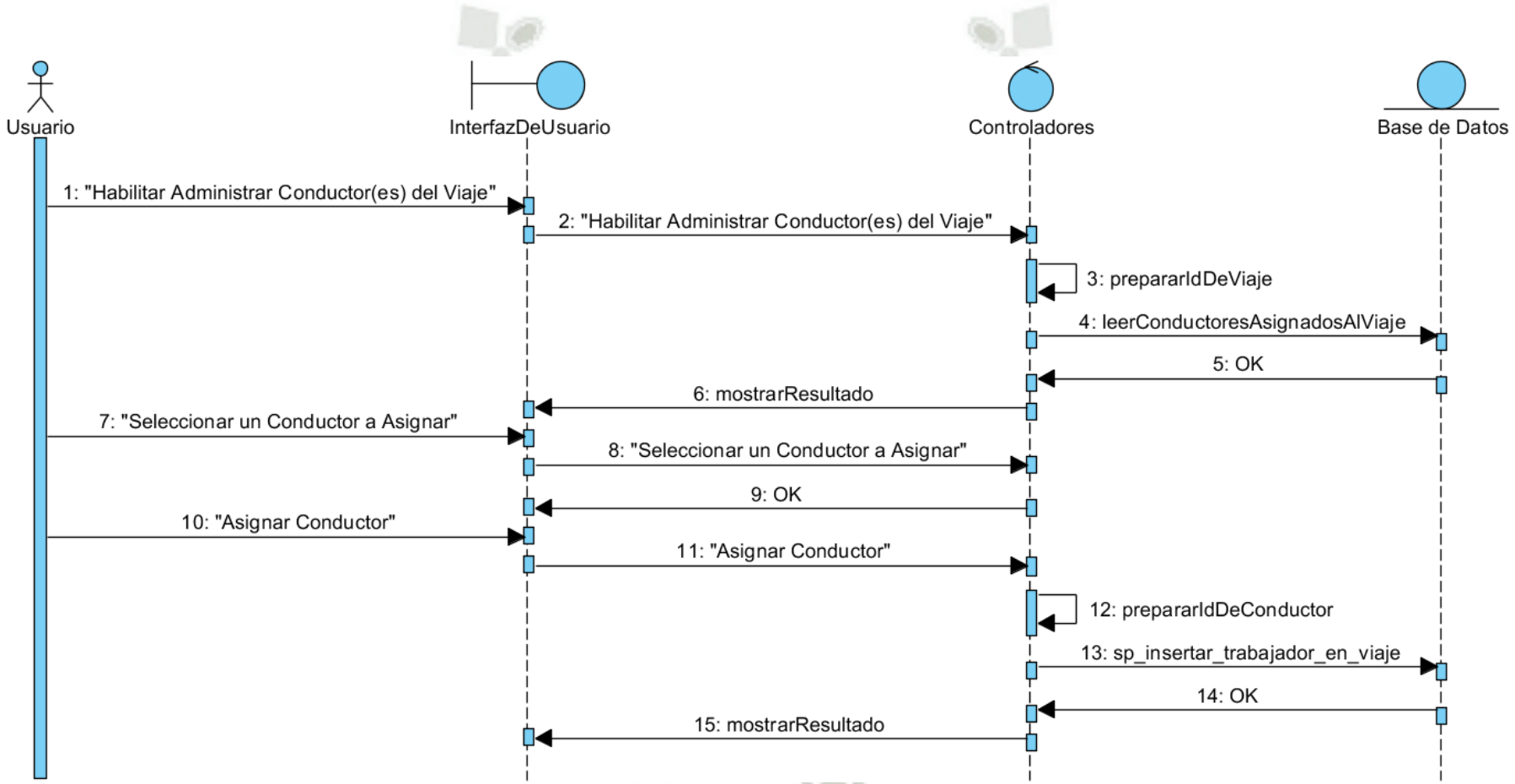


Figura 3.25. Diagrama de secuencia: Asignar Conductor en Viaje.

Fuente: Elaboración propia.

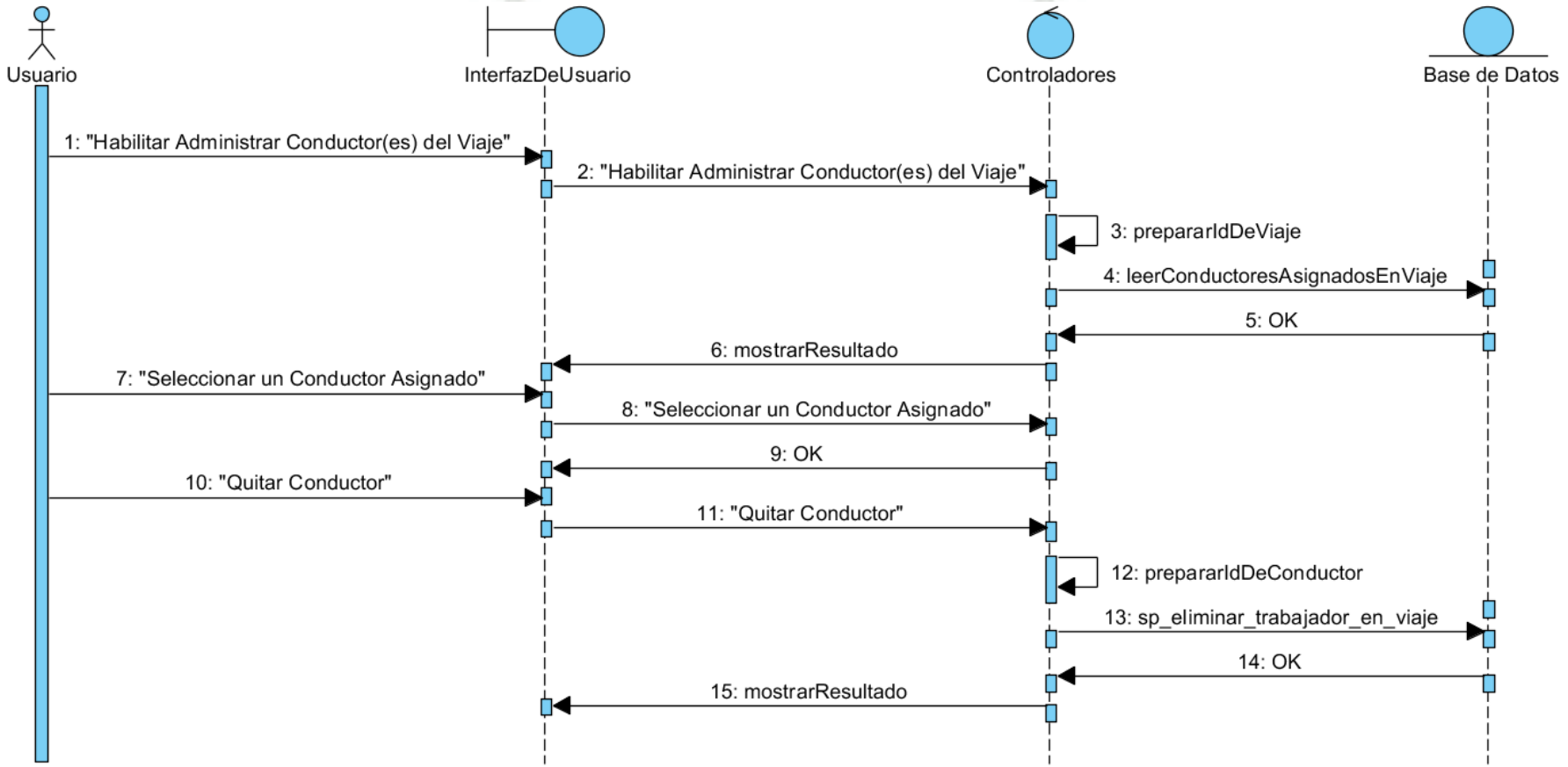


Figura 3.26. Diagrama de secuencia: Quitar Conductor en Viaje.

Fuente: Elaboración propia.

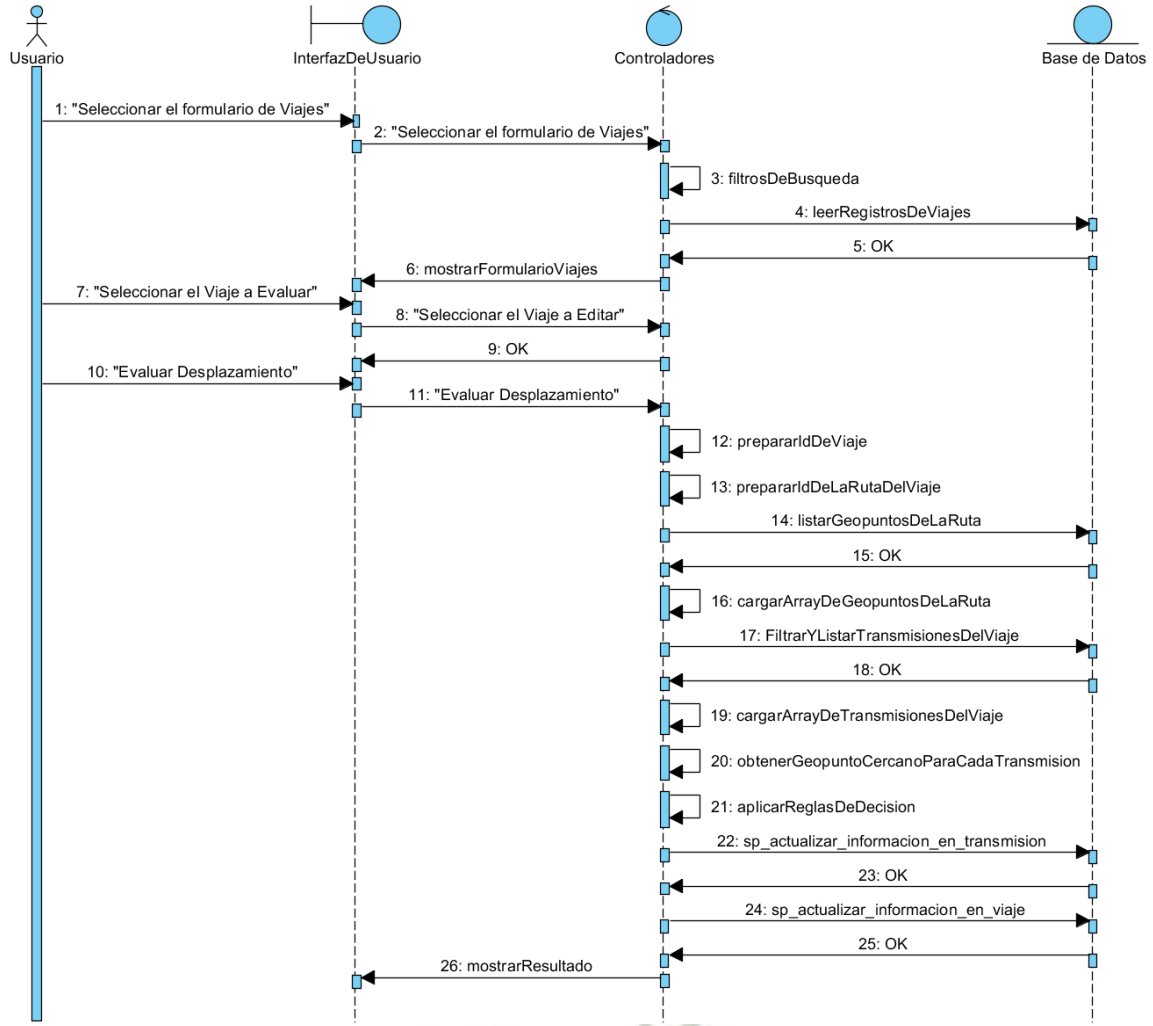


Figura 3.27. Diagrama de secuencia: Evaluar Desplazamiento.

Fuente: Elaboración propia.

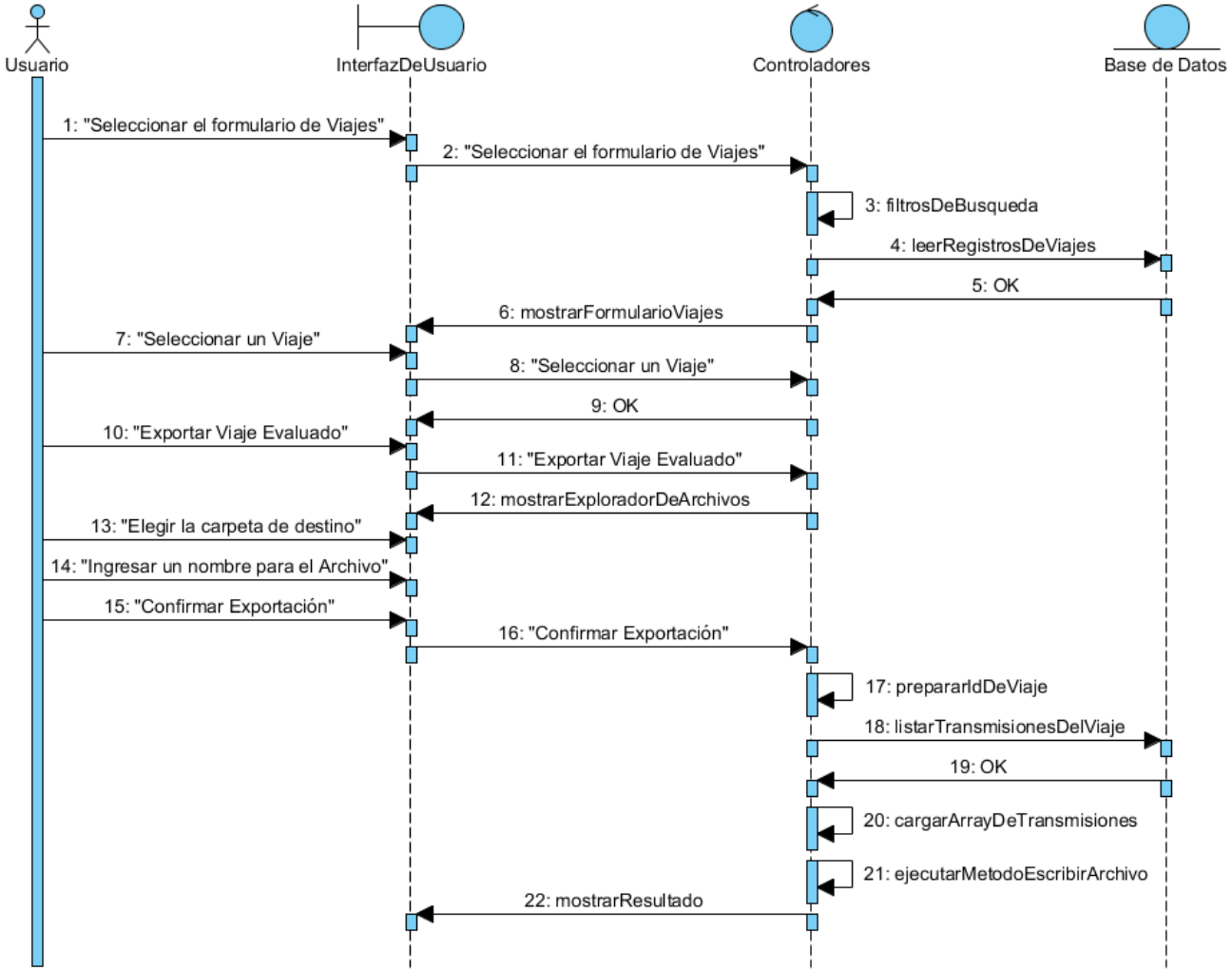


Figura 3.28. Diagrama de secuencia: Exportar Viaje Evaluado.

Fuente: Elaboración propia.

3.5. Fase 5: Evaluación.

En esta fase se realizan una serie de análisis en base a la validación de los resultados obtenidos a partir de un archivo de prueba; teniendo en cuenta que el nivel de confiabilidad e integridad de los conjuntos de entrenamiento debe ser muy alto, debido a que estos, cumplen un rol muy importante y crítico a la vez al momento de realizar la interpretación de los resultados que pueden generar información equivocada para la toma de decisiones. Por tanto, esta fase está sujeta a un análisis y reevaluación más detallada del conjunto de entrenamiento o geopuntos, contemplando así la creación, eliminación y/o actualización de un registro en el conjunto de entrenamiento. Esta fase de evaluación, se apoya en la retroalimentación y puede caracterizarse como cíclica, ya que un sistema o aplicación siempre está susceptible a ser mejorado.

3.6. Fase 6: Despliegue.

En esta fase se procederá a implementar el sistema de monitoreo para la visualización de los datos obtenidos en los diferentes procesos de tratamiento de los datos en el camino a la obtención de información concisa que nos permitirá evidenciar si hubo o no riesgo en el desplazamiento de un vehículo. Así lograremos una mejor interpretación respecto a los reportes clásicos usados hasta ahora en la plataforma de monitoreo principal.

Manual de usuario del sistema

En el Anexo B, se muestran el contenido del manual del sistema con los siguientes puntos:

- Introducción.
- Carpetas y archivos del sistema.
- Descripción de los archivos principales.
- Utilización del sistema.
    - ❖ Configuración inicial.
    - ❖ Flujo de trabajo principal del sistema.

Descripción de Archivos - Código fuente del Sistema

Para mayor especificación del código fuente principal, ver Anexo C.

Tablas y Procedimientos almacenados del Sistema

Las tablas y los procedimientos almacenados necesarios para el funcionamiento del sistema, se encuentran especificados en el Anexo D.

CAPÍTULO 4

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

En esta sección se realizará el análisis y discusión de resultados a partir de la evaluación de expertos. El análisis principal se enfoca en los reportes obtenidos por el sistema propuesto en esta tesis.

4.1. Perfil de expertos

Experiencia en manejo de aplicaciones de monitoreo satelital de vehículos vinculadas a la SUTRAN.

4.2. Marco de la evaluación de expertos

Con la finalidad de evaluar la situación actual del sistema de monitoreo utilizado actualmente por la SUTRAN y respaldar el desarrollo del sistema de monitoreo vehicular propuesto en la presente tesis, se realizó la siguiente dinámica:

- Presentación del sistema propuesto mediante el desarrollo de un caso de prueba propuesto, mediante el cual se evidencia la obtención de información relevante a partir de los vectores de datos o transmisiones del desplazamiento vehicular registradas en un viaje.

- Evaluación de 10 expertos o usuarios finales del sistema requerido en los terminales terrestres, a través de un cuestionario (ver Anexo E).
- Interpretación de encuestas.

4.2.1. Caso de prueba propuesto

El viaje realizado por un Bus interprovincial con placa: A6N960, de una empresa que cuenta con el servicio vigente de monitoreo satelital de vehículos online, se resume de la siguiente manera:

- Ruta: Puno - Arequipa
- Fecha y hora de salida: 31-12-2015 16:45
- Fecha y hora de llegada: 31-12-2015 22:17
- Distancia recorrida: 312 Km.
- Transmisiones del GPS a la SUTRAN: 575 transmisiones.

Resultados del caso de prueba con el sistema de monitoreo satelital vigente

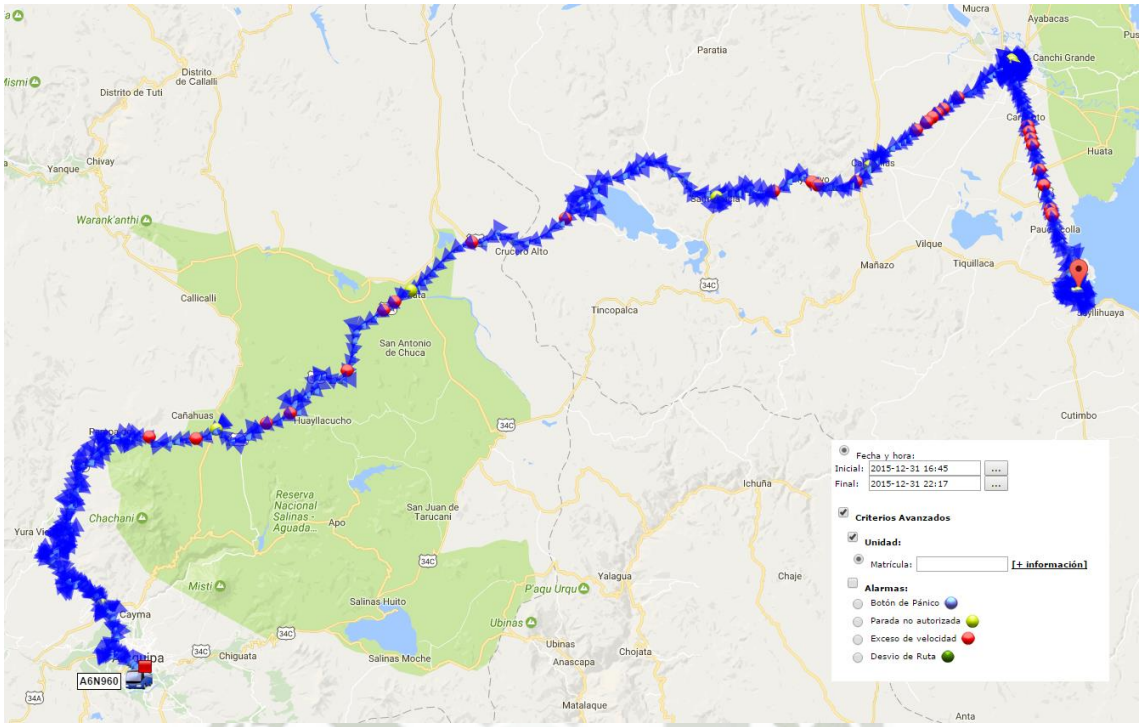


Figura 4.1. Reporte del caso de prueba con el sistema vigente.

Fuente: (Reporte de Tracking de Unidades, 2016).

La tabla 4.1, muestra un resumen de los eventos registrados para el caso de prueba con el sistema vigente, a partir de las 575 transmisiones históricas:

Tabla 4.1. Eventos registrados para el caso de prueba con el sistema vigente.

Botón de pánico	Parada no autorizada	Exceso de velocidad	Desvió de ruta
0	15	34	0

Tipo de espacio	Total de transmisiones
Recta	575
Curva Normal	0
Otro (curva peligrosa o zona urbana)	0

Fuente: Elaboración Propia.

Resultados del caso de prueba con el sistema propuesto

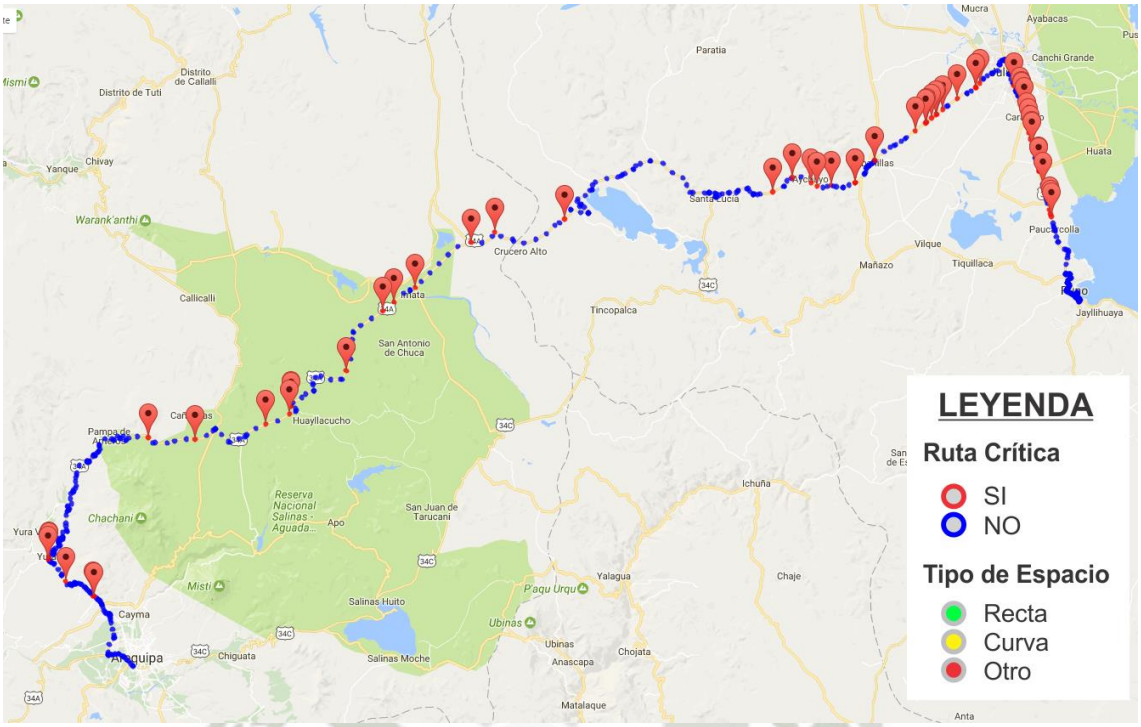


Figura 4.2. Reporte del caso de prueba con el sistema propuesto.
Fuente: Elaboración Propia.

La tabla 4.2, muestra la información obtenida para el caso de prueba con el sistema propuesto; el cual, considero a 561 transmisiones históricas válidas de las 575 iniciales.

Tabla 4.2. Información obtenida para el caso de prueba con el sistema propuesto.

Regiones críticas en rectas	Regiones críticas en curvas	Regiones críticas en curvas peligrosas o zonas urbanas
33	2	19

Tipo de espacio	Total de transmisiones
Recta	226
Curva Normal	48
Otro (curva peligrosa o zona urbana)	287 (255 Zona Urbana)

Fuente: Elaboración Propia.

4.2.2. Interpretación de encuestas

1. **¿Cómo calificaría el procedimiento de monitoreo satelital de vehículos vigente requerido por la SUTRAN en los terminales terrestres?**

Tabla 4.3. Pregunta 1: Evaluación del procedimiento de monitoreo vigente.

Respuesta	Periodicidad	Porcentaje
Bueno	0	0,0%
Regular	6	60,0%
Malo	4	40,0%
Total	10	100,0%

Fuente: Elaboración Propia.

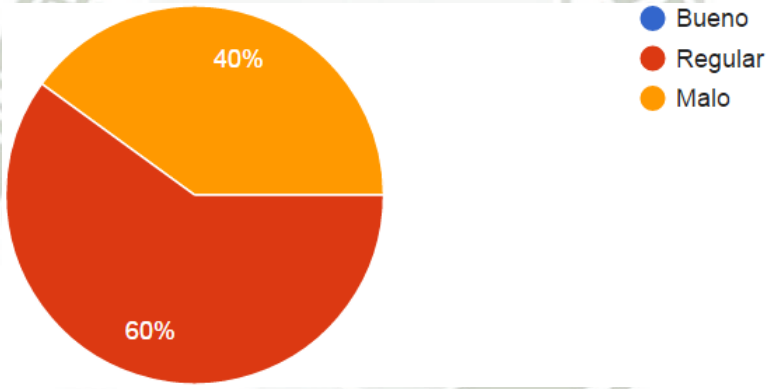


Figura 4.3. Pregunta 1: Evaluación del procedimiento de monitoreo vigente.
Fuente: Elaboración propia.

Interpretación: de acuerdo a la información obtenida un 60,0% de los expertos consideran que el procedimiento de monitoreo requerido en los terminales terrestres es regular, mientras que un 40,0% califican al procedimiento como malo. Los datos obtenidos reflejan que el procedimiento puede no estar bien definido o carece de importancia.

2. **La detección de regiones críticas en una ruta de transporte, ¿Nos puede permitir entender la magnitud real de dichos incidentes registrados?**

Tabla 4.4. Pregunta 2: Respaldo en la interpretación de la magnitud de los incidentes registrados.

Respuesta	Periodicidad	Porcentaje
Sí	10	100,0%
No	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

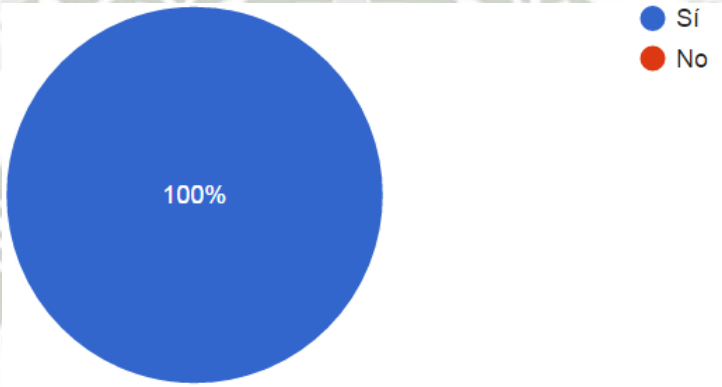


Figura 4.4. Pregunta 2: Respaldo en la interpretación de la magnitud de los incidentes registrados.
Fuente: Elaboración propia.

Interpretación: del 100% de los expertos encuestados consideran que la interpretación resumen en uno o varios viajes realizados sí puede permitirnos entender o tomar conciencia de la magnitud real de los incidentes registrados.

3. **Analizar el desplazamiento vehicular registrado según el porcentaje de regiones críticas o incidentes ocultos detectados es:**

Tabla 4.5. Pregunta 3: Análisis del desplazamiento vehicular según el porcentaje de regiones críticas detectadas.

Respuesta	Periodicidad	Porcentaje
Necesario	7	70,0%
Recomendable	3	30,0%
Innecesario	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

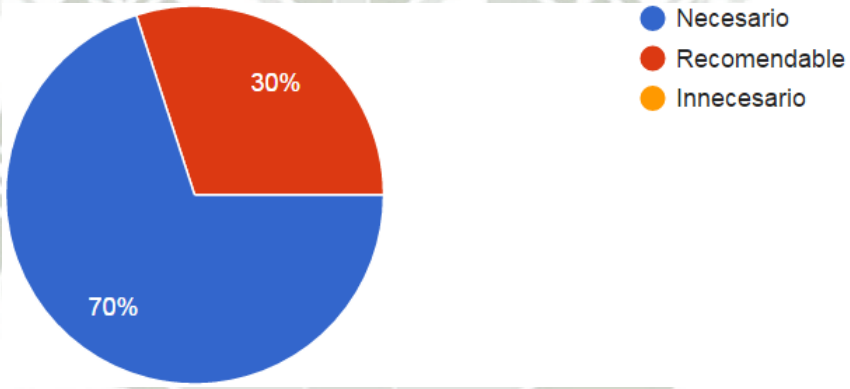


Figura 4.5. Pregunta 3: Análisis del desplazamiento vehicular según el porcentaje de regiones críticas detectadas.
Fuente: Elaboración propia.

Interpretación: definitivamente, realizar el análisis del desplazamiento vehicular registrado según el porcentaje de regiones críticas o incidentes ocultos detectados resulta necesario para el 70,0% de los encuestados y resulta recomendable para el 30,0% restante.

4. **En base al porcentaje de éxito de clasificación de una nueva transmisión en ruta, ¿Cuál es el nivel de efectividad del sistema propuesto teniendo en cuenta la nueva información obtenida (e.g., regiones críticas en zonas urbanas)?**

Tabla 4.6. Pregunta 4: Nivel de efectividad del sistema propuesto.

Respuesta	Periodicidad	Porcentaje
Muy bueno	7	70,0%
Aceptable	3	30,0%
Malo	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

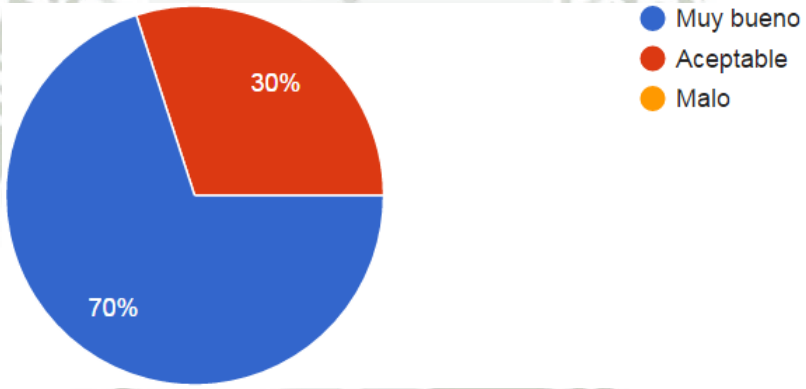


Figura 4.6. Pregunta 4: Nivel de efectividad del sistema propuesto.
Fuente: Elaboración propia.

Interpretación: 70,0% de los expertos califican el nivel de efectividad del sistema propuesto como muy bueno, mientras que el 30,0% restante califican como aceptable; a partir de estas apreciaciones, podemos señalar que la nueva información obtenida por el sistema propuesto podría ser útil.

5. **El nivel de integridad de la información contenida en el vector de datos o conjunto de atributos de un registro de transmisión del desplazamiento vehicular monitoreado con el sistema vigente, ¿Es aceptable?**

Tabla 4.7. Pregunta 5: Aceptación del nivel de integridad de la información contenida en las transmisiones del desplazamiento vehicular monitoreado.

Respuesta	Periodicidad	Porcentaje
Sí	5	50,0%
No	5	50,0%
Total	10	100,0%

Fuente: Elaboración Propia.

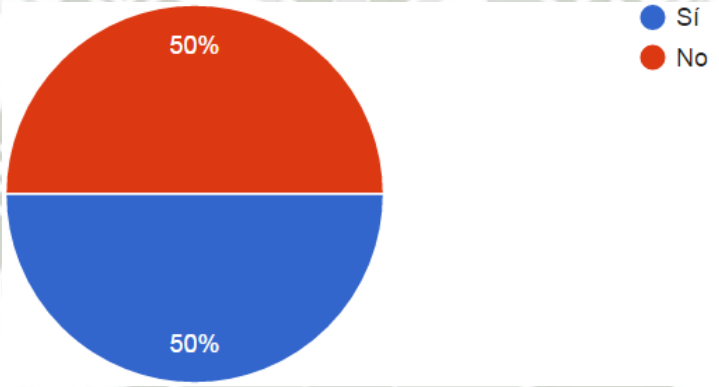


Figura 4.7. Pregunta 5: Aceptación del nivel de integridad de la información contenida en las transmisiones del desplazamiento vehicular monitoreado.
Fuente: Elaboración propia.

Interpretación: 5 expertos equivalentes al 50,0%, señalan que el nivel de integridad de la información contenida en cada una de las transmisiones sí es aceptable, y los 5 expertos restantes indican todo lo contrario. Una de las razones de esta casi inconformidad se debe a la falta de claridad del lugar de referencia vinculado en la mayoría de transmisiones vehiculares registradas.

6. **¿Considera que se necesita agregar nuevos atributos en los registros de transmisión de desplazamiento vehicular, para interpretar mejor los registros de monitoreo satelital de vehículos?**

Tabla 4.8. Pregunta 6: Necesidad de agregar nuevos atributos en los registros de transmisión.

Respuesta	Periodicidad	Porcentaje
Sí	10	100,0%
No	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

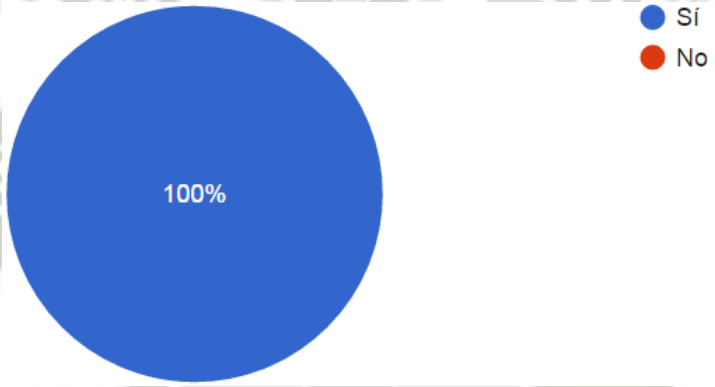


Figura 4.8. Pregunta 6: Necesidad de agregar nuevos atributos en los registros de transmisión.
Fuente: Elaboración propia.

Interpretación: el 100% de los expertos consideran que sí es necesario agregar atributos en los registros del desplazamiento vehicular monitoreado, de esta manera podemos apreciar o distinguir una nueva necesidad de los usuarios finales del sistema vigente en los terminales terrestres.

7. **La reevaluación y clasificación de las transmisiones históricas a través del modelo diseñado para detectar las regiones críticas o incidentes ocultos es:**

Tabla 4.9. Pregunta 7: Necesidad de mejorar la detección de incidentes.

Respuesta	Periodicidad	Porcentaje
Necesario	6	60,0%
Recomendable	4	40,0%
Innecesario	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

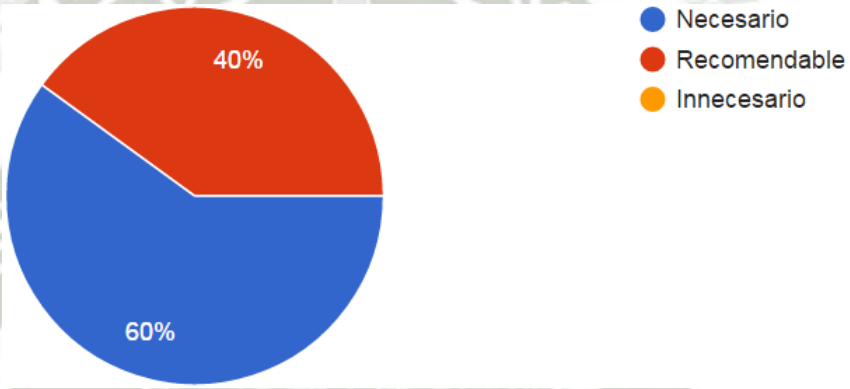


Figura 4.9. Pregunta 7: Necesidad de mejorar la detección de incidentes.
Fuente: Elaboración propia.

Interpretación: 60,0% de los encuestados, representado por 6 expertos, consideran que realizar la reevaluación de las transmisiones históricas para detectar incidentes ocultos es necesario, y el 40,0% restante señalan que realizar la reevaluación es recomendable. Tener conocimiento de todos o la mayoría de los incidentes registrados puede ser representar una ventaja competitiva.

8. Según la información obtenida con el sistema propuesto, ¿Cuál es el nivel de aceptación del modelo de clasificación de transmisiones?

Tabla 4.10. Pregunta 8: Nivel de aceptación del modelo de clasificación de transmisiones propuesto.

Respuesta	Periodicidad	Porcentaje
Alto	8	100,0%
Regular	2	20,0%
Bajo	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

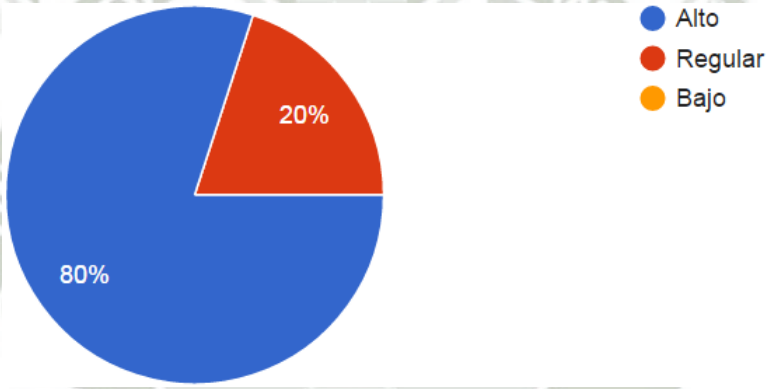


Figura 4.10. Pregunta 8: Nivel de aceptación del modelo de clasificación de transmisiones propuesto.
Fuente: Elaboración propia.

Interpretación: de acuerdo a la información obtenida un 80,0% de los expertos evaluados consideran que el nivel de aceptación del modelo propuesto es alto, mientras que un 20,0% califican a dicho modelo como regular. A partir de los datos obtenidos, podemos señalar que el modelo de clasificación de transmisiones propuesto en esta tesis es válido o aceptable.

9. **En el sistema de monitoreo vehicular propuesto, ¿Cómo considera la funcionalidad que permite obtener la información resumen de los viajes registrados?**

Tabla 4.11. Pregunta 9: Consideración de la disponibilidad de información resumen de los viajes.

Respuesta	Periodicidad	Porcentaje
Ideal	7	70,0%
Recomendable	3	30,0%
Innecesaria	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

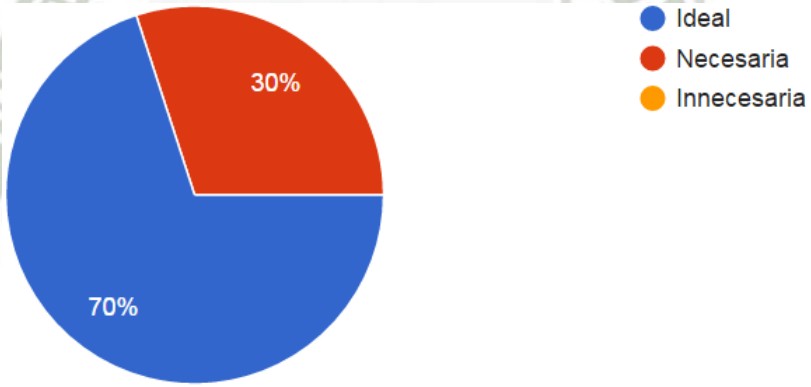


Figura 4.11. Pregunta 9: Consideración de la disponibilidad de información resumen de los viajes.

Fuente: Elaboración propia.

Interpretación: 70,0% de los expertos consideran como ideal a la funcionalidad que permite la disponibilidad de información resumen de los distintos viajes registrados, el 30,0% de los expertos restantes califican como recomendable dicha funcionalidad.

10. Como parte de la calidad del servicio de transporte brindado, monitorear los límites máximos permisibles en zonas urbanas, lugares con señalización restringida y curvas peligrosas es:

Tabla 4.12. Pregunta 10: Necesidad de geopuntos claves para un mejor control.

Respuesta	Periodicidad	Porcentaje
Necesario	7	66,7%
Recomendable	3	33,3%
Innecesario	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

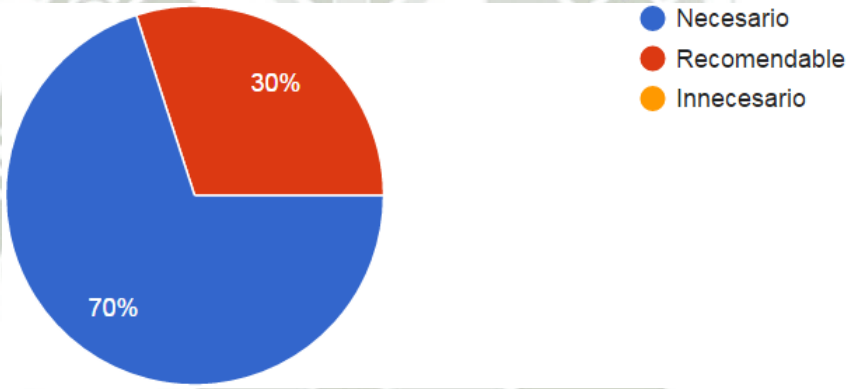


Figura 4.12. Pregunta 10: Necesidad de geopuntos claves para un mejor control.
Fuente: Elaboración propia.

Interpretación: del 100% de los encuestados, 70,0% consideran que es necesario monitorear los límites máximos permisibles según el tipo de zona correspondiente para cada una de las transmisiones registradas durante el servicio de transporte, mientras que el 30,0% restante consideran como recomendable esta medida de control de calidad del servicio brindado.

11. ¿Cuál es el nivel de aceptación de los sistemas de monitoreo satelital de vehículos utilizados en los terminales terrestres?

Tabla 4.13. Pregunta 11: Nivel de aceptación del sistema vigente.

Respuesta	Periodicidad	Porcentaje
Alto	0	0,0%
Regular	4	40,0%
Bajo	6	60,0%
Total	10	100,0%

Fuente: Elaboración Propia.

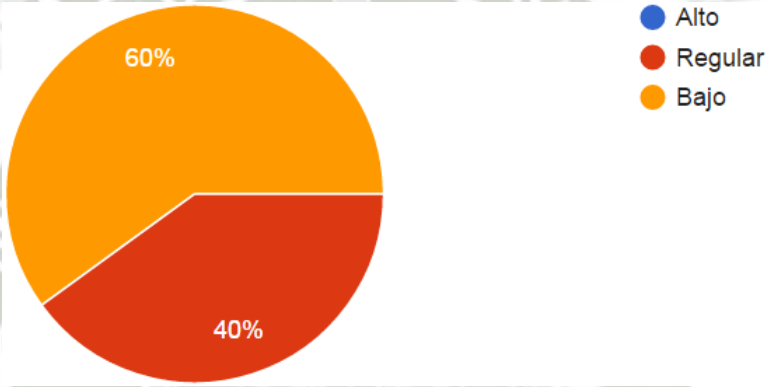


Figura 4.13. Pregunta 11: Nivel de aceptación del sistema vigente.
Fuente: Elaboración propia.

Interpretación: de los 10 expertos, 40,0% que representan a 4 encuestados, consideran que el nivel de aceptación de los sistemas utilizados en los terminales terrestres es regular, mientras que 60,0% restante considera que el nivel de aceptación es bajo.

12. ¿Cómo considera el nuevo flujo propuesto para revisar el desplazamiento histórico de un vehículo a nivel de viajes?

Tabla 4.14. Pregunta 12: Consideración del nuevo flujo de trabajo propuesto para revisar el desplazamiento histórico.

Respuesta	Periodicidad	Porcentaje
Adecuado	8	80,0%
Se puede mejorar	2	20,0%
Inadecuado	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

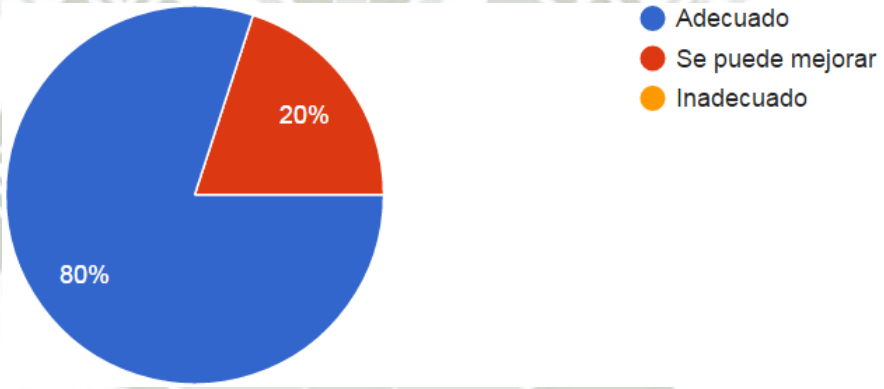


Figura 4.14. Pregunta 12: Consideración del nuevo flujo de trabajo propuesto para revisar el desplazamiento histórico.
Fuente: Elaboración propia.

Interpretación: el nuevo flujo propuesto es aceptable, debido a que el 80,0% de los expertos coinciden que el flujo propuesto para revisar el desplazamiento histórico a nivel de viajes es muy necesario, y el resto de expertos equivalente al 20,0%, indican que el flujo propuesto en esta tesis, se puede mejorar.

13. **El flujo de trabajo del sistema propuesto en comparación con los sistemas utilizados en los terminales terrestres, ¿Ofrece un mejor control de las transmisiones del desplazamiento vehicular?**

Tabla 4.15. Pregunta 13: Mejora en el control de las transmisiones vehiculares monitoreadas.

Respuesta	Periodicidad	Porcentaje
Sí	10	100,0%
No	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

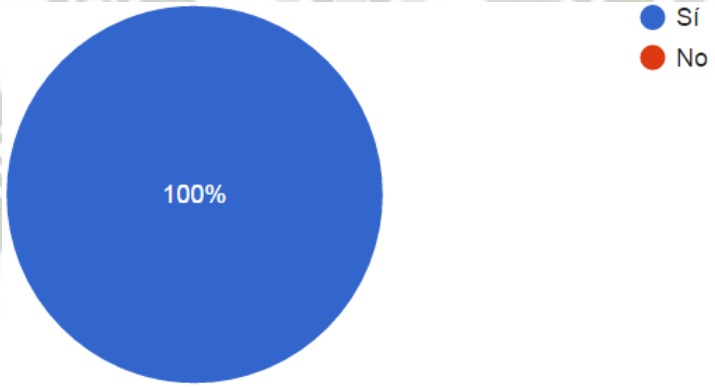


Figura 4.15. Pregunta 13: Mejora en el control de las transmisiones vehiculares monitoreadas.
Fuente: Elaboración propia.

Interpretación: el 100% de los expertos señalan que para el flujo de trabajo obtenido en el sistema propuesto en comparación a los sistemas utilizados en los terminales terrestres, sí ofrece un mejor control de las transmisiones del desplazamiento vehicular monitoreado.

14. ¿Considera que el sistema propuesto puede servir para la toma de decisiones relacionada con el servicio de transporte terrestre?

Tabla 4.16. Pregunta 14: Posibilidad de utilizar el sistema propuesto para la toma de decisiones.

Respuesta	Periodicidad	Porcentaje
Sí	10	100,0%
No	0	0,0%
Total	10	100,0%

Fuente: Elaboración Propia.

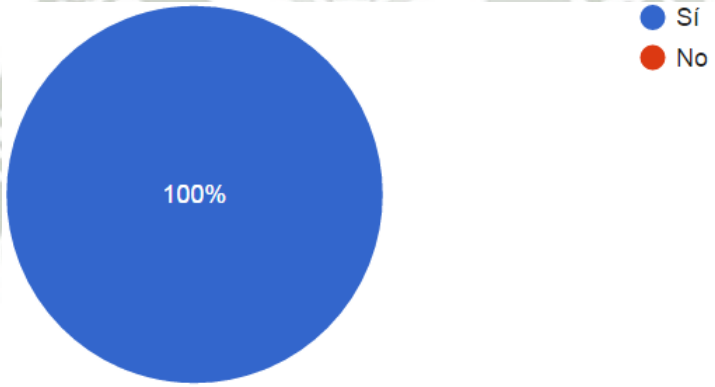


Figura 4.16. Pregunta 14: Posibilidad de utilizar el sistema propuesto para la toma de decisiones.

Fuente: Elaboración propia.

Interpretación: el 100% de los expertos consideran que el sistema propuesto sí puede servir para realizar la toma de decisiones relacionada con el servicio de transporte terrestre.

CONCLUSIONES

1. Es factible desarrollar un sistema de monitoreo vehicular que permita determinar las regiones críticas o incidentes detectados en una ruta de transporte terrestre; dicha solución tecnológica se muestra como una herramienta para realizar un mejor análisis del desplazamiento vehicular monitoreado.

2. El modelo de clasificación supervisada de transmisiones cumple con su finalidad y fue validado por los usuarios finales o personal encargado de realizar el monitoreo.

3. La efectividad del sistema propuesto se ve reflejada en la obtención del nuevo conocimiento útil, considerando además, que la funcionalidad para personalizar los límites de velocidad permisible para una determinada región o geopunto es una necesidad de los usuarios finales.

4. Sí es posible obtener y/o proponer un flujo de trabajo orientado a desarrollar un mejor control de los viajes registrados, el cual brinda un flujo de trabajo diferenciado y más completo en comparación con el sistema vigente requerido en los terminales terrestres.

5. Este sistema propuesto, puede representar el respaldo necesario en la toma de decisiones debido a que genera un valor agregado a los almacenes de datos de los sistemas de monitoreo satelital de vehículos utilizados en los terminales terrestres.

RECOMENDACIONES Y TRABAJOS FUTUROS

1. Se recomienda disminuir el intervalo de transmisión de datos del equipo GPS al Centro de Control y Monitoreo de Flota de la SUTRAN para obtener resultados más favorables; debido a que, la posibilidad de transmitir en un perímetro correspondiente a un geopunto de riesgo considerable o zonas de incidentes ocultos, es inversamente proporcional al intervalo de tiempo.

2. Si existe la posibilidad de implementar un monitoreo más avanzado, en el cual intervengan datos de sensores de humedad, calor, luz, temperatura y otros tanto del ambiente interno como externo del vehículo; el sistema propuesto, se puede complementar con el uso de algoritmos como redes neuronales u otros algoritmos de clasificación.

3. Se recomienda integrar un módulo de detección de incidentes ocultos al sistema vigente de monitoreo, para brindar una mayor funcionalidad al sistema de monitoreo y mejorar el control de la calidad del servicio brindado.

4. Tratándose de un sistema de monitoreo satelital de vehículos que realizan viajes de 4 a 8 horas en promedio, se recomienda agregar una funcionalidad para realizar el control según a la hoja de ruta predeterminada y personalizada según la política de cada empresa.

5. Se recomienda tomar como referencia la presente investigación para mejorar los procedimientos, funcionalidades, cualidades y/o alcances del sistema de monitoreo satelital de vehículos vigente en el Perú.

REFERENCIAS

Accidentes Declarados en las Unidades de la PNP (2015). *Accidentes de tránsito fatales y no fatales por año, según causa: 2006-2015*. Recuperado el 28 de Julio de 2015, de http://www.mtc.gob.pe/estadisticas/files/cuadros/Transportes_Carretero_2_5_2.xlsx

Basulto, M. (2013). *Descubrimiento de conocimiento sobre accidentes de tránsito en una base de datos concerniente a las afectaciones a la infraestructura de las telecomunicaciones en ETECSA.* La Habana: Instituto de Cibernética, Matemática y Física.

Centro de Control y Monitoreo de Flotas (2015). Recuperado el 17 de Enero de 2015, de http://www.sutran.gob.pe/monitoreo-gps/

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. & Wirth, R. (2000). *CRISP-DM 1.0 Step-by-step data mining*.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. & Wirth, R. (2007). *Metodología CRISP-DM para minería de datos.* Recuperado el 16 de Noviembre de 2015, de http://www.dataprix.com/CRISP-DM

Consejo Nacional de Seguridad Vial (2015). *Plan Nacional de Seguridad Vial 2015-2024*. Recuperado el 28 de Julio de 2015, de https://www.mtc.gob.pe/cnsv/Proyecto%20del%20Plan%20Nacional%20de%20Seguridad%20Vial%202015_2024.pdf

Decreto Supremo N° 009-2015-MTC (2015). *Aprueban modificaciones al Reglamento Nacional de Administración de Transporte, al Texto Único Ordenado del Reglamento Nacional de Tránsito, al Reglamento Nacional de Inspecciones Técnicas Vehiculares, al Reglamento Nacional de Licencias de Conducir Vehículos Automotores y No Motorizados de Transporte Terrestre y al Reglamento de Placa Única Nacional de Rodaje*. Recuperado el 08 de Octubre de 2015, de http://transparencia.mtc.gob.pe/idm_docs/normas_legales/1_0_3611.pdf

Decreto Supremo N° 016-2009-MTC (2009). *Texto Único Ordenado del Reglamento Nacional de Tránsito.* Recuperado el 15 de Agosto de 2015, de http://transparencia.mtc.gob.pe/idm_docs/normas_legales/1_0_1669.pdf

Decreto Supremo N° 017-2009-MTC (2009). *Reglamento Nacional de Administración de Transporte.* Recuperado el 19 de Agosto de 2015, de http://transparencia.mtc.gob.pe/idm_docs/normas_legales/1_0_2789.pdf

Elmasri, R. & Navathe, S. (2007). *Fundamentos de sistemas de bases de datos.* Madrid: Pearson, pp. 823-849.

García, F., Escobar, D. & Vásquez, L. (2010). *Minería de datos para la determinación de variables de tránsito mediante la aplicación de monitoreo satelital.* Universidad Nacional de Colombia.

Huerta, E., Mangiaterra, A. & Noguera, G. (2005). *GPS: Posicionamiento satelital.* UNR Editora, pp. 1-14.

Informe N° 224-2013/GEL - INDECOPI (2013). *Informe sobre Proyecto de Ley N° 2799/2013-CR.* Recuperado el 08 de Octubre de 2015, de https://www.indecopi.gob.pe/documents/20182/178372/INF-224-2013-GEL.pdf/14557f10-bb2e-4ed9-9b49-d104a010044b

Lara, J. (2014). *Minería de datos.* Madrid: UDIMA, pp. 9-122.

Ley N° 29380 (2009). Ley de creación de la Superintendencia de Transporte Terrestre de Personas, Carga y Mercancías (SUTRAN). Recuperado el 21 de Setiembre de 2015, de http://www.sutran.gob.pe/wp-content/uploads/2015/08/Ley_29380.pdf

Pajares, G. & Santos, M. (2006). *Inteligencia Artificial e Ingeniería del Conocimiento.* México: Alfaomega, pp. 217-231.

Palma, J., & Morales, R. (2008). *Inteligencia Artificial. Técnicas, métodos y aplicaciones*. Madrid: McGraw Hill, pp. 725-761.

Pérez, C. & Santín, D. (2007). *Minería de Datos Técnicas y Herramientas*. Madrid: Thomson, pp. 1-12.

Quinlan, J. (1986). *Induction of Decision Trees. Machine Learning*, pp. 81-106.

Reporte de Tracking de Unidades (2016). Recuperado el 02 de Enero de 2016, de http://www.uvicar.com.pe/servicios

Sarmiento, D. (2012). *Prototipo de sistema de información geográfico en entorno web para monitorear eventos en tiempo real de vehículos con GPS aplicando tecnologías de cache de mapas, AJAX y Base de Datos espaciales.* Arequipa: Universidad Católica de Santa María.

Sierra, B. (2006). *Aprendizaje automático: Conceptos básicos y avanzados.* Madrid: Pearson, pp. 9-17, 23-58.

Silberschatz, A., Korth, H. & Sudarshan, S. (2006). *Fundamentos de bases de datos*. Madrid: McGraw Hill, pp. 601-629.

ANEXO A

GLOSARIO DE TÉRMINOS

- **CRISP-DM (Cross - Industry Standard Process for Data Mining):** Estándar de proceso para la minería de datos.
- **DB (Data Base)** Base de datos.
- **DM (Data Mining)** Minería de Datos.
- **DW (Data Warehouses)** Almacenes de datos: son depósitos (o archivos) de información reunida de varios orígenes, almacenada bajo un esquema unificado en un solo sitio. Una vez reunida, los datos se almacenan mucho tiempo, lo que permite el acceso a datos históricos.
- **ETL (Extract, Transform and Load):** Proceso de Extracción, Transformación y Carga de datos.
- **ID3 (Iterative Dichotomiser 3 algorithm):** Algoritmo ID3, utilizado dentro del ámbito de la inteligencia artificial.
- **IDE (Integrated Development Enviroment)** Entorno de Desarrollo Integrado: es una herramienta que integra todos los recursos que necesita un programador para codificar, compilar, depurar, ejecutar y documentar sus programas.
- **INDECOPI:** Instituto Nacional de Defensa de la Competencia y de la Protección de la Propiedad Intelectual.
- **GPS (Global Position System)** Sistema Global de Posicionamiento.
- **KDD (Knowledge Discovery in Databases)** Descubrimiento de Conocimiento en Bases de Datos.

- **KNN (K-Nearest Neigbors algorith)** Algoritmo de K vecinos más próximos.
- **MTC:** Ministerio de Transportes y Comunicaciones.
- **MySQL:** Sistema de gestión de bases de datos relacional, multihilo y multiusuario; este software proporciona un servidor de base de datos SQL.
- **SUTRAN:** Superintendencia de Transporte Terrestre de Personas, Carga y Mercancías.
- **SQL (Structured Query Language)** Lenguaje de Consulta Estructurado.

ANEXO B

MANUAL DE USUARIO DEL SISTEMA

1. Introducción

El sistema propuesto está desarrollado con la IDE NetBeans y corresponde a una aplicación de escritorio para utilizar al máximo los recursos disponibles como la memoria que posibilita la realización de operaciones aritmética y lógicas en tiempo de ejecución; para el seguimiento de este manual, se asume que el desarrollador posee conocimientos en el lenguaje de programación Java. Este manual brinda los pasos principales para realizar la implementación del sistema propuesto.

2. Carpetas y archivos del sistema

La distribución de los archivos está contenida en carpetas como se muestra en la siguiente figura.

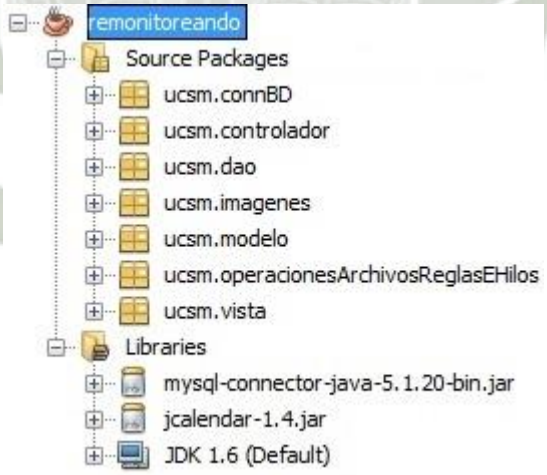


Figura Anexo B.1. Carpetas y archivos del sistema.

Fuente: Elaboración Propia.

En la siguiente tabla, se detalla la descripción del contenido de las carpetas y archivos del sistema.

Tabla Anexo B.1.

Descripción del contenido del sistema.

Nombre	Nombre Completo	Descripción
remonitoreando	Remonitoreando	Es la carpeta principal contenedora de todo el sistema.
src	Source Packages	Esta carpeta contiene todo el código fuente del sistema distribuido en los siguientes paquetes: ucsm.connDB ucsm.controlador ucsm.dao ucsm.imagenes ucsm.modelo ucsm.operacionesArchivosReglasEHilos ucsm.vista
libraries	Libraries	Contiene las librerías auxiliares necesarias para el funcionamiento correcto del sistema mysql-connector-java-5.1.20-bin.jar jcalendar-1.4.jar JDK 1.6 (Default)

Fuente: Elaboración Propia.

3. Descripción de los archivos principales

- **Carpeta src**

  ❖ Carpeta ucsm.connDB

  - Conexión.java: Clase para conexión a la base de datos “monitoreo”.
  - Mensajes.java: Clase para visualizar el resultado de la sentencia sql mediante el lenguaje de manipulación de datos realizado.

Tabla Anexo B.2.

Descripción de funciones de las clases en ucsm.connDB.

Método	Parámetros Entrada	Parámetros Salida	Descripción
conexionDB	String operacion	Connection	Recibe una cadena referencial al tipo de operación que hace la llamada a la conexión de base de datos.
cerrarConexionDB	-	-	Cierra la conexión a la base de datos.
mostrarMensajeSentenciaSql	int rptaDML, String objetoNombre	-	Recibe la respuesta según el éxito del tipo de sentencia sql ejecutada y el nombre del objeto, para visualizar el mensaje según corresponda al tipo de operación realizado.

Fuente: Elaboración Propia.

- Carpeta ucsm.controlador

  - Cargos.java: Clase Cargos para controlar el formulario VICargo.
  - Geopuntos.java: Clase Geopuntos para controlar el formulario VIGeopunto.
  - Rutas.java: Clase Rutas para controlar el formulario VIRuta.
  - Trabajadores.java: Clase Trabajadores para controlar el formulario VITrabajador.
  - Transmisiones.java: Clase Transmisiones para controlar el formulario VITransmision.
  - Vehiculos.java: Clase Vehiculos para controlar el formulario VIVehiculo.
  - Viajes.java: Clase Viajes para controlar el formulario VIViaje.

Tabla Anexo B.3.

Descripción de funciones de las clases en ucsm.controlador.

Método	Parámetros Entrada	Parámetros Salida	Descripción
Cargos	FICargo fICargo, CargoDAO cargoDAO	-	Constructor de la clase, inicializa los contenedores y controles del formulario FICargo y el componente cargoDAO.
habilitarControlesC	boolean b	-	Habilita los controles del formulario FICargo según el tipo del valor booleano.
limpiarElementosC	-	-	Restablece los valores de los controles en el formulario FICargo.
llenarTablaCargos	JTable tablaCargos, String busqueda	-	Llena en el formulario FICargo la tabla contenedora de registros de Cargos según la cadena de búsqueda enviada.

actionPerformed	ActionEvent e	-	Método para captar y ejecutar la acción para el evento seleccionado correspondiente a los siguientes botones en el formulario FICargo: - fICargo.btnNuevo - fICargo.btnBuscar - fICargo.btnEditar - fICargo.btnEliminar - fICargo.btnGuardar - fICargo.btnCancelar
Geopuntos	FIGeopunto fIGeopunto, GeopuntoDAO geopuntoDAO	-	Constructor de la clase, inicializa los contenedores y controles del formulario FIGeopunto y el componente geopuntoDAO.
habilitarControlesG	boolean b	-	Habilita los controles del formulario FIGeopunto según el tipo del valor booleano.
limpiarElementosG	-	-	Restablece los valores de los controles en el formulario FIGeopunto.
llenarTablaGeopuntos	JTable tablaGeopuntos, String busqueda	-	Llena en el formulario FIGeopunto la tabla contenedora de registros de Geopuntos según la cadena de búsqueda enviada.
actionPerformed	ActionEvent e	-	Método para captar y ejecutar la acción para el evento seleccionado correspondiente a los siguientes botones en el formulario FIGeopunto: - fIGeopunto.btnNuevo - fIGeopunto.btnBuscar - fIGeopunto.btnEditar - fIGeopunto.btnEliminar - fIGeopunto.btnGuardar - fIGeopunto.btnCancelar - fIGeopunto.btnAdministrar Rutas - fIGeopunto.btnAsignarRuta

			- fIGeopunto.btnQuitarRuta - fIGeopunto.btnAdministrar Geopuntos
Rutas	FIRuta fIRuta, RutaDAO rutaDAO	-	Constructor de la clase, inicializa los contenedores y controles del formulario FIRuta y el componente rutaDAO.
habilitarControlesR	boolean b	-	Habilita los controles del formulario FIRuta según el tipo del valor booleano.
limpiarElementosR	-	-	Restablece los valores de los controles en el formulario FIRuta.
llenarTablaRutas	JTable tablaRutas, String busqueda	-	Llena en el formulario FIRuta la tabla contenedora de registros de Rutas según la cadena de búsqueda enviada.
actionPerformed	ActionEvent e	-	Método para captar y ejecutar la acción para el evento seleccionado correspondiente a los siguientes botones en el formulario FIRuta: - fIRuta.btnNuevo - fIRuta.btnBuscar - fIRuta.btnEditar - fIRuta.btnEliminar - fIRuta.btnGuardar - fIRuta.btnCancelar - fIRuta.btnImportarGeopuntos - fIRuta.btnExportarRuta
leerArchivoGeopuntos	int rutaId	-	Método para leer un archivo para importar Geopuntos en una determinada ruta.
escribirArchivoGeopuntos	int rutaId	-	Método para escribir un archivo para exportar Geopuntos de una determinada ruta.
Trabajadores	FITrabajador	-	Constructor de la clase,

	fITrabajador, TrabajadorDAO trabajadorDAO		inicializa los contenedores y controles del formulario FITrabajador y el componente trabajadorDAO.
habilitarControlesT	boolean b	-	Habilita los controles del formulario FITrabajador según el tipo del valor booleano.
limpiarElementosT	-	-	Restablece los valores de los controles en el formulario FITrabajador.
llenarTablaTrabajadores	JTable tablaTrabajadores, String busqueda	-	Llena en el formulario FITrabajador la tabla contenedora de registros de Trabajadores según la cadena de búsqueda enviada.
llenarComboCargos	-	-	Carga y llena el contenedor de elementos de descripción de cargos almacenados en la DB.
actionPerformed	ActionEvent e	-	Método para captar y ejecutar la acción para el evento seleccionado correspondiente a los siguientes botones en el formulario FITrabajador: - fITrabajador.btnNuevo - fITrabajador.btnBuscar - fITrabajador.btnEditar - fITrabajador.btnEliminar - fITrabajador.btnGuardar - fITrabajador.btnCancelar
Transmisiones	FITransmision fITransmision, TransmisionDAO transmisionDAO	-	Constructor de la clase, inicializa los contenedores y controles del formulario FITransmision y el componente transmisionDAO.
llenarTablaTransmisiones	JTable tablaTransmisiones, String busqueda	-	Llena en el formulario FITransmision la tabla contenedora de registros de Transmisiones según la cadena de búsqueda enviada.

llenarCombo Vehiculos	-	-	Carga y llena el contenedor de placas de vehículos almacenados en la DB.
actionPerformed	ActionEvent e	-	Método para captar y ejecutar la acción para el evento seleccionado correspondiente a los siguientes botones en el formulario FITransmision: - fITransmision.btnBuscar - fITransmision.btnFiltrarPlacaYFechas - fITransmision.btnFiltrarFechas - fITransmision.btnImportarTransmisiones
leerArchivoTransmisiones	-	-	Método para leer un archivo para importar transmisiones generadas según el procedimiento de monitoreo actual.
Vehiculos	FIVehiculo fIVehiculo, VehiculoDAO vehiculoDAO	-	Constructor de la clase, inicializa los contenedores y controles del formulario FIVehiculo y el componente vehiculoDAO.
habilitarControlesVh	boolean b	-	Habilita los controles del formulario FIVehiculo según el tipo del valor booleano.
limpiarElementosVh	-	-	Restablece los valores de los controles en el formulario FIVehiculo.
llenarTablaVehiculos	JTable tablaVehiculos, String busqueda	-	Llena en el formulario FIVehiculo la tabla contenedora de registros de Geopuntos según la cadena de búsqueda enviada.
actionPerformed	ActionEvent e	-	Método para captar y ejecutar la acción para el evento seleccionado correspondiente a los siguientes botones en el

			formulario FIVehiculo: - fIVehiculo.btnNuevo - fIVehiculo.btnBuscar - fIVehiculo.btnEditar - fIVehiculo.btnEliminar - fIVehiculo.btnGuardar - fIVehiculo.btnCancelar
Viajes	FIViaje fIViaje, ViajeDAO viajeDAO	-	Constructor de la clase, inicializa los contenedores y controles del formulario FIViaje y el componente viajeDAO.
habilitarControlesVj	boolean b	-	Habilita los controles del formulario FIViaje según el tipo del valor booleano.
limpiarElementosVj	-	-	Restablece los valores de los controles en el formulario FIViaje.
llenarTablaViajes	JTable tablaViajes, String busqueda	-	Llena en el formulario FIViaje la tabla contenedora de registros de Viajes según la cadena de búsqueda enviada.
llenarComboVehiculos	-	-	Carga y llena el contenedor de placas de vehículos almacenados en la DB.
llenarComboRutas	-	-	Carga y llena el contenedor de descripción de rutas almacenadas en la DB.
llenarTablaConductores	JTable tablaConductores, int viajeId	-	Llena la tabla de conductores existentes y asignados en el formulario FIViaje según corresponda al tipo de búsqueda enviada por el id de un viaje.
llenarDatosViaje	int filaElegida	-	Llena los datos de un viaje seleccionado en el formulario FIViaje.
actionPerformed	ActionEvent e	-	Método para captar y ejecutar la acción para el evento seleccionado correspondiente a

			los siguientes botones en el formulario FIViaje: - fIViaje.btnNuevo - fIViaje.btnBuscar - fIViaje.btnEditar - fIViaje.btnEliminar - fIViaje.btnGuardar - fIViaje.btnCancelar - fIViaje.btnFiltrarPlacaYFechas - fIViaje.btnFiltrarFechas - fIViaje.btnAdministrarConductores - fIViaje.btnAsignarConductor - fIViaje.btnQuitarConductor - fIViaje.btnAdministrarViajes - fIViaje.btnEvaluarDesplazamiento - fIViaje.btnExportarViajeEvaluado
evaluarDesplazamientoEnTransmision	int transmisionId, int transmisionVelocidad, Object[] objGeopunto, int viajeId	-	Método para evaluar el desplazamiento en base a la velocidad de cada una de las transmisiones que finalmente serán procesadas y asociadas a un único viaje según el id de viaje.
escribirArchivoViajeEvaluado	int viajeId	-	Método para escribir un archivo para exportar las transmisiones procesadas al 100% en un determinado viaje.

Fuente: Elaboración Propia.

- Carpeta ucsm.dao

  - CargoDAO.java: esta clase brinda la interfaz de comunicación entre el formulario FICargo y la DB mediante el uso del objeto Cargo.
  - GeopuntoDAO.java: esta clase brinda la interfaz de comunicación entre el formulario FIGeopunto y la DB mediante el uso del objeto Geopunto.
  - GeopuntoEnRutaDAO.java: esta clase brinda la interfaz de comunicación entre los formularios FIRuta y FIGeopunto, y la DB mediante el uso del objeto GeopuntoEnRuta.
  - RutaDAO.java: esta clase brinda la interfaz de comunicación entre el formulario FIRuta y la DB mediante el uso del objeto Ruta.
  - TrabajadorDAO.java: esta clase brinda la interfaz de comunicación entre el formulario FITrabajador y la DB mediante el uso del objeto Trabajador.
  - TransmisionDAO.java: esta clase brinda la interfaz de comunicación entre el formulario FITransmision y la DB mediante el uso del objeto Transmision.
  - VehiculoDAO.java: esta clase brinda la interfaz de comunicación entre el formulario FIVehiculo y la DB mediante el uso del objeto Vehiculo.
  - ViajeDAO.java: esta clase brinda la interfaz de comunicación entre el formulario FIGeopunto y la DB mediante el uso del objeto Geopunto.

Tabla Anexo B.4.

Descripción de funciones de las clases en ucsm.dao.

Método	Parámetros Entrada	Parámetros Salida	Descripción
CargoDAO	-	-	Constructor de la clase, instancia las variables de conexión a la DB y de la clase Mensajes.

insertarCargo	Cargo objCargo	-	Inserta un registro Cargo en la BD.
actualizarCargo	Cargo objCargo	-	Actualiza un registro Cargo en la BD.
eliminarCargo	int cargoId	-	Elimina un registro Cargo en la BD.
obtenerCargoId	String cargoDescripcion	int	Obtiene el id numérico correspondiente a la descripción de un registro Cargo.
contarCargos	String busqueda	int	Obtiene el número de registros de Cargos según el tipo de búsqueda.
listarCargos	int filas, int columnas, String busqueda	Object[][]	Carga los registros de Cargos en un array de objetos según el tipo de búsqueda.
GeopuntoDAO	-	-	Constructor de la clase, instancia las variables de conexión a la DB y de la clase Mensajes.
insertarGeopunto	Geopunto objGeopunto, boolean operacionNormal	-	Inserta un registro Geopunto en la BD según el tipo de operación de inserción requerido.
actualizarGeopunto	Geopunto objGeopunto	-	Actualiza un registro Geopunto en la BD.
eliminarGeopunto	int geopuntoId	-	Elimina un registro Geopunto en la BD.
obtenerGeopuntoId	double geopuntoLatitud, double geopuntoLongitud	int	Obtiene el id numérico correspondiente a la descripción de un registro Geopunto.
contarGeopuntos	String busqueda	int	Obtiene el número de registros de Geopuntos según el tipo de

			búsqueda.
listarGeopuntos	int filas, int columnas, String busqueda	Object[][]	Carga los registros de Geopuntos en un array de objetos según el tipo de búsqueda.
contarGeopuntosVecinos	double latitudInicial, double latitudFinal, double longitudInicial, double longitudFinal, int rutaId	int	Obtiene el número de registros de Geopuntos existentes entre los límites especificados por latitud y longitud correspondientes a una determinada ruta.
listarGeopuntosVecinos	int filas, double latitudInicial, double latitudFinal, double longitudInicial, double longitudFinal, int rutaId	Object[][]	Carga los registros de Geopuntos existentes entre los límites especificados por latitud y longitud correspondentes a una determinada ruta.
cargarGeopuntoElegido	int geopuntoId	Object[]	Carga los valores del registro Geopunto en un objeto según el id numérico.
GeopuntoEnRutaDAO	-	-	Constructor de la clase, instancia la variable de conexión a la DB.
insertarGeopuntoEnRuta	GeopuntoEnRuta objGeopuntoEnRuta, boolean operacionNormal	-	Inserta un registro GeopuntoEnRuta en la BD según el tipo de operación de inserción requerido.
eliminarGeopuntoEnRuta	int rutaId, int geopuntoId	-	Elimina un registro GeopuntoEnRuta en la BD.
contarGeopuntosEnRuta	int rutaId	int	Obtiene el número de registros de Geopuntos correspondientes

			a una determinada ruta.
listarGeopuntosEnRuta	int filas, int columnas, int rutaId	Object[][]	Carga los registros de Geopuntos asociados a una determinada Ruta.
obtenerPromedioDeGeolocalizacionEnRuta	int rutaId	double[]	Obtiene la latitud y longitud promedios para una determinada Ruta.
verificarRegistroGeopuntoEnRuta	int rutaId, int geopuntoId	boolean	Verifica si un registro de Geopunto en Ruta ya está en la DB.
validarCredenciales	String usuarioNombre, String usuarioContrasenia	int	Obtiene el número de coincidencias en la DB para las credenciales de un usuario.
RutaDAO	-	-	Constructor de la clase, instancia las variables de conexión a la DB y de la clase Mensajes.
insertarRuta	Ruta objRuta	-	Inserta un registro Ruta en la BD.
actualizarRuta	Ruta objRuta	-	Actualiza un registro Ruta en la BD.
eliminarRuta	int rutaId	-	Elimina un registro Ruta en la BD.
obtenerRutaId	String rutaDescripcion	int	Obtiene el id numérico correspondiente a la descripción de un registro Ruta.
contarRutas	int geopuntoId, String busqueda	int	Obtiene el número de registros de Rutas según el tipo de búsqueda.
listarRutas	int filas, int columnas, int geopuntoId, String busqueda	Object[][]	Carga los registros de Rutas en un array de objetos según el tipo de búsqueda.

TrabajadorDAO	-	-	Constructor de la clase, instancia las variables de conexión a la DB y de la clase Mensajes.
insertarTrabajador	Trabajador objTrabajador	-	Inserta un registro Trabajador en la BD.
actualizarTrabajador	Trabajador objTrabajador	-	Actualiza un registro Trabajador en la BD.
eliminarTrabajador	trabajadorId	-	Elimina un registro Trabajador en la BD.
obtenerTrabajadorId	String trabajadorAbreviacion	int	Obtiene el id numérico correspondiente a la abreviación de un registro Trabajador.
contarTrabajadores	String busqueda	int	Obtiene el número de registros de Trabajadores según el tipo de búsqueda.
listarTrabajadores	int filas, int columnas, String busqueda	Object[][]	Carga los registros de Trabajadores en un array de objetos según el tipo de búsqueda.
contarConductores	int viajeId	int	Obtiene el número de registros de Trabajadores que son conductores según el tipo de búsqueda.
listarConductores	int filas, int columnas, int viajeId	Object[][]	Carga los registros de Trabajadores que son conductores en un array de objetos registrados en un determinado Viaje.
	int filas	Object[]	Carga las abreviaciones de los Trabajadores que son conductores en un array de objetos.
TrabajadorEnViajeDAO	-	-	Constructor de la clase, instancia la variable de conexión a la DB.

insertarTrabajadorEnViaje	TrabajadorEnViaje objTrabajadorEnViaje	-	Inserta un registro TrabajadorEnViaje en la BD según el tipo de operación de inserción requerido.
eliminarTrabajadorEnViaje	int viajeId, int trabajadorId	-	Elimina un registro TrabajadorEnViaje la BD.
TransmisionDAO	-	-	Constructor de la clase, instancia las variables de conexión a la DB y de la clase Mensajes.
insertarTransmision	Transmision objTransmision, String bloque, int numRegistro, int transmisionId	-	Inserta un registro histórico de Transmision en la BD.
actualizarInformacionEnTransmision	int transmisionId, int excesoVelocidad, String tipoVelocidad, int estabilidad, int incidente, int viajeId, int geopuntoId	-	Actualiza la información de un registro Transmisión en la BD.
actualizarTransmisionProcesada	Transmision objTransmision	String	Actualiza la información de un registro Transmisión en la BD.
verificarCodigoDeTransmision	String transmisionCodificada, String bloque, int numRegistro	int	Obtiene el id numérico correspondiente a un registro Transmisión que ya está registrado; si no existe, se procede a registrar en la Importación de Transmisiones.
contrarTransmisiones	int viajeId, int vehiculoId, String fechaInicial,	int	Obtiene el número de registros de Transmisiones según el tipo de búsqueda.

	String fechaFinal, String busqueda		
listarTransmisiones	int filas, int columnas, int viajeId, int vehiculoId, String fechaInicial, String fechaFinal, String busqueda	Object[][]	Carga los registros de Transmisiones en un array de objetos según el tipo de búsqueda.
recuentoDeTransmisionesConOSinIncidentes	int viajeId, String fechaInicial, String fechaFinal, boolean soloConIncidentes	int	Obtiene en número de Transmisiones registradas en un determinado Viaje y clasificadas como Transmisión con incidente.
obtenerPromedioDeGeolocalizacionEnViaje	int viajeId	double[]	Obtiene la latitud y longitud promedios para un determinado Viaje.
VehiculoDAO	-	-	Constructor de la clase, instancia las variables de conexión a la DB y de la clase Mensajes.
insertarVehiculo	Vehiculo objVehiculo	-	Inserta un registro Vehículo en la BD.
actualizarVehiculo	Vehiculo objVehiculo	-	Actualiza un registro Vehículo en la BD.
eliminarVehiculo	vehiculoId	-	Elimina un registro Vehículo en la BD.
obtenerVehiculoId	String vehiculoPlaca	int	Obtiene el id numérico correspondiente a la placa de un registro Vehículo.
contarVehicu	String	int	Obtiene el número de registros

los	busqueda		de Vehículos según el tipo de búsqueda.
listarVehiculos	int filas, int columnas, String busqueda	Object[][]	Carga los registros de Vehículos en un array de objetos según el tipo de búsqueda.
listarPlacasVehiculos	int filas	Object[]	Carga una lista de placas de todos los registros Vehículos.
ViajeDAO	-	-	Constructor de la clase, instancia las variables de conexión a la DB y de la clase Mensajes.
insertarViaje	Viaje objViaje	-	Inserta un registro Viaje en la BD.
actualizarViaje	Viaje objViaje	-	Actualiza un registro Viaje en la BD.
actualizarInformacionEnViaje	int viajeId, int numTransmisionesAsignadas, int numTransmisionesConIncidentes	-	Actualiza la información de un registro Viaje en la BD.
eliminarViaje	int viajeId	-	Elimina un registro Vehículo en la BD.
contarViajes	int numeroDeOperacion, int numeroDeId, String fechaInicial, String fechaFinal, String busqueda	int	Obtiene el número de registros de Viajes según el tipo de búsqueda, considerando que también se puede realizar la búsqueda por id del Vehículo o id del Trabajador
listarViajes	int filas, int columnas, int numeroDeOperacion, int numeroDeId,	Object[][]	Carga los registros de Viajes en un array de objetos según el tipo de búsqueda, considerando que también se puede realizar la búsqueda por id del

	String fechaInicial, String fechaFinal, String busqueda		Vehículo o id del Trabajador.

Fuente: Elaboración Propia.

- Carpeta ucsm.imagenes
  - icono.png: archivo del icono del sistema propuesto.
- Carpeta ucsm.modelo
  - Cargo.java: clase para representar al objeto Cargo.
  - Geopunto.java: clase para representar al objeto Cargo.
  - GeopuntoEnRuta.java: clase para representar al objeto GeopuntoEnRuta.
  - Ruta.java: clase para representar al objeto Ruta.
  - Trabajador.java: clase para representar al objeto Trabajador.
  - TrabajadorEnViaje.java: clase para representar al objeto TrabajadorEnViaje.
  - Transmision.java: clase para representar al objeto Transmision.
  - Vehiculo.java: clase para representar al objeto Vehiculo.
  - Viaje.java: clase para representar al objeto Viaje.

Tabla Anexo B.5.

Descripción de funciones de las clases en ucsm.modelo.

Método	Parámetros Entrada	Descripción
Cargo	String cargo_descripcion	Constructor de la clase Cargo.

Geopunto	double geopunto_latitud, double geopunto_longitud, String geopunto_referencia, int geopunto_limite_velocidad, String geopunto_tipo_espacio, int geopunto_diametro_perimetro, boolean geopunto_zona_urbana	Constructor de la clase Geopunto.
GeopuntoEnRuta	int rutas_ruta_id, int geopuntos_geopunto_id	Constructor de la clase GeopuntoEnRuta.
Ruta	String ruta_descripcion	Constructor de la clase Ruta.
Trabajador	String trabajador_dni, String trabajador_nombres, String trabajador_apellidos, String trabajador_abreviacion, String trabajador_email, String trabajador_telefono, int cargos_cargo_id	Constructor de la clase Trabajador.
TrabajadorEnViaje	int viajes_viaje_id, int trabajadores_trabajador_id	Constructor de la clase TrabajadorEnViaje.
Transmision	String transmision_fecha_hora, double transmision_latitud, double transmision_longitud, double transmision_velocidad, double transmision_odometro, String transmision_referencia, String transmision_codificada, int vehiculos_vehiculo_id	Constructor de la clase Transmision.
	int vehículos_vehiculo_id	Constructor de la clase Transmision para cuando se realiza la Importación de Transmisiones desde un archivo.
Vehiculo	String vehiculo_placa, String vehiculo_marca, String vehiculo_modelo, int vehiculo_anio_fabrica, String vehiculo_nro_gps, String vehiculo_nro_celular, String vehiculo_fecha_registro	Constructor de la clase Vehiculo.

Viaje	String viaje_fecha_hora_salida, String viaje_fecha_hora_llegada, int rutas_ruta_id, int vehiculos_vehiculo_id	Constructor de la clase Viaje.

Fuente: Elaboración Propia.

- ❖ Carpeta ucsm.operacionesArchivosReglasEHilos
  - ExportarRuta.java: Clase para realizar la escritura de un archivo contenedor de una determinada Ruta.
  - ExportarViaje.java: Clase para realizar la escritura de un archivo contenedor de un determinado Viaje.
  - ImportarArchivo.java: Clase para realizar la lectura de un archivo, permitiendo realizar la Importación de las Transmisiones o la Carga del conjunto de entrenamiento de las reglas para obtener los cálculos necesarios para interpretar dichas Reglas.
  - ModelarMatriz.java: Clase que contiene los cálculos realizados para obtener las reglas mediante el uso de matrices multidimensionales.
  - Reglas.java: Clase para Cargar el archivo contenedor del conjunto de entrenamiento de las reglas, Visualizar las muestras y Visualizar los cálculos a partir de dicho conjunto de entrenamiento.
  - TransmisionesConHilos.java: Clase para Importar las Transmisiones por bloques mediante el uso de hilos.

- **Carpeta Libraries**

  - ❖ mysql-connector-java-5.1.20-bin.jar: librería que permiten establecer conexiones con las bases de datos, ejecutar sentencias SQL y contar con rutas de ejecución que permiten almacenar el resultado de una consulta.
  - ❖ jcalendar-1.4.jar: librería que permite trabajar con datos del tipo Fecha.

- JDK 1.6 (Default): JDK o Java Development Kit, es un software que provee herramientas de desarrollo para la creación de programas en Java.

4. Utilización del sistema

4.1. Configuración inicial

a) Descomprimir el archivo "SISTEMA_DE_MONITOREO.rar"; contiene las carpetas: DB y remonitoreando.

b) Iniciar los servicios Apache y MySQL mediante el servidor independiente de plataforma XAMPP.

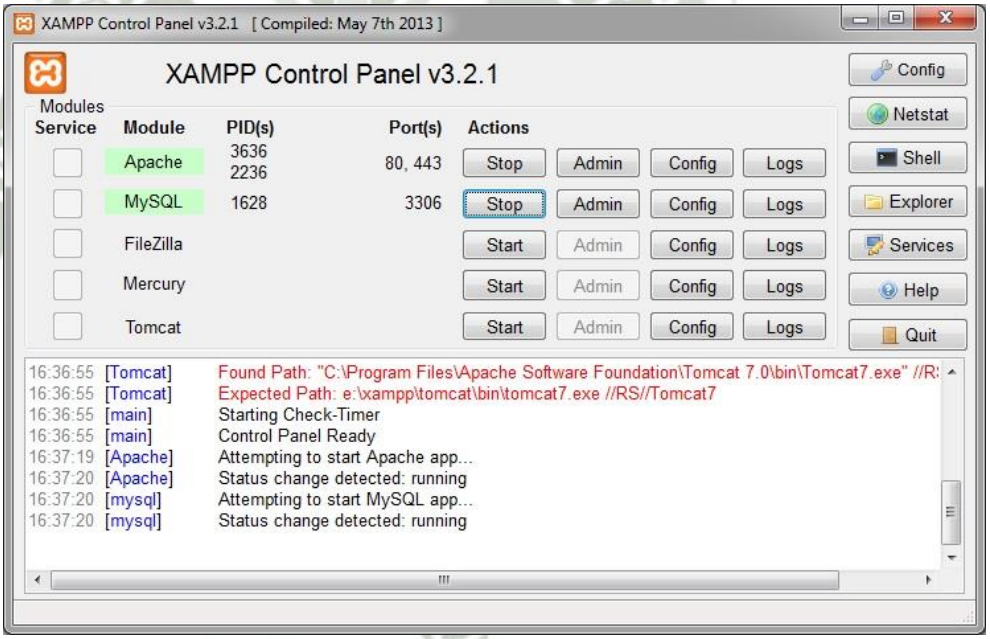


Figura Anexo B.2. Iniciar los servicios en XAMPP.
Fuente: Elaboración Propia.

c) Acceder a la url http://localhost/phpmyadmin/ y crear una nueva base de datos con el nombre de "monitoreo" e importar el script de la carpeta DB.

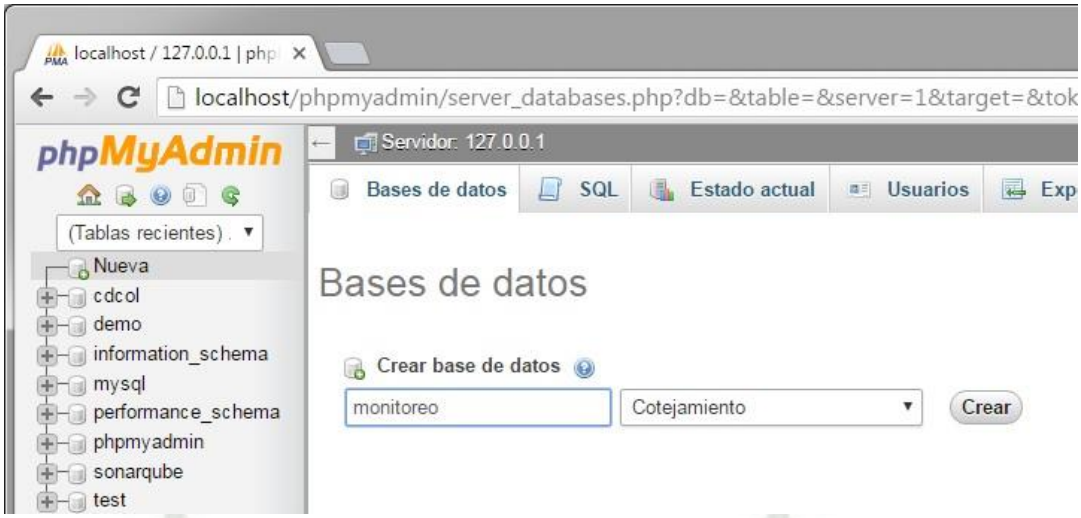


Figura Anexo B.3. Creación de la DB "monitoreo".

Fuente: Elaboración Propia.

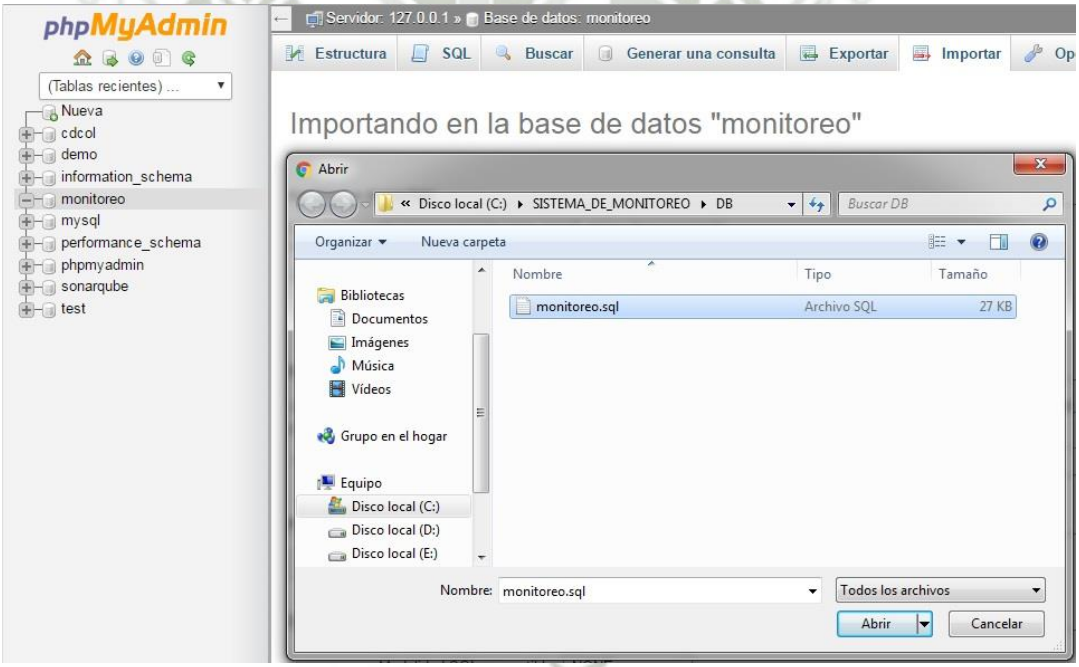


Figura Anexo B.4. Importación desde archivo "monitoreo.sql".

Fuente: Elaboración Propia.

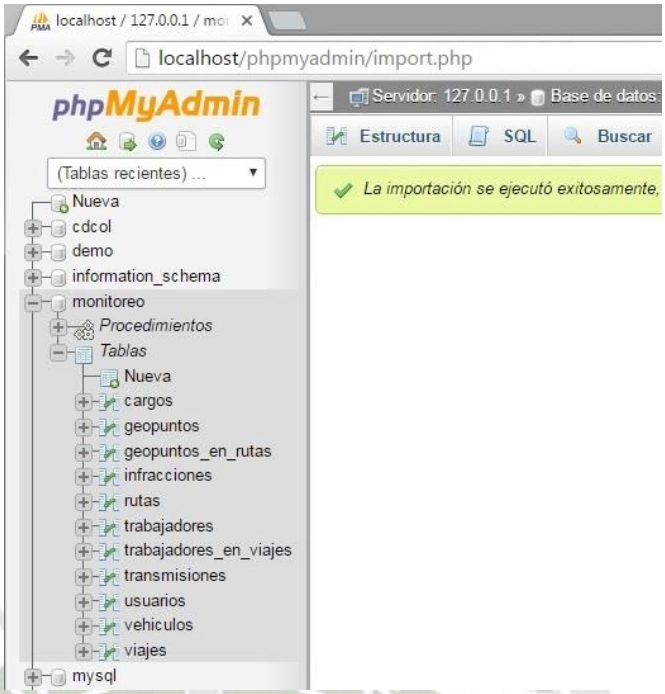


Figura Anexo B.5. Base de datos "monitoreo" operativa.

Fuente: Elaboración Propia.

d) Utilizar la IDE Netbeans para abrir el sistema propuesto "remonitoreando".

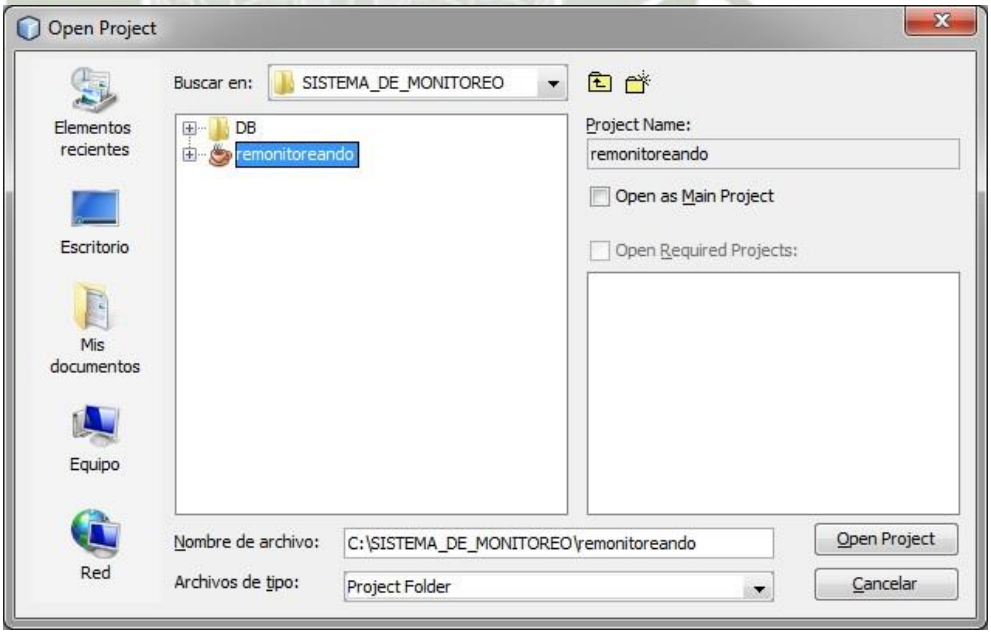


Figura Anexo B.6. Abrir proyecto "remonitoreando" en NetBeans.

Fuente: Elaboración Propia.

Para realizar este procedimiento se debe seleccionar en el menú de herramientas la opción File, luego la opción Open Project…, seguidamente se elige el proyecto remonitoreando de la carpeta correspondiente y se confirma pulsando el botón Open Project de la ventana emergente de búsqueda.

e) Ejecutar en modo desarrollo el programa para realizar el proceso de KDD a partir de las transmisiones históricas.

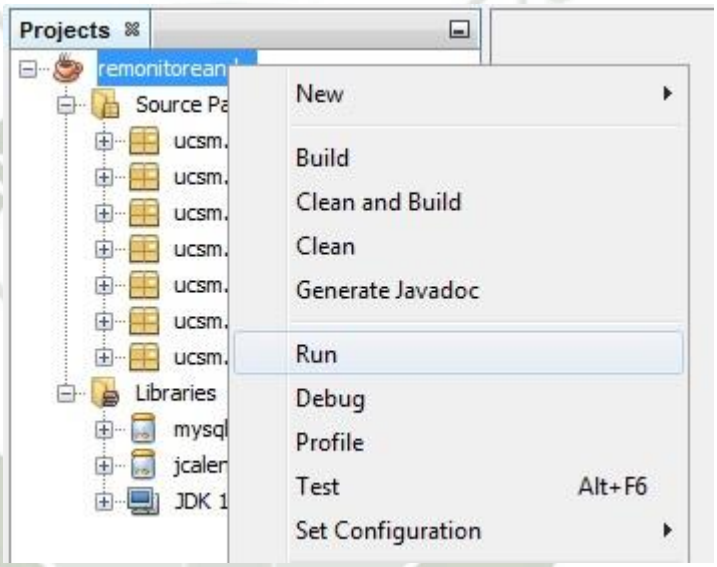


Figura Anexo B.7. Ejecución del sistema en modo de desarrollo.

Fuente: Elaboración Propia.

4.2. Flujo de trabajo principal del sistema

En esta sección, se procederá a representar el flujo principal del sistema propuesto mediante una serie de capturas de pantalla que van desde el formulario "Login" hasta el formulario "Viajes", debido a que para interactuar de forma ideal con el formulario "Viajes" se debe considerar la existencia de los registros necesarios que se utilizaran en el formulario final; tales como el conjunto de entrenamiento de los vecinos cercanos o Geopuntos, que tienen que estar previamente registrados. Además, se debe tener registros de Transmisiones históricas sin asignar a un Viaje correspondientes a los desplazamientos vehiculares realizados. Finalmente y teniendo en cuenta las consideraciones anteriores, el flujo de trabajo principal del sistema propuesto es como sigue:

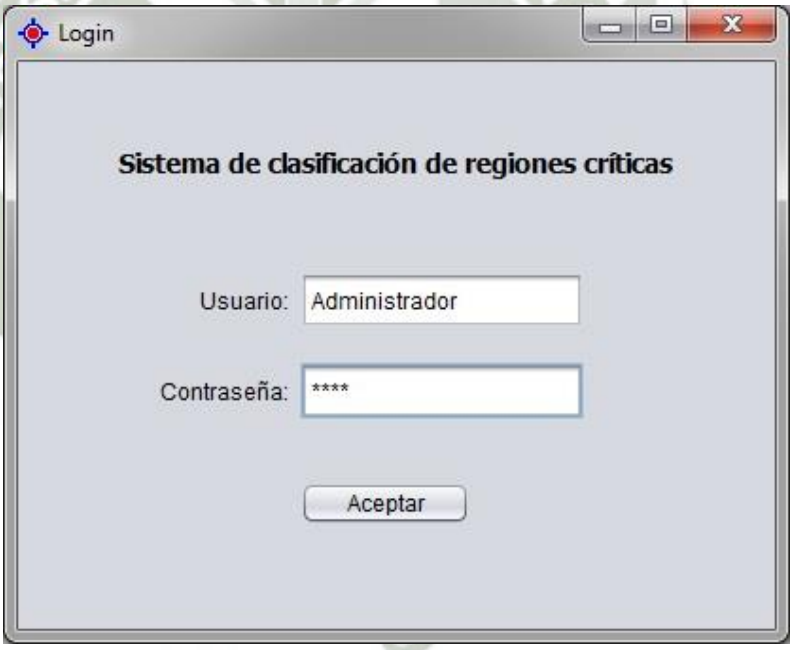


Figura Anexo B.8. Formulario "Login", para el usuario "Administrador".
Fuente: Elaboración Propia.

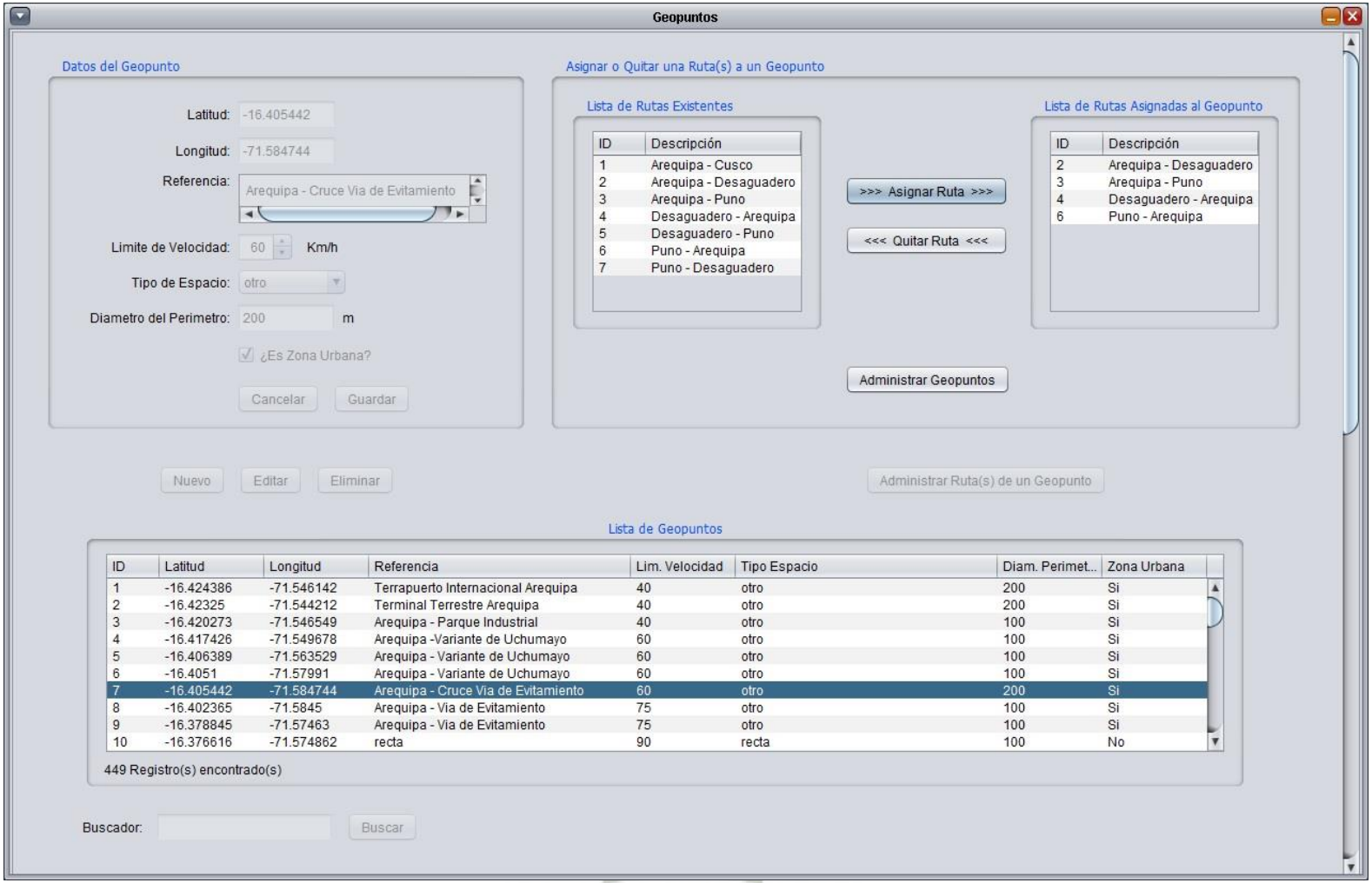


Figura Anexo B.9. Formulario para Administrar Geopuntos.

Fuente: Elaboración Propia.

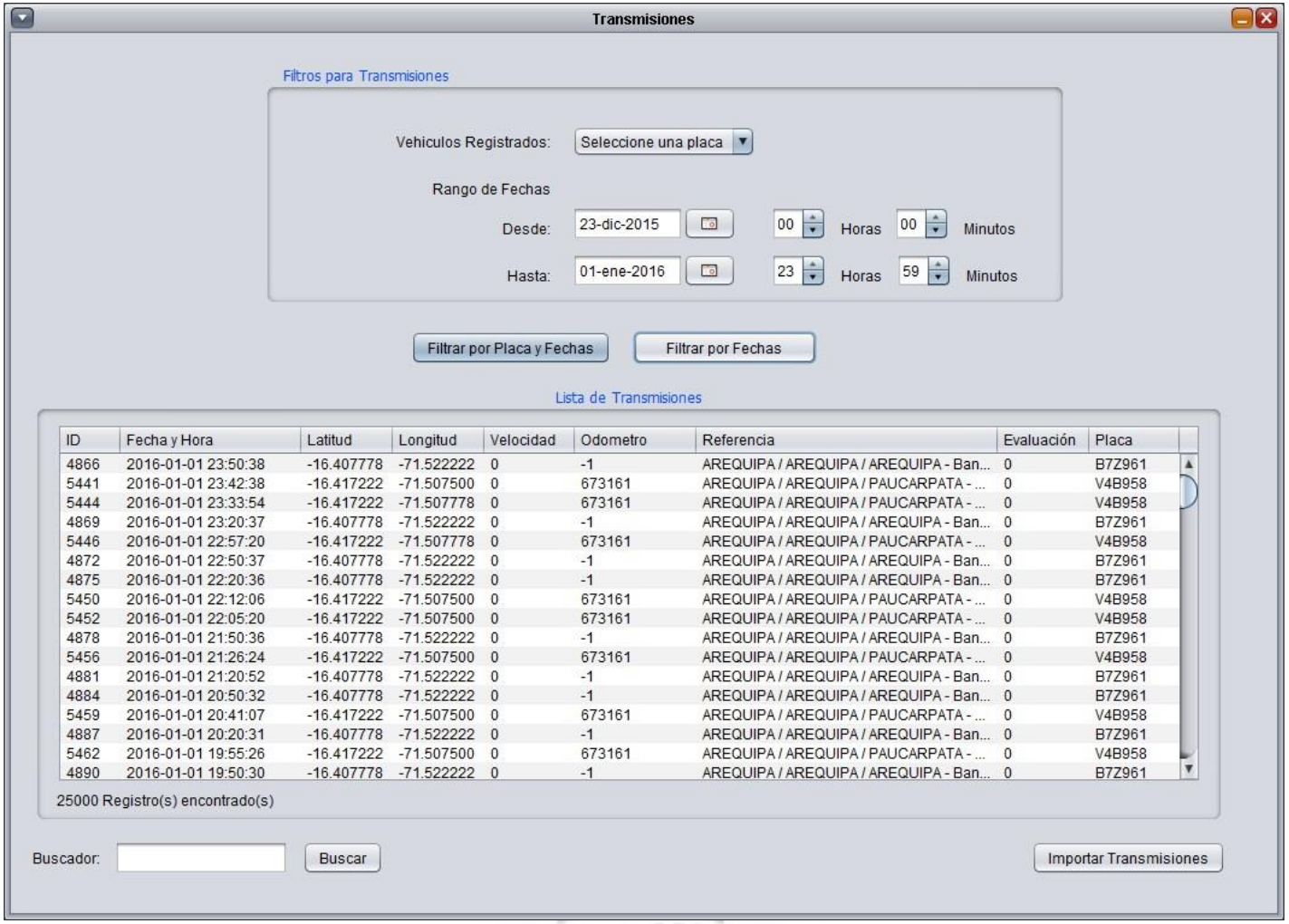


Figura Anexo B.10. Formulario para Importar Transmisiones.

Fuente: Elaboración Propia.

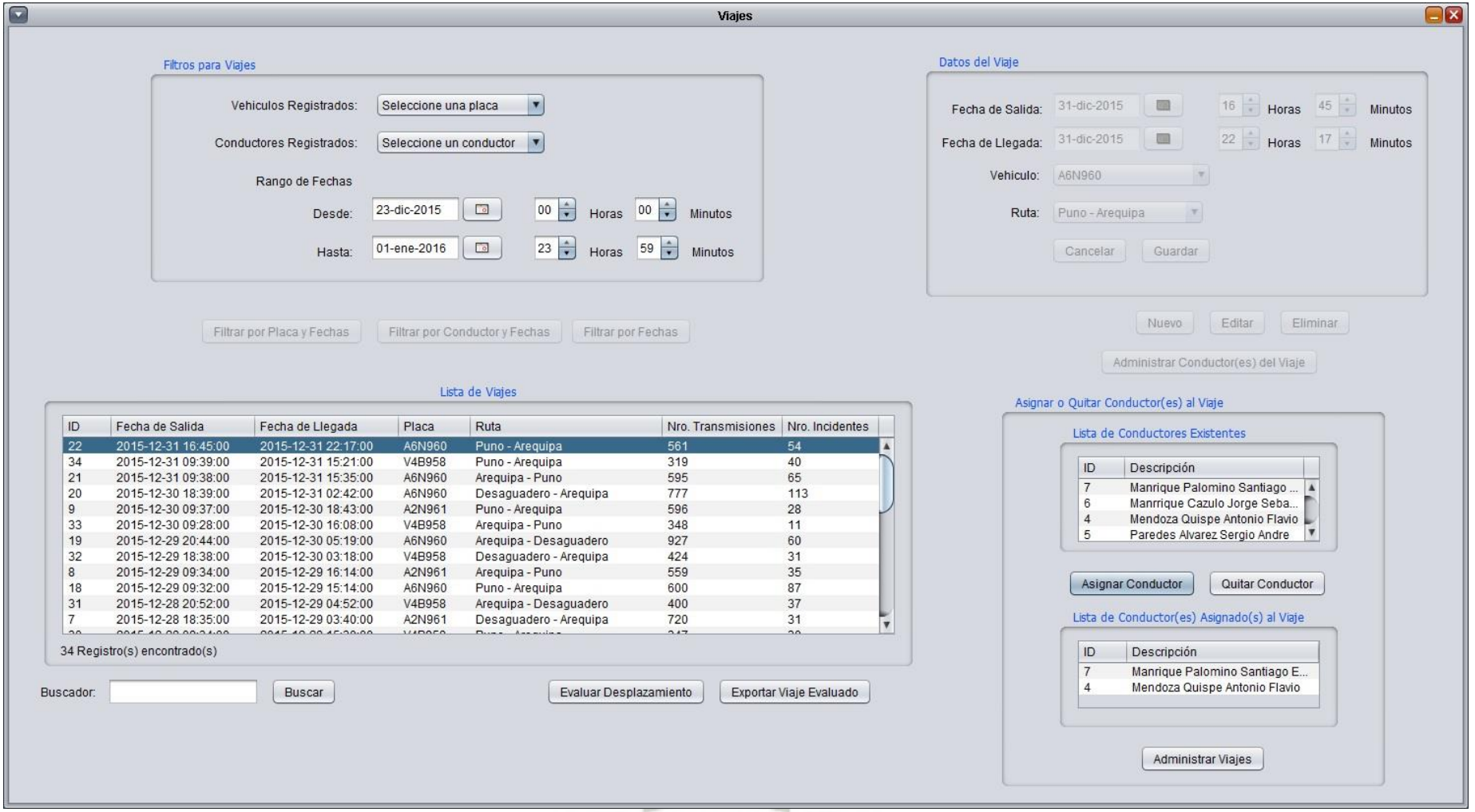


Figura Anexo B.11. Formulario para Administrar Viajes.

Fuente: Elaboración Propia.

ANEXO C

CÓDIGO FUENTE DEL SISTEMA

Las carpetas y archivos del sistema propuesto en esta tesis, están distribuidos como se indica en la siguiente figura.

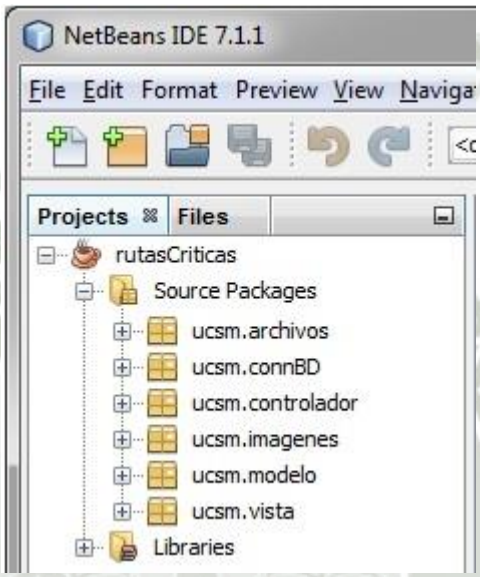


Figura Anexo C.1. Archivos y carpetas del sistema propuesto.

Fuente: Elaboración propia.

- **CARPETA UCSM.CONNBD, CONTIENE EL ARCHIVO:** Conexion.java

  - **Código fuente del archivo Conexion.java (conexión a la base de datos):**

```
public class Conexion {
   public static Connection myConn;

   public static Connection conexionDB(String operacion, Boolean
cargaMasiva) throws SQLException {
      myConn = null;
      String driver = "com.mysql.jdbc.Driver";
      String dbUrl = "jdbc:mysql://localhost:3306/monitoreo";
      String user = "root";
      String pass = "ucsm";
```

```
        try {
            Class.forName(driver);
            myConn = DriverManager.getConnection(dbUrl, user, pass);
        } catch (SQLException sqlEx) {
            System.out.println("SQL Exception: " + sqlEx.toString());
        } catch (ClassNotFoundException classEx) {
            System.out.println("Class Not Found Exception: " +
classEx.toString());
        } catch (Exception ex) {
            System.out.println("Exception: " + ex.toString());
        } finally {
            if (myConn != null) {
                if(!cargaMasiva){
                    System.out.println("Conexion Exitosa a la Base de Datos: " +
operacion);
                }
            } else {
                System.out.println("Conexion Fallida!: " + operacion);
            }
            return myConn;
        }
    }

    public static void cerrarConexionDB() throws SQLException {
        try {
            if (myConn != null) {
                myConn.close();
            }
        } catch (Exception ex) {
            System.out.println("Exception: " + ex.toString());
        }
    }
}
```

❖ **Código fuente del archivo Mensajes.java (conexión a la base de datos):**

```
public class Mensajes {

    public Mensajes() {
    }

    public void mostrarMensajeSentenciaSql(int rptaDML, String
objetoNombre) {
        String mensaje;
        switch (rptaDML) {
            case 1:
```

```
                mensaje = "Registro de "+objetoNombre+" GUARDADO con éxito.";
                break;
            case 2:
                mensaje = "Registro de "+objetoNombre+" ACTUALIZADO con éxito.";
                break;
            case 3:
                mensaje = "Registro de "+objetoNombre+" ELIMINADO con éxito.";
                break;
            case 4:
                mensaje = "El registro de "+objetoNombre+", no se puede ELIMINAR, debido a que está vinculado a un registro de otro tipo.";
                break;
            default:
                mensaje = "Registro de "+objetoNombre+" ERRÓNEO.";
                break;
        }
        JOptionPane.showMessageDialog(null, mensaje);
    }
}
```

- **CARPETA UCSM.MODELO, CONTIENE LOS ARCHIVOS:** Cargo.java, Geopunto.java, GeopuntoEnRuta.java, Ruta.java, Trabajador.java, TrabajadorEnViaje.java, Transmision.java, Vehiculo.java y Viaje.java

  - **En el archivo Cargo.java:**

    Atributos de la clase Cargo:

```
private int cargo_id;
private String cargo_descripcion;
```

  - **En el archivo Geopunto.java (conjunto de entrenamiento para el algoritmo NN):**

    Atributos de la clase Geopunto:

```
private int geopunto_id;
private double geopunto_latitud;
private double geopunto_longitud;
private String geopunto_referencia;
private int geopunto_limite_velocidad;
```

```
private String geopunto_tipo_espacio;
private int geopunto_diametro_perimetro;
private boolean geopunto_zona_urbana;
```

- **En el archivo GeopuntoEnRuta.java (relación entre geopunto-ruta):**

Atributos de la clase GeopuntoEnRuta:

```
private int rutas_ruta_id;
private int geopuntos_geopunto_id;
```

- **En el archivo Ruta.java:**

Atributos de la clase Ruta:

```
private int ruta_id;
private String ruta_descripcion;
```

- **En el archivo Trabajador.java:**

Atributos de la clase Trabajador:

```
private int trabajador_id;
private String trabajador_dni;
private String trabajador_nombres;
private String trabajador_apellidos;
private String trabajador_abreviacion;
private String trabajador_email;
private String trabajador_telefono;
private int cargos_cargo_id;
```

Constructores de la clase Trabajador:

```
public Trabajador(String trabajador_dni, String trabajador_nombres, String trabajador_apellidos, String trabajador_abreviacion, String trabajador_email, String trabajador_telefono, int cargos_cargo_id) {
    this.trabajador_dni = trabajador_dni;
    this.trabajador_nombres = trabajador_nombres;
    this.trabajador_apellidos = trabajador_apellidos;
    this.trabajador_abreviacion = trabajador_abreviacion;
    this.trabajador_email = trabajador_email;
    this.trabajador_telefono = trabajador_telefono;
    this.cargos_cargo_id = cargos_cargo_id;
}
```

❖ **En el archivo TrabajadorEnViaje.java:**

Atributos de la clase TrabajadorEnViaje:

```
private int viajes_viaje_id;
private int trabajadores_trabajador_id;
```

❖ **En el archivo Transmision.java:**

Atributos de la clase Transmision:

```
private int transmision_id;
private String transmision_fecha_hora;
private double transmision_latitud;
private double transmision_longitud;
private double transmision_velocidad;
private double transmision_odometro;
private String transmision_referencia;
private boolean transmision_evaluada;
private boolean transmision_exceso_velocidad;
private String transmision_tipo_velocidad;
private boolean transmision_dificultad_reaccion;
private boolean transmision_incidente;
private String transmision_codificada;
private int geopuntos_geopunto_id;
private int vehiculos_vehiculo_id;
private int viajes_viaje_id;
```

Constructores de la clase Transmision:

```
public Transmision(String transmision_fecha_hora, double transmision_latitud, double transmision_longitud, int transmision_velocidad, double transmision_odometro, String transmision_referencia, int vehiculos_vehiculo_id) {
    this.transmision_fecha_hora = transmision_fecha_hora;
    this.transmision_latitud = transmision_latitud;
    this.transmision_longitud = transmision_longitud;
    this.transmision_velocidad = transmision_velocidad;
    this.transmision_odometro = transmision_odometro;
    this.transmision_referencia = transmision_referencia;
    this.transmision_codificada = transmision_codificada;
    this.vehiculos_vehiculo_id = vehiculos_vehiculo_id;
}

public Transmision(int vehiculos_vehiculo_id) {
    this.vehiculos_vehiculo_id = vehiculos_vehiculo_id;
}
```

❖ **En el archivo Vehiculo.java:**

Atributos de la clase Vehiculo:

```
private int vehiculo_id;
private String vehiculo_placa;
private String vehiculo_marca;
private String vehiculo_modelo;
private int vehiculo_anio_fabrica;
private String vehiculo_nro_gps;
private String vehiculo_nro_celular;
private String vehiculo_fecha_registro;
```

❖ **En el archivo Viaje.java:**

Atributos de la clase Viaje:

```
private int viaje_id;
private String viaje_fecha_hora_salida;
private String viaje_fecha_hora_llegada;
private int viaje_nro_transmisiones;
private int viaje_nro_incidentes;
private int rutas_ruta_id;
private int vehiculos_vehiculo_id;
```

Constructor de la clase Viaje:

```
public Viaje(String viaje_fecha_hora_salida, String viaje_fecha_hora_llegada, int rutas_ruta_id, int vehiculos_vehiculo_id) {
    this.viaje_fecha_hora_salida = viaje_fecha_hora_salida;
    this.viaje_fecha_hora_llegada = viaje_fecha_hora_llegada;
    this.rutas_ruta_id = rutas_ruta_id;
    this.vehiculos_vehiculo_id = vehiculos_vehiculo_id;
}
```

- **CARPETA UCSM.DAO, CONTIENE LOS ARCHIVOS:** CargoDAO.java, GeopuntoDAO.java, GeopuntoEnRutaDAO.java, LoginDAO, RutaDAO.java, TrabajadorDAO.java, TransmisionDAO.java, TrabajadorEnViajeDAO.java, VehiculoDAO.java y ViajeDAO.java.

❖ **En el archivo LoginDAO.java:**

```
public class LoginDAO {
```

```
    Conexion conexion = new Conexion();

    public int validarCredenciales(String usuarioNombre, String
usuarioContrasenia) throws Exception {
        int rptaDML = 0;
        try {
            String sql = "call sp_validar_credenciales_usuario(?,?,?)";
            Connection accesoDB = conexion.conexionDB(sql, false);
            CallableStatement cs = accesoDB.prepareCall(sql);
            cs.setString(1, usuarioNombre);
            cs.setString(2, usuarioContrasenia);
            cs.setInt(3, Types.INTEGER);
            cs.execute();
            rptaDML = cs.getInt(3);
        } catch (Exception e) {
            System.out.println("Exception: " + e.toString());
        }
        return rptaDML;
    }
}
```

❖ **En el archivo ViajeDAO.java:**

```
public class ViajeDAO {

    Conexion conexion;
    Mensajes mensaje;

    public ViajeDAO() {conexion = new Conexion(); mensaje = new
Mensajes();}

    public String insertarViaje(Viaje objViaje) throws Exception {
        String rptaRegistro = null;
        try {
            String sql = "call sp_insertar_viaje(?,?,?,?)";
            Connection accesoDB = conexion.conexionDB(sql, false);
            CallableStatement cs = accesoDB.prepareCall(sql);
            cs.setString(1, objViaje.getViaje_fecha_hora_salida());
            cs.setString(2, objViaje.getViaje_fecha_hora_llegada());
            cs.setInt(3, objViaje.getVehiculos_vehiculo_id());
            cs.setInt(4, objViaje.getRutas_ruta_id());
            cs.execute();
            if (!cs.wasNull()) {
                rptaRegistro = "Registro de Viaje Guardado con éxito.";
            }
        } catch (Exception e) {
```

```
            System.out.println("Exception: " + e.toString());
        }
        return rptaRegistro;
    }

    public String actualizarViaje(Viaje objViaje) throws Exception {
        String rptaRegistro = null;
        try {
            String sql = "call sp_actualizar_viaje(?,?,?,?,?)";
            Connection accesoDB = conexion.conexionDB(sql, false);
            CallableStatement cs = accesoDB.prepareCall(sql);
            cs.setInt(1, objViaje.getViaje_id());
            cs.setString(2, objViaje.getViaje_fecha_hora_salida());
            cs.setString(3, objViaje.getViaje_fecha_hora_llegada());
            cs.setInt(4, objViaje.getVehiculos_vehiculo_id());
            cs.setInt(5, objViaje.getRutas_ruta_id());
            cs.execute();
            if (!cs.wasNull()) {
                rptaRegistro = "Registro de Viaje Actualizado con éxito.";
            }
        } catch (Exception e) {
            System.out.println("Exception: " + e.toString());
        }
        return rptaRegistro;
    }

    public String actualizarInformacionEnViaje(int viajeId, int
numTransmisionesAsignadas, int numTransmisionesConIncidentes)throws
Exception {
        String rptaRegistro = null;
        try {
            String sql = "call sp_actualizar_informacion_en_viaje(?,?,?)";
            Connection accesoDB = conexion.conexionDB(sql, false);
            CallableStatement cs = accesoDB.prepareCall(sql);
            cs.setInt(1, viajeId);
            cs.setInt(2, numTransmisionesAsignadas);
            cs.setInt(3, numTransmisionesConIncidentes);
            cs.execute();
            if (!cs.wasNull()) {
                rptaRegistro = "Registro de Viaje Actualizado con éxito.";
            }
        } catch (Exception e) {
            System.out.println("Exception: " + e.toString());
        }
        return rptaRegistro;
    }

    public String eliminarViaje(int viajeId) throws Exception {
```

```
        String rptaRegistro = null;
        try {
            String sql = "call sp_eliminar_viaje(?)";
            Connection accesoDB = conexion.conexionDB(sql);
            CallableStatement cs = accesoDB.prepareCall(sql);
            cs.setInt(1, viajeId);
            cs.execute();
            if (!cs.wasNull()) {
                rptaRegistro = "Registro de Viaje Eliminado con exito.";
            }
        } catch (Exception e) {
            System.out.println("Exception: " + e.toString());
        }
        return rptaRegistro;
    }

    public int contarViajes(int numeroDeOperacion, int numeroDeId, String
fechaInicial, String fechaFinal, String busqueda) {
        int rpta = 0;
        //…PreparedStatement ps;
        //…ps = accesoDB.prepareStatement("CONSULTA SQL segun el
número de operación, considerando que también se puede realizar la búsqueda
por id del Vehículo o id del Trabajador");
        //…ResultSet rs = ps.executeQuery();
        //…rpta = rs.getInt(1);
        return rpta;
    }

    public Object[][] listarViajes(int filas, int columnas, int
numeroDeOperacion, int numeroDeId, String fechaInicial, String fechaFinal,
String busqueda) {
        Object[][] listaDeViajes = new Object[filas][columnas];
        //…PreparedStatement ps;
        //…ps = accesoDB.prepareStatement("CONSULTA SQL segun el
número de operación, considerando que también se puede realizar la búsqueda
por id del Vehículo o id del Trabajador ");
        //…ResultSet rs = ps.executeQuery();
        //…listaDeViajes [cont][0] = rs.getInt(1);
        //…listaDeViajes [cont][1] = rs.getString(2);
        return listaDeViajes;
    }
```

- **CARPETA UCSM.CONTROLADOR, CONTIENE LOS ARCHIVOS:** Cargos.java, Geopuntos.java, Rutas.java, Trabajadores.java, Transmisiones.java, Vehiculos.java y Viajes.java.

❖ **En el archivo Viajes.java:**

```
public class Viajes implements ActionListener {

    FIViaje fIViaje = new FIViaje();
    ViajeDAO viajeDAO = new ViajeDAO();
    VehiculoDAO vehiculoDAO = new VehiculoDAO();
    RutaDAO rutaDAO = new RutaDAO();
    TrabajadorDAO trabajadorDAO = new TrabajadorDAO();
    TrabajadorEnViajeDAO trabajadorEnViajeDAO = new
TrabajadorEnViajeDAO();
    TransmisionDAO transmisionDAO = new TransmisionDAO();
    GeopuntoDAO geopuntoDAO = new GeopuntoDAO();

    boolean estado;
    boolean filtrarPorPlaca = true;
    boolean filtrarPorConductor = false;
    boolean administraConductores;
    boolean evaluando;
    int[] anchoColumnasV = {50, 150, 150, 75, 200, 125, 100};
    int[] anchoColumnasC = {50, 200};
    int idViaje;
    int idConductor;

    public Viajes(FIViaje fIViaje, ViajeDAO viajeDAO) {
        this.viajeDAO = viajeDAO;
        this.fIViaje = fIViaje;
        this.fIViaje.btnNuevo.addActionListener(this);
        this.fIViaje.btnEditar.addActionListener(this);
        this.fIViaje.btnEliminar.addActionListener(this);
        this.fIViaje.btnGuardar.addActionListener(this);
        this.fIViaje.btnCancelar.addActionListener(this);
        this.fIViaje.btnFiltrarPlacaYFechas.addActionListener(this);
        this.fIViaje.btnFiltrarConductorYFechas.addActionListener(this);
        this.fIViaje.btnFiltrarFechas.addActionListener(this);
        this.fIViaje.btnBuscar.addActionListener(this);
        this.fIViaje.btnAdministrarConductores.addActionListener(this);
        this.fIViaje.btnAsignarConductor.addActionListener(this);
        this.fIViaje.btnQuitarConductor.addActionListener(this);
        this.fIViaje.btnAdministrarViajes.addActionListener(this);
        this.fIViaje.btnEvaluarDesplazamiento.addActionListener(this);
        this.fIViaje.btnExportarViajeEvaluado.addActionListener(this);
        llenarComboVehiculosFiltros();
        llenarComboConductoresFiltros();
        llenarTablaViajes(fIViaje.tDatosViajes);
        llenarTablaConductores(fIViaje.tDatosConductoresExistentes, 0);
        llenarTablaConductores(fIViaje.tDatosConductoresAsignados, -1);
    }
```

```
    private void habilitarControlesVj(boolean estado) {
        if (administraConductores) {
            fIViaje.btnAsignarConductor.setEnabled(estado);
            fIViaje.btnQuitarConductor.setEnabled(estado);
            fIViaje.btnAdministrarViajes.setEnabled(estado);
            fIViaje.tDatosConductoresExistentes.enable(estado);
            fIViaje.tDatosConductoresAsignados.enable(estado);
            fIViaje.tDatosViajes.enable(!estado);

fIViaje.getRootPane().setDefaultButton(fIViaje.btnAsignarConductor);
        } else {
            if (!evaluando) {
                fIViaje.dCViajeFechaSalida.setEnabled(estado);
                fIViaje.spnrViajeFechaHorasSalida.setEnabled(estado);
                fIViaje.spnrViajeFechaMinutosSalida.setEnabled(estado);
                fIViaje.dCViajeFechaLlegada.setEnabled(estado);
                fIViaje.spnrViajeFechaHorasLlegada.setEnabled(estado);
                fIViaje.spnrViajeFechaMinutosLlegada.setEnabled(estado);
                fIViaje.cBViajeVehiculosDatos.setEnabled(estado);
                fIViaje.cBViajeRutas.setEnabled(estado);
                fIViaje.btnGuardar.setEnabled(estado);
                fIViaje.btnCancelar.setEnabled(estado);
                fIViaje.btnBuscar.setEnabled(!estado);
                fIViaje.txtFBuscadorEnViajes.enable(!estado);
                fIViaje.dCViajeFechaSalida.grabFocus();
            }
        }
        JButton[] control = {fIViaje.btnNuevo, fIViaje.btnEditar,
fIViaje.btnEliminar, fIViaje.btnFiltrarConductorYFechas,
fIViaje.btnAdministrarConductores, fIViaje.btnFiltrarPlacaYFechas,
fIViaje.btnFiltrarFechas};
        for (JButton objButton : control) {
            objButton.setEnabled(!estado);
        }
    }

    public void reiniciarContenedoresVj() {
        fIViaje.dCViajeFechaSalida.setDate(null);
        fIViaje.spnrViajeFechaHorasSalida.setValue("00");
        fIViaje.spnrViajeFechaMinutosSalida.setValue("00");
        fIViaje.dCViajeFechaLlegada.setDate(null);
        fIViaje.spnrViajeFechaHorasLlegada.setValue("00");
        fIViaje.spnrViajeFechaMinutosLlegada.setValue("00");
        llenarComboVehiculosDatos();
        llenarComboRutas();
    }
```

```
    public void llenarTablaViajes(JTable tablaViajes) {
        DefaultTableModel tViajes = new DefaultTableModel();
        tablaViajes.setModel(tViajes);
        tViajes.addColumn("ID");
        tViajes.addColumn("Fecha de Salida");
        tViajes.addColumn("Fecha de Llegada");
        tViajes.addColumn("Placa");
        tViajes.addColumn("Ruta");
        tViajes.addColumn("Nro. Transmisiones");
        tViajes.addColumn("Nro. Incidentes");
        String vehiculo_placa = (String)
fIViaje.cBViajesVehiculosFiltros.getSelectedItem();
        int vehiculoId = -1;
        if (vehiculo_placa.compareTo("Seleccione una placa") != 0) {
            vehiculoId = vehiculoDAO.obtenerVehiculoId(vehiculo_placa);
        }
        String trabajador_abreviacion = (String)
fIViaje.cBViajesConductoresFiltros.getSelectedItem();
        int trabajadorId = -1;
        if (trabajador_abreviacion.compareTo("Seleccione un conductor") != 0)
{
            trabajadorId =
trabajadorDAO.obtenerTrabajadorId(trabajador_abreviacion);
        }
        SimpleDateFormat formatoFecha = new SimpleDateFormat("yyyy-MM-
dd");
        Date fechaTmp = new Date();
        String fechaIni = null;
        if (this.fIViaje.dCViajesFechaInicio.getDate() != null) {
            fechaIni =
formatoFecha.format(fIViaje.dCViajesFechaInicio.getDate());
        } else {
            this.fIViaje.dCViajesFechaInicio.setDate(fechaTmp);
            fechaIni = formatoFecha.format(fechaTmp);
        }
        fechaIni = fechaIni + " " +
fIViaje.spnrViajesFechaHorasInicio.getValue().toString() + ":" +
fIViaje.spnrViajesFechaMinutosInicio.getValue().toString() + ":00";
        String fechaFin = null;
        if (this.fIViaje.dCViajesFechaFin.getDate() != null) {
            fechaFin =
formatoFecha.format(fIViaje.dCViajesFechaFin.getDate());
        } else {
            this.fIViaje.dCViajesFechaFin.setDate(fechaTmp);
            fechaFin = formatoFecha.format(fechaTmp);
        }
```

```
        fechaFin = fechaFin + " " + fIViaje.spnrViajesFechaHorasFin.getValue().toString() + ":" + fIViaje.spnrViajesFechaMinutosFin.getValue().toString() + ":00";
        String busqueda = fIViaje.txtFBuscadorEnViajes.getText();
        int numRegistros = 0;
        Object[][] registroV;
        if (filtrarPorPlaca) {
            numRegistros = viajeDAO.contarViajes(1, vehiculoId, fechaIni, fechaFin, busqueda);
            registroV = viajeDAO.listarViajes(numRegistros, 7, 1, vehiculoId, fechaIni, fechaFin, busqueda);
            for (int i = 0; i < numRegistros; i++) {
                tViajes.addRow(registroV[i]);
            }
        } else if (filtrarPorConductor) {
            numRegistros = viajeDAO.contarViajes(2, trabajadorId, fechaIni, fechaFin, busqueda);
            registroV = viajeDAO.listarViajes(numRegistros, 7, 2, trabajadorId, fechaIni, fechaFin, busqueda);
            for (int i = 0; i < numRegistros; i++) {
                tViajes.addRow(registroV[i]);
            }
        } else{
            numRegistros = viajeDAO.contarViajes(3, 0, fechaIni, fechaFin, busqueda);
            registroV = viajeDAO.listarViajes(numRegistros, 7, 3, 0, fechaIni, fechaFin, busqueda);
            for (int i = 0; i < numRegistros; i++) {
                tViajes.addRow(registroV[i]);
            }
        }

this.fIViaje.tDatosViajes.getColumnModel().getColumn(0).setPreferredWidth(anchoColumnasV[0]);

this.fIViaje.tDatosViajes.getColumnModel().getColumn(1).setPreferredWidth(anchoColumnasV[1]);

this.fIViaje.tDatosViajes.getColumnModel().getColumn(2).setPreferredWidth(anchoColumnasV[2]);

this.fIViaje.tDatosViajes.getColumnModel().getColumn(3).setPreferredWidth(anchoColumnasV[3]);

this.fIViaje.tDatosViajes.getColumnModel().getColumn(4).setPreferredWidth(anchoColumnasV[4]);
```

```
this.fIViaje.tDatosViajes.getColumnModel().getColumn(5).setPreferredWidth
(anchoColumnasV[5]);

this.fIViaje.tDatosViajes.getColumnModel().getColumn(6).setPreferredWidth
(anchoColumnasV[6]);
      this.fIViaje.lblContadorDeViajes.setText(numRegistros + " Registro(s)
encontrado(s)");
   }

   public void llenarComboVehiculosDatos() {
      fIViaje.cBViajeVehiculosDatos.removeAllItems();
      int numRegistros = vehiculoDAO.contarVehiculos("");
      Object[] registroV = vehiculoDAO.listarPlacasVehiculos(numRegistros);
      fIViaje.cBViajeVehiculosDatos.addItem("Seleccione una placa");
      for (int i = 0; i < numRegistros; i++) {
         fIViaje.cBViajeVehiculosDatos.addItem(registroV[i]);
      }
   }

   public void llenarComboVehiculosFiltros() {
      fIViaje.cBViajesVehiculosFiltros.removeAllItems();
      int numRegistros = vehiculoDAO.contarVehiculos("");
      Object[] registroV = vehiculoDAO.listarPlacasVehiculos(numRegistros);
      fIViaje.cBViajesVehiculosFiltros.addItem("Seleccione una placa");
      for (int i = 0; i < numRegistros; i++) {
         fIViaje.cBViajesVehiculosFiltros.addItem(registroV[i]);
      }
   }

   public void llenarComboConductoresFiltros() {
      fIViaje.cBViajesConductoresFiltros.removeAllItems();
      int numRegistros = trabajadorDAO.contarConductores(0);
      Object[] registroC = trabajadorDAO.listarConductores(numRegistros);
      fIViaje.cBViajesConductoresFiltros.addItem("Seleccione                un
conductor");
      for (int i = 0; i < numRegistros; i++) {
         fIViaje.cBViajesConductoresFiltros.addItem(registroC[i]);
      }
   }

   public void llenarComboRutas() {
      fIViaje.cBViajeRutas.removeAllItems();
      int numRegistros = rutaDAO.contarRutas(0, "");
      Object[][] registroR = rutaDAO.listarRutas(numRegistros, 2, 0, "");
      fIViaje.cBViajeRutas.addItem("Seleccione una ruta");
      for (int i = 0; i < numRegistros; i++) {
         fIViaje.cBViajeRutas.addItem(registroR[i][1]);
```

```
        }
    }

    public void llenarTablaConductores(JTable tablaConductores, int viajeId) {
        DefaultTableModel tConductores = new DefaultTableModel();
        tablaConductores.setModel(tConductores);
        tConductores.addColumn("ID");
        tConductores.addColumn("Descripción");
        int numRegistros = trabajadorDAO.contarConductores(viajeId);
        Object[][] objConductores =
trabajadorDAO.listarConductores(numRegistros, 2, viajeId);
        for (int i = 0; i < numRegistros; i++) {
            tConductores.addRow(objConductores[i]);
        }

this.fIViaje.tDatosConductoresExistentes.getColumnModel().getColumn(0).s
etPreferredWidth(anchoColumnasC[0]);

this.fIViaje.tDatosConductoresExistentes.getColumnModel().getColumn(1).s
etPreferredWidth(anchoColumnasC[1]);

this.fIViaje.tDatosConductoresAsignados.getColumnModel().getColumn(0).s
etPreferredWidth(anchoColumnasC[0]);

this.fIViaje.tDatosConductoresAsignados.getColumnModel().getColumn(1).s
etPreferredWidth(anchoColumnasC[1]);
    }

    public void llenarDatosViaje(int filaElegida) {
        llenarComboVehiculosDatos();
        llenarComboRutas();

fIViaje.txtViajeId.setText(String.valueOf(fIViaje.tDatosViajes.getValueAt(fil
aElegida, 0)));
        SimpleDateFormat formatoFecha = new SimpleDateFormat("yyyy-MM-
dd");

fIViaje.txtFViajeFechaSalida.setText(String.valueOf(fIViaje.tDatosViajes.get
ValueAt(filaElegida, 1)));
        String fechaHoraSalida = fIViaje.txtFViajeFechaSalida.getText();
        String horasSalida = fechaHoraSalida.substring(11, 13);
        String minutosSalida = fechaHoraSalida.substring(14, 16);
        Date fechaSalida = null;
        try {
            fechaSalida = formatoFecha.parse(fechaHoraSalida);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
```

```
        fIViaje.dCViajeFechaSalida.setDate(fechaSalida);
        fIViaje.spnrViajeFechaHorasSalida.setValue(horasSalida);
        fIViaje.spnrViajeFechaMinutosSalida.setValue(minutosSalida);

fIViaje.txtFViajeFechaSalida.setText(String.valueOf(fIViaje.tDatosViajes.get
ValueAt(filaElegida, 2)));
        String fechaHoraLlegada = fIViaje.txtFViajeFechaSalida.getText();
        String horasLlegada = fechaHoraLlegada.substring(11, 13);
        String minutosLlegada = fechaHoraLlegada.substring(14, 16);
        Date fechaLlegada = null;
        try {
            fechaLlegada = formatoFecha.parse(fechaHoraLlegada);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        fIViaje.dCViajeFechaLlegada.setDate(fechaLlegada);
        fIViaje.spnrViajeFechaHorasLlegada.setValue(horasLlegada);
        fIViaje.spnrViajeFechaMinutosLlegada.setValue(minutosLlegada);

fIViaje.cBViajeVehiculosDatos.setSelectedItem(String.valueOf(fIViaje.tDato
sViajes.getValueAt(filaElegida, 3)));

fIViaje.cBViajeRutas.setSelectedItem(String.valueOf(fIViaje.tDatosViajes.ge
tValueAt(filaElegida, 4)));
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == fIViaje.btnNuevo) {
            habilitarControlesVj(true);
            reiniciarContenedoresVj();
            fIViaje.getRootPane().setDefaultButton(fIViaje.btnGuardar);
            estado = true;
        }
        if (e.getSource() == fIViaje.btnBuscar) {
            filtrarPorPlaca = false;
            filtrarPorConductor = false;
            llenarTablaViajes(fIViaje.tDatosViajes);

fIViaje.getRootPane().setDefaultButton(fIViaje.btnFiltrarPlacaYFechas);
            fIViaje.txtFBuscadorEnViajes.setText("");
        }
        if (e.getSource() == fIViaje.btnEditar) {
            int filaAEditar = fIViaje.tDatosViajes.getSelectedRow();
            int filasSeleccionadas = fIViaje.tDatosViajes.getSelectedRowCount();
            if (filaAEditar >= 0 && filasSeleccionadas == 1) {
                habilitarControlesVj(true);
                fIViaje.getRootPane().setDefaultButton(fIViaje.btnGuardar);
                estado = false;
```

```
            llenarDatosViaje(filaAEditar);
        } else {
            JOptionPane.showMessageDialog(null, "Debe seleccionar un registro en la lista de Viajes.");
        }
    }
    if (e.getSource() == fIViaje.btnEliminar) {
        try {
            int filaAEliminar = fIViaje.tDatosViajes.getSelectedRow();
            int filasSeleccionadas = fIViaje.tDatosViajes.getSelectedRowCount();
            if (filaAEliminar >= 0 && filasSeleccionadas == 1) {
                int viajeId = (Integer) (fIViaje.tDatosViajes.getValueAt(filaAEliminar, 0));
                String viajeFechaSalida = (String) (fIViaje.tDatosViajes.getValueAt(filaAEliminar, 1));
                String viajeVehiculoPlaca = (String) (fIViaje.tDatosViajes.getValueAt(filaAEliminar, 3));
                int rptaUsuario = JOptionPane.showConfirmDialog(null, "Desea ELIMINAR el registro con fecha de salida: " + viajeFechaSalida + " del vehículo:, " + viajeVehiculoPlaca + "?");
                if (rptaUsuario == 0) {
                    String rptaRegistro = viajeDAO.eliminarViaje(viajeId);
                    if (rptaRegistro != null) {
                        JOptionPane.showMessageDialog(null, rptaRegistro);
                    } else {
                        JOptionPane.showMessageDialog(null, "Registro Erroneo");
                    }
                }
                llenarTablaViajes(fIViaje.tDatosViajes);
                reiniciarContenedoresVj();
                habilitarControlesVj(false);
                fIViaje.getRootPane().setDefaultButton(fIViaje.btnNuevo);
            } else {
                JOptionPane.showMessageDialog(null, "Debe seleccionar un registro en la lista de Viajes.");
            }
        } catch (Exception ex) {
            System.out.println("Exception: " + ex.toString());
        }
    }
    if (e.getSource() == fIViaje.btnGuardar) {
        try {
            SimpleDateFormat formatoFecha = new SimpleDateFormat("yyyy-MM-dd");
            String fechaSalida = formatoFecha.format(fIViaje.dCViajeFechaSalida.getDate());
```

```
            fechaSalida = fechaSalida + " " +
fIViaje.spnrViajeFechaHorasSalida.getValue().toString() + ":" +
fIViaje.spnrViajeFechaMinutosSalida.getValue().toString() + ":00";
            fIViaje.txtFViajeFechaSalida.setText(fechaSalida);
            String fechaLlegada =
formatoFecha.format(fIViaje.dCViajeFechaLlegada.getDate());
            fechaLlegada = fechaLlegada + " " +
fIViaje.spnrViajeFechaHorasLlegada.getValue().toString() + ":" +
fIViaje.spnrViajeFechaMinutosLlegada.getValue().toString() + ":00";
            fIViaje.txtFViajeFechaLlegada.setText(fechaLlegada);
            String vehiculo_placa = (String)
fIViaje.cBViajeVehiculosDatos.getSelectedItem();
            int vehiculo_id =
vehiculoDAO.obtenerVehiculoId(vehiculo_placa);
            String ruta_descripcion = (String)
fIViaje.cBViajeRutas.getSelectedItem();
            int ruta_id = rutaDAO.obtenerRutaId(ruta_descripcion);
            Viaje viaje = new Viaje(fechaSalida, fechaLlegada, vehiculo_id,
ruta_id);
            if (estado) {
//              nuevo
              String rptaRegistro = viajeDAO.insertarViaje(viaje);
              if (rptaRegistro != null) {
                JOptionPane.showMessageDialog(null, rptaRegistro);
              } else {
                JOptionPane.showMessageDialog(null, "Registro de Viaje
Erroneo");
              }
            } else {
//              editar
              String id = fIViaje.txtViajeId.getText();
              viaje.setViaje_id(Integer.parseInt(fIViaje.txtViajeId.getText()));
              String rptaRegistro = viajeDAO.actualizarViaje(viaje);
              if (rptaRegistro != null) {
                JOptionPane.showMessageDialog(null, rptaRegistro);
              } else {
                JOptionPane.showMessageDialog(null, "Registro de Viaje
Erroneo");
              }
            }
            llenarTablaViajes(fIViaje.tDatosViajes);
            habilitarControlesVj(false);
            fIViaje.getRootPane().setDefaultButton(fIViaje.btnNuevo);
          } catch (Exception ex) {
            System.out.println("Exception: " + ex.toString());
          }
        }
        if (e.getSource() == fIViaje.btnCancelar) {
```

```
            habilitarControlesVj(false);
            reiniciarContenedoresVj();
            fIViaje.getRootPane().setDefaultButton(fIViaje.btnNuevo);
        }
        if (e.getSource() == fIViaje.btnFiltrarPlacaYFechas) {
            filtrarPorPlaca = true;
            filtrarPorConductor = false;
            llenarTablaViajes(fIViaje.tDatosViajes);
        }
        if (e.getSource() == fIViaje.btnFiltrarConductorYFechas) {
            filtrarPorPlaca = false;
            filtrarPorConductor = true;
            llenarTablaViajes(fIViaje.tDatosViajes);
        }
        if (e.getSource() == fIViaje.btnFiltrarFechas) {
            filtrarPorPlaca = false;
            filtrarPorConductor = false;
            llenarTablaViajes(fIViaje.tDatosViajes);
        }
        if (e.getSource() == fIViaje.btnAdministrarConductores) {
            int filaAdministrarConductores =
fIViaje.tDatosViajes.getSelectedRow();
            int filasSeleccionadas = fIViaje.tDatosViajes.getSelectedRowCount();
            if (filaAdministrarConductores >= 0 && filasSeleccionadas == 1) {
                administraConductores = true;
                habilitarControlesVj(true);
                llenarDatosViaje(filaAdministrarConductores);
                idViaje =
Integer.valueOf(String.valueOf(fIViaje.tDatosViajes.getValueAt(filaAdminist
rarConductores, 0)));
                llenarTablaConductores(fIViaje.tDatosConductoresExistentes, 0);
                llenarTablaConductores(fIViaje.tDatosConductoresAsignados,
idViaje);
            } else {
                JOptionPane.showMessageDialog(null, "Debe seleccionar un
registro en la lista de Viajes.");
            }
        }
        if (e.getSource() == fIViaje.btnAsignarConductor) {
            int filaConductorExistente =
fIViaje.tDatosConductoresExistentes.getSelectedRow();
            int filasSeleccionadas =
fIViaje.tDatosConductoresExistentes.getSelectedRowCount();
            if (filaConductorExistente >= 0 && filasSeleccionadas == 1) {

fIViaje.getRootPane().setDefaultButton(fIViaje.btnAsignarConductor);
```

```
            idConductor =
Integer.valueOf(String.valueOf(fIViaje.tDatosConductoresExistentes.getValu
eAt(filaConductorExistente, 0)));
            try {
               TrabajadorEnViaje trabajadorViaje = new
TrabajadorEnViaje(idViaje, idConductor);

trabajadorEnViajeDAO.insertarTrabajadorEnViaje(trabajadorViaje);
            } catch (Exception ex) {
               System.out.println("Exception: " + ex.toString());
            }
            llenarTablaConductores(fIViaje.tDatosConductoresAsignados,
idViaje);
         } else {
            JOptionPane.showMessageDialog(null, "Debe seleccionar un
registro en la lista de Conductores existentes.");
         }
      }
      if (e.getSource() == fIViaje.btnQuitarConductor) {
         int filaConductorAsignado =
fIViaje.tDatosConductoresAsignados.getSelectedRow();
         int filasSeleccionadas =
fIViaje.tDatosConductoresAsignados.getSelectedRowCount();
         if (filaConductorAsignado >= 0 && filasSeleccionadas == 1) {

fIViaje.getRootPane().setDefaultButton(fIViaje.btnQuitarConductor);
            idConductor =
Integer.valueOf(String.valueOf(fIViaje.tDatosConductoresAsignados.getValu
eAt(filaConductorAsignado, 0)));
            try {
               trabajadorEnViajeDAO.eliminarTrabajadorEnViaje(idViaje,
idConductor);
            } catch (Exception ex) {
               System.out.println("Exception: " + ex.toString());
            }
            llenarTablaConductores(fIViaje.tDatosConductoresAsignados,
idViaje);
         } else {
            JOptionPane.showMessageDialog(null, "Debe seleccionar un
registro en la lista de Conductores asignados.");
         }
      }
      if (e.getSource() == fIViaje.btnAdministrarViajes) {
         reiniciarContenedoresVj();
         habilitarControlesVj(false);
         fIViaje.getRootPane().setDefaultButton(fIViaje.btnNuevo);
         llenarTablaConductores(fIViaje.tDatosConductoresAsignados, -1);
         administraConductores = false;
```

```
        }
        if (e.getSource() == fIViaje.btnEvaluarDesplazamiento) {
            try {
                int filaElegida = fIViaje.tDatosViajes.getSelectedRow();
                int numFS = fIViaje.tDatosViajes.getSelectedRowCount();
                if (filaElegida >= 0 && numFS == 1) {
                    evaluando = true;
                    habilitarControlesVj(true);
                    fIViaje.getRootPane().setDefaultButton(fIViaje.btnGuardar);
                    estado = false;
                    llenarDatosViaje(filaElegida);
                    evaluando = false;
                    int idViaje =
Integer.parseInt(String.valueOf(fIViaje.tDatosViajes.getValueAt(filaElegida,
0)));
                    String fechaSalida =
String.valueOf(fIViaje.tDatosViajes.getValueAt(filaElegida, 1));
                    String fechaLlegada =
String.valueOf(fIViaje.tDatosViajes.getValueAt(filaElegida, 2));
                    int idVehiculo =
vehiculoDAO.obtenerVehiculoId(String.valueOf(fIViaje.tDatosViajes.getVal
ueAt(filaElegida, 3)));
                    int idRuta =
rutaDAO.obtenerRutaId(String.valueOf(fIViaje.tDatosViajes.getValueAt(fila
Elegida, 4)));
                    int numTransmisiones =
transmisionDAO.contarTransmisiones(0, idVehiculo, fechaSalida,
fechaLlegada, "");
                    Object[][] transmisionesV =
transmisionDAO.listarTransmisiones(numTransmisiones, 6, 0, idVehiculo,
fechaSalida, fechaLlegada, "");
                    int rptaUsuario = JOptionPane.showConfirmDialog(null, "Desea
evaluar el desplazamiento del viaje: " + idViaje + "?");
                    int contTransSinVecinos = 0;
                    for (int contTrans = 0; contTrans < numTransmisiones;
contTrans++) {
                        int faseProcesamiento =
Integer.parseInt(transmisionesV[contTrans][5].toString());
                        if (rptaUsuario == 0 || faseProcesamiento == 0) {
                            int idTransmision =
Integer.parseInt(transmisionesV[contTrans][0].toString());
                            String fechaTransmision =
transmisionesV[contTrans][1].toString();
                            double latitudTransmision =
Double.parseDouble(transmisionesV[contTrans][2].toString());
                            double longitudTransmision =
Double.parseDouble(transmisionesV[contTrans][3].toString());
```

```
                int velocidadTransmision = Integer.parseInt(transmisionesV[contTrans][4].toString());
                int numVecinos = 0;
                int[] perimetro = {3, 5, 8, 13, 21, 34, 55, 89, 144};
                int contPerimetro = numVecinos;
                double latitudI, latitudF, longitudI, longitudF;
                do {
                    latitudI = latitudTransmision - (0.001 * perimetro[contPerimetro] / 2);
                    latitudF = latitudTransmision + (0.001 * perimetro[contPerimetro] / 2);
                    longitudI = longitudTransmision - (0.001 * perimetro[contPerimetro] / 2);
                    longitudF = longitudTransmision + (0.001 * perimetro[contPerimetro] / 2);
                    numVecinos = geopuntoDAO.contarGeopuntosVecinos(latitudI, latitudF, longitudI, longitudF, idRuta);
                    contPerimetro++;
                    if (numVecinos > 0) {
                        System.out.println("nume veci " + numVecinos + " perimetro" + perimetro[contPerimetro]);
                    }
                } while (numVecinos == 0 && contPerimetro < perimetro.length - 1);
                Object[][] GeopuntosCandidatos = geopuntoDAO.listarGeopuntosVecinos(numVecinos, latitudI, latitudF, longitudI, longitudF, idRuta);
                double distEuclidea = 100;
                int idGeopunto = 0;
                for (int contVecinos = 0; contVecinos < numVecinos; contVecinos++) {
                    double latitudTmp = Double.parseDouble(GeopuntosCandidatos[contVecinos][1].toString());
                    double longitudTmp = Double.parseDouble(GeopuntosCandidatos[contVecinos][2].toString());
                    double rpta = calcularDistanciaEuclidea(latitudTransmision, longitudTransmision, latitudTmp, longitudTmp);
                    if (rpta < distEuclidea) {
                        distEuclidea = rpta;
                        idGeopunto = Integer.parseInt(GeopuntosCandidatos[contVecinos][0].toString());
                    }
                }
                if (idGeopunto != 0) {
                    Object[] GeopuntoElegido = geopuntoDAO.cargarGeopuntoElegido(idGeopunto);
```

```
                        evaluarDesplazamientoEnTransmision(idTransmision,
velocidadTransmision, GeopuntoElegido, idViaje);
                    } else {
                        contTransSinVecinos++;
                        System.out.println("=*=*=*= No se encontro un geopunto
cercano para la Transmision con id: " + idTransmision + " =*=*=*=");
                    }
                }
            }
            System.out.println("== num transmi: " + numTransmisiones + "
==");
            int                    numTransmEnViaje                    =
transmisionDAO.recuentoDeTransmisionesConOSinIncidentes(idViaje,
fechaSalida, fechaLlegada, false);
            System.out.println("==          num          transmiAsig:          "          +
numTransmEnViaje + " ==");
            int                    numIncidentes                    =
transmisionDAO.recuentoDeTransmisionesConOSinIncidentes(idViaje,
fechaSalida, fechaLlegada, true);
            System.out.println("== num transmiIncidase: " + numIncidentes
+ " ==");
            try {
                viajeDAO.actualizarInformacionEnViaje(idViaje,
numTransmEnViaje, numIncidentes);
            } catch (Exception ex) {
                System.out.println("Exception: " + ex.toString());
            }
            if (numTransmEnViaje < numTransmisiones) {
                JOptionPane.showMessageDialog(null, "Existen un total de "
+ contTransSinVecinos + " Transmisiones que no poseen Geopuntos muy
cercanos.");
            }
        } else {
            JOptionPane.showMessageDialog(null, "Debe seleccionar un
registro en la lista de Viajes.");
        }
    } catch (Exception ex) {
        System.out.println("Exception: " + ex.toString());
    }
}
if (e.getSource() == fIViaje.btnExportarViajeEvaluado) {
    try {
        int filaSeleccionada = fIViaje.tDatosViajes.getSelectedRow();
        int numFS = fIViaje.tDatosViajes.getSelectedRowCount();
        if (filaSeleccionada >= 0 && numFS == 1) {
            int                viajeId                =                (Integer)
(fIViaje.tDatosViajes.getValueAt(filaSeleccionada, 0));
```

```
                int idVehiculo = vehiculoDAO.obtenerVehiculoId(String.valueOf(fIViaje.tDatosViajes.getValueAt(filaSeleccionada, 3)));
                int idRuta = rutaDAO.obtenerRutaId(String.valueOf(fIViaje.tDatosViajes.getValueAt(filaSeleccionada, 4)));
                String rutaDescripcion = (String) (fIViaje.tDatosViajes.getValueAt(filaSeleccionada, 4));
                System.out.println("idVehiculo " + idVehiculo);
                int numTransmisiones = transmisionDAO.contarTransmisiones(0, idVehiculo, null, null, null);
                System.out.println("numTransmisiones " + numTransmisiones);
                System.out.println("viajeId " + viajeId);
                System.out.println("rutaDescripcion " + rutaDescripcion);
                System.out.println("rutaID " + idRuta);
                Object[][] transmisionesV = transmisionDAO.listarTransmisiones(numTransmisiones, 14, 0, viajeId, null, null, null);
                int rptaUsuario = JOptionPane.showConfirmDialog(null, "Desea EXPORTAR el viaje " + viajeId + " con ruta: " + rutaDescripcion + "?");
                if (rptaUsuario == 0) {
                    ExportarViaje exportarViaje = new ExportarViaje(viajeId, transmisionDAO);
                    exportarViaje.escribirTransmisionesEnViaje();
                }
                fIViaje.getRootPane().setDefaultButton(fIViaje.btnNuevo);
            } else {
                JOptionPane.showMessageDialog(null, "Debe seleccionar un registro en la lista de Viajes.");
            }
        } catch (Exception ex) {
            System.out.println("Exception: " + ex.toString());
        }
    }
}

    public double calcularDistanciaEuclidea(double latitudTransmision, double longitudTransmision, double latitudTmp, double longitudTmp) {
        double rpta = Math.sqrt(Math.pow((latitudTmp - latitudTransmision), 2) + Math.pow((longitudTmp - longitudTransmision), 2));
        return rpta;
    }

    public void evaluarDesplazamientoEnTransmision(int transmisionId, int transmisionVelocidad, Object[] objGeopunto, int viajeId) {
        int incidente = 0;
        int idGeopunto = Integer.parseInt(objGeopunto[0].toString());
        int limiteVelocidadG = Integer.parseInt(objGeopunto[4].toString());
```

```
        String tipoEspacioG = objGeopunto[5].toString();
        int excesoVelocidad = 0;
        String tipoVelocidad = "prudente";
        int estabilidad = 1;
        if(transmisionVelocidad>limiteVelocidadG){
           excesoVelocidad = 1;
           if(transmisionVelocidad > 45 && transmisionVelocidad<=90){
              tipoVelocidad = "normal rapida";
              if(transmisionVelocidad>60){
                 estabilidad =0;
                 if(tipoEspacioG.compareTo("recta")!=0){
                    incidente = 1;
                 }
              }
           }else if (transmisionVelocidad>90){
              tipoVelocidad = "excesiva";
              estabilidad =0;
              incidente =1;
           }
        }
//Si(exceso de velocidad=no)=>(clase=-)
//Si(exceso de velocidad=sí)  y (tipo de velocidad=prudente)=>(clase=-)
//Si(exceso de velocidad=sí)  y (tipo de velocidad=excesiva)=>(clase=+)
//Si(exceso de velocidad=sí)  y (tipo de velocidad=normal rápida)  y
(estabilidad=sí)=>(clase=-)
//Si(exceso de velocidad=sí)  y (tipo de velocidad=normal rápida)  y
(estabilidad=no)  y (tipo de espacio=recta)  =>(clase=-)
//Si(exceso de velocidad=sí)  y (tipo de velocidad=normal rápida)  y
(estabilidad=no)  y (tipo de espacio=curva)=>(clase=+)
//Si(exceso de velocidad=sí)  y (tipo de velocidad=normal rápida)  y
(estabilidad=no)  y (tipo de espacio=otro)  =>(clase=+)
        try {
           transmisionDAO.actualizarInformacionEnTransmision(transmisionId,
excesoVelocidad, tipoVelocidad, estabilidad, incidente, viajeId, idGeopunto);
        } catch (Exception ex) {
                 System.out.println("Exception: " + ex.toString());
        }
     }
}
```

ANEXO D

TABLAS Y PROCEDIMIENTOS ALMACENADOS

Las tablas y procedimientos almacenados se encuentrar contenidos en un script de la siguiente forma:

```
DROP SCHEMA IF EXISTS `monitoreo`;
CREATE SCHEMA IF NOT EXISTS `monitoreo` DEFAULT CHARACTER SET latin1 COLLATE latin1_spanish_ci;
USE `monitoreo`;

--
-- Base de datos: `monitoreo`
--

DELIMITER $$
--
-- Procedimientos
--
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actualizar_cargo`(IN `x_cargo_id` INT, IN `x_cargo_descripcion` VARCHAR(25))
BEGIN
update cargos set cargo_descripcion = x_cargo_descripcion
where cargo_id = x_cargo_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actualizar_geopunto`(IN `x_geopunto_id` INT, IN `x_geopunto_latitud` DECIMAL(9,6), IN `x_geopunto_longitud` DECIMAL(9,6), IN `x_geopunto_referencia` VARCHAR(125), IN `x_geopunto_limite_velocidad` INT, IN `x_geopunto_tipo_espacio` VARCHAR(75), IN `x_geopunto_diametro_perimetro` INT, IN `x_geopunto_zona_urbana` BOOLEAN)
BEGIN
update geopuntos set geopunto_latitud = x_geopunto_latitud, geopunto_longitud = x_geopunto_longitud, geopunto_referencia = x_geopunto_referencia, geopunto_limite_velocidad = x_geopunto_limite_velocidad, geopunto_tipo_espacio = x_geopunto_tipo_espacio, geopunto_diametro_perimetro = x_geopunto_diametro_perimetro, geopunto_zona_urbana = x_geopunto_zona_urbana
where geopunto_id = x_geopunto_id;
```

```
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actualizar_informacion_en_transmision`(IN `x_transmision_id` INT, IN `x_transmision_exceso_velocidad` BOOLEAN, IN `x_transmision_tipo_velocidad` VARCHAR(15), IN `x_transmision_dificultad_reaccion` BOOLEAN, IN `x_transmision_incidente` BOOLEAN, IN `x_viaje_id` INT, IN `x_geopunto_id` INT)
BEGIN
update transmisiones set transmision_exceso_velocidad = x_transmision_exceso_velocidad, transmision_tipo_velocidad = x_transmision_tipo_velocidad, transmision_dificultad_reaccion = x_transmision_dificultad_reaccion, transmision_incidente = x_transmision_incidente, viajes_viaje_id = x_viaje_id, geopuntos_geopunto_id = x_geopunto_id, transmision_evaluada = 1 where transmision_id = x_transmision_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actualizar_informacion_en_viaje`(IN `x_viaje_id` INT, IN `x_viaje_nro_transmisiones` INT, IN `x_viaje_nro_incidentes` INT)
BEGIN
update viajes set viaje_nro_transmisiones = x_viaje_nro_transmisiones, viaje_nro_incidentes = x_viaje_nro_incidentes where viaje_id = x_viaje_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actualizar_ruta`(IN `x_ruta_id` INT, IN `x_ruta_descripcion` VARCHAR(55))
BEGIN
update rutas set ruta_descripcion = x_ruta_descripcion
where ruta_id = x_ruta_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actualizar_trabajador`(IN `x_trabajador_id` INT, IN `x_trabajador_dni` VARCHAR(8), IN `x_trabajador_nombres` VARCHAR(145), IN `x_trabajador_apellidos` VARCHAR(145), IN `x_trabajador_abreviacion` VARCHAR(25), IN `x_trabajador_email` VARCHAR(45), IN `x_trabajador_telefono` VARCHAR(30), IN `x_cargo_id` INT)
BEGIN
update trabajadores set trabajador_dni = x_trabajador_dni, trabajador_nombres = x_trabajador_nombres, trabajador_apellidos = x_trabajador_apellidos, trabajador_abreviacion = x_trabajador_abreviacion, trabajador_email = x_trabajador_email, trabajador_telefono = x_trabajador_telefono, cargos_cargo_id = x_cargo_id
where trabajador_id = x_trabajador_id;
```

```
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actualizar_vehiculo`(IN `x_vehiculo_id` INT, IN `x_vehiculo_placa` VARCHAR(15), IN `x_vehiculo_marca` VARCHAR(25), IN `x_vehiculo_modelo` VARCHAR(35), IN `x_vehiculo_anio_fabrica` INT, IN `x_vehiculo_nro_gps` VARCHAR(45), IN `x_vehiculo_nro_celular` VARCHAR(15), IN `x_vehiculo_fecha_registro` TIMESTAMP)
BEGIN
update vehiculos set vehiculo_placa = x_vehiculo_placa, vehiculo_marca = x_vehiculo_marca, vehiculo_modelo = x_vehiculo_modelo, vehiculo_anio_fabrica = x_vehiculo_anio_fabrica, vehiculo_nro_gps = x_vehiculo_nro_gps, vehiculo_nro_celular = x_vehiculo_nro_celular, vehiculo_fecha_registro = x_vehiculo_fecha_registro
where vehiculo_id = x_vehiculo_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actualizar_viaje`(IN `x_viaje_id` INT, IN `x_viaje_fecha_hora_salida` TIMESTAMP, IN `x_viaje_fecha_hora_llegada` TIMESTAMP, IN `x_vehiculo_id` INT, IN `x_ruta_id` INT)
BEGIN
update viajes set viaje_fecha_hora_salida = x_viaje_fecha_hora_salida, viaje_fecha_hora_llegada = x_viaje_fecha_hora_llegada, vehiculos_vehiculo_id = x_vehiculo_id, rutas_ruta_id = x_ruta_id where viaje_id = x_viaje_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_eliminar_cargo`(IN `x_cargo_id` INT)
BEGIN
delete from cargos where cargo_id = x_cargo_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_eliminar_geopunto`(IN `x_geopunto_id` INT)
BEGIN
delete from geopuntos where geopunto_id = x_geopunto_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_eliminar_geopunto_en_ruta`(IN `x_ruta_id` INT, IN `x_geopunto_id` INT)
BEGIN
```

```
delete from geopuntos_en_rutas where rutas_ruta_id = x_ruta_id AND
geopuntos_geopunto_id = x_geopunto_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_eliminar_ruta`(IN
`x_ruta_id` INT)
BEGIN
delete from rutas where ruta_id = x_ruta_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_eliminar_trabajador`(IN
`x_trabajador_id` INT)
BEGIN
delete from trabajadores where trabajador_id = x_trabajador_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE
`sp_eliminar_trabajador_en_viaje`(IN `x_viaje_id` INT, IN `x_trabajador_id` INT)
BEGIN
delete from trabajadores_en_viajes where viajes_viaje_id = x_viaje_id AND
trabajadores_trabajador_id = x_trabajador_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_eliminar_vehiculo`(IN
`x_vehiculo_id` INT)
BEGIN
delete from vehiculos where vehiculo_id = x_vehiculo_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_eliminar_viaje`(IN
`x_viaje_id` INT)
BEGIN
delete from viajes where viaje_id = x_viaje_id;
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_cargo`(IN
`x_cargo_descripcion` VARCHAR(25))
BEGIN
  insert into cargos(cargo_descripcion) values(x_cargo_descripcion);
    commit;
END$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_geopunto`(IN `x_geopunto_latitud` DECIMAL(9,6), IN `x_geopunto_longitud` DECIMAL(9,6), IN `x_geopunto_referencia` VARCHAR(125), IN `x_geopunto_limite_velocidad` INT, IN `x_geopunto_tipo_espacio` VARCHAR(75), IN `x_geopunto_diametro_perimetro` INT, IN `x_geopunto_zona_urbana` BOOLEAN)
BEGIN
 insert into geopuntos(geopunto_latitud, geopunto_longitud, geopunto_referencia, geopunto_limite_velocidad, geopunto_tipo_espacio, geopunto_diametro_perimetro, geopunto_zona_urbana)
 values(x_geopunto_latitud, x_geopunto_longitud, x_geopunto_referencia, x_geopunto_limite_velocidad, x_geopunto_tipo_espacio, x_geopunto_diametro_perimetro, x_geopunto_zona_urbana);
 commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_geopunto_en_ruta`(IN `x_ruta_id` INT, IN `x_geopunto_id` INT)
BEGIN
insert into geopuntos_en_rutas(rutas_ruta_id, geopuntos_geopunto_id) values(x_ruta_id, x_geopunto_id);
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_ruta`(IN `x_ruta_descripcion` VARCHAR(55))
BEGIN
 insert into rutas(ruta_descripcion) values(x_ruta_descripcion);
  commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_trabajador`(IN `x_trabajador_dni` VARCHAR(8), IN `x_trabajador_nombres` VARCHAR(145), IN `x_trabajador_apellidos` VARCHAR(145), IN `x_trabajador_abreviacion` VARCHAR(25), IN `x_trabajador_email` VARCHAR(45), IN `x_trabajador_telefono` VARCHAR(30), IN `x_cargo_id` INT)
BEGIN
 insert into trabajadores(trabajador_dni, trabajador_nombres, trabajador_apellidos, trabajador_abreviacion, trabajador_email, trabajador_telefono, cargos_cargo_id)
 values(x_trabajador_dni, x_trabajador_nombres, x_trabajador_apellidos, x_trabajador_abreviacion, x_trabajador_email, x_trabajador_telefono, x_cargo_id);
 commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_trabajador_en_viaje`(IN `x_viaje_id` INT, IN `x_trabajador_id` INT)
BEGIN
insert into trabajadores_en_viajes(viajes_viaje_id, trabajadores_trabajador_id) values(x_viaje_id, x_trabajador_id);
```

```
commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_transmision`(IN `x_transmision_fecha_hora` TIMESTAMP, IN `x_transmision_latitud` DECIMAL(9,6), IN `x_transmision_longitud` DECIMAL(9,6), IN `x_transmision_velocidad` DECIMAL(5,2), IN `x_transmision_odometro` DECIMAL(9,2), IN `x_transmision_referencia` VARCHAR(125), IN `x_transmision_codificada` VARCHAR(15), IN `x_vehiculo_id` INT)
BEGIN
insert into transmisiones(transmision_fecha_hora, transmision_latitud, transmision_longitud, transmision_velocidad, transmision_odometro, transmision_referencia, transmision_codificada, vehiculos_vehiculo_id)
 values(x_transmision_fecha_hora, x_transmision_latitud, x_transmision_longitud, x_transmision_velocidad, x_transmision_odometro, x_transmision_referencia, x_transmision_codificada, x_vehiculo_id);
 commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_vehiculo`(IN `x_vehiculo_placa` VARCHAR(15), IN `x_vehiculo_marca` VARCHAR(25), IN `x_vehiculo_modelo` VARCHAR(35), IN `x_vehiculo_anio_fabrica` INT, IN `x_vehiculo_nro_gps` VARCHAR(45), IN `x_vehiculo_nro_celular` VARCHAR(15), IN `x_vehiculo_fecha_registro` TIMESTAMP)
BEGIN
 insert into vehiculos(vehiculo_placa, vehiculo_marca, vehiculo_modelo, vehiculo_anio_fabrica, vehiculo_nro_gps, vehiculo_nro_celular, vehiculo_fecha_registro)
 values(x_vehiculo_placa, x_vehiculo_marca, x_vehiculo_modelo, x_vehiculo_anio_fabrica, x_vehiculo_nro_gps, x_vehiculo_nro_celular, x_vehiculo_fecha_registro);
 commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insertar_viaje`(IN `x_viaje_fecha_hora_salida` TIMESTAMP, IN `x_viaje_fecha_hora_llegada` TIMESTAMP, IN `x_ruta_id` INT, IN `x_vehiculo_id` INT)
BEGIN
 insert into viajes(viaje_fecha_hora_salida, viaje_fecha_hora_llegada, rutas_ruta_id, vehiculos_vehiculo_id)
 values(x_viaje_fecha_hora_salida, x_viaje_fecha_hora_llegada, x_ruta_id, x_vehiculo_id);
 commit;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_obtener_cargo_id`(IN `x_cargo_descripcion` VARCHAR(25), OUT `x_cargo_id` INT)
BEGIN
```

```
SELECT cargo_id INTO x_cargo_id FROM cargos WHERE cargo_descripcion LIKE
x_cargo_descripcion;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_obtener_geopunto_id`(IN
`x_geopunto_latitud` DECIMAL(9,6), IN `x_geopunto_longitud` DECIMAL(9,6),
OUT `x_geopunto_id` INT)
BEGIN
SELECT geopunto_id INTO x_geopunto_id FROM geopuntos WHERE
geopunto_latitud = x_geopunto_latitud AND geopunto_longitud =
x_geopunto_longitud;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_obtener_ruta_id`(IN
`x_ruta_descripcion` VARCHAR(55), OUT `x_ruta_id` INT)
BEGIN
SELECT ruta_id INTO x_ruta_id FROM rutas WHERE ruta_descripcion LIKE
x_ruta_descripcion;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_obtener_trabajador_id`(IN
`x_trabajador_abreviacion` VARCHAR(25), OUT `x_trabajador_id` INT)
BEGIN
SELECT trabajador_id INTO x_trabajador_id FROM trabajadores WHERE
trabajador_abreviacion LIKE x_trabajador_abreviacion;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_obtener_vehiculo_id`(IN
`x_vehiculo_placa` VARCHAR(15), OUT `x_vehiculo_id` INT)
BEGIN
SELECT vehiculo_id INTO x_vehiculo_id FROM vehiculos WHERE vehiculo_placa
LIKE x_vehiculo_placa;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE
`sp_validar_credenciales_usuario`(IN `x_usuario_nombre` VARCHAR(45), IN
`x_usuario_contrasenia` VARCHAR(45), OUT `x_usuario_id` INT)
BEGIN
SELECT usuario_id INTO x_usuario_id FROM usuarios WHERE
STRCMP(usuario_nombre, x_usuario_nombre)=0 AND STRCMP(usuario_contrasenia,
x_usuario_contrasenia)=0;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE
`sp_verificar_transmision_codificada`(IN `x_transmision_codificada` VARCHAR(15),
OUT `x_transmision_id` INT)
BEGIN
```

```
SELECT transmision_id INTO x_transmision_id FROM transmisiones WHERE STRCMP(TRIM(transmision_codificada), TRIM(x_transmision_codificada))=0;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_verificar_transmision_fecha_hora_vehiculo_id`(IN `x_transmision_fecha_hora` VARCHAR(21), IN `x_vehiculo_id` INT, OUT `x_transmision_id` INT)
BEGIN
SELECT transmision_id FROM transmisiones WHERE transmision_fecha_hora LIKE x_transmision_fecha_hora AND vehiculos_vehiculo_id = x_vehiculo_id;
END$$

DELIMITER ;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `cargos`
--

CREATE TABLE IF NOT EXISTS `cargos` (
  `cargo_id` int(11) NOT NULL AUTO_INCREMENT,
  `cargo_descripcion` varchar(25) COLLATE latin1_spanish_ci NOT NULL,
  PRIMARY KEY (`cargo_id`),
  UNIQUE KEY `cargo_descripcion_UNIQUE` (`cargo_descripcion`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci AUTO_INCREMENT=5 ;

--
-- Volcado de datos para la tabla `cargos`
--

INSERT INTO `cargos` (`cargo_id`, `cargo_descripcion`) VALUES
(1, 'Administrador'),
(2, 'Secretaria'),
(3, 'Conductor'),
(4, 'Operador');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `geopuntos`
--

CREATE TABLE IF NOT EXISTS `geopuntos` (
  `geopunto_id` int(11) NOT NULL AUTO_INCREMENT,
  `geopunto_latitud` decimal(9,6) NOT NULL,
  `geopunto_longitud` decimal(9,6) NOT NULL,
```

```
  `geopunto_referencia` varchar(125) COLLATE latin1_spanish_ci DEFAULT NULL,
  `geopunto_limite_velocidad` int(11) NOT NULL,
  `geopunto_tipo_espacio` varchar(75) COLLATE latin1_spanish_ci NOT NULL,
  `geopunto_diametro_perimetro` int(11) NOT NULL,
  `geopunto_zona_urbana` tinyint(1) NOT NULL,
  PRIMARY KEY (`geopunto_id`),
  UNIQUE KEY `geopunto_latitud_longitud_UNIQUE`
(`geopunto_latitud`,`geopunto_longitud`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `geopuntos_en_rutas`
--

DROP TABLE IF EXISTS `geopuntos_en_rutas` ;

CREATE TABLE IF NOT EXISTS `geopuntos_en_rutas` (
  `rutas_ruta_id` int(11) NOT NULL ,
  `geopuntos_geopunto_id` int(11) NOT NULL ,
  PRIMARY KEY (`rutas_ruta_id`, `geopuntos_geopunto_id`) ,
  KEY `fk_geopuntos_en_rutas_rutas1` (`rutas_ruta_id`) ,
  KEY `fk_geopuntos_en_rutas_geopuntos1` (`geopuntos_geopunto_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `infracciones`
--

CREATE TABLE IF NOT EXISTS `infracciones` (
  `infraccion_id` int(11) NOT NULL AUTO_INCREMENT,
  `infraccion_duracion_evento` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `infraccion_espacio_recorrido` varchar(7) COLLATE latin1_spanish_ci NOT NULL,
  `transmisiones_transmision_id` int(11) NOT NULL,
  `viajes_viaje_id` int(11) NOT NULL,
  PRIMARY KEY (`infraccion_id`),
  KEY `fk_infracciones_transmisiones1` (`transmisiones_transmision_id`),
  KEY `fk_infracciones_viajes1` (`viajes_viaje_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
```

```
-- Estructura de tabla para la tabla `rutas`
--

CREATE TABLE IF NOT EXISTS `rutas` (
  `ruta_id` int(11) NOT NULL AUTO_INCREMENT,
  `ruta_descripcion` varchar(55) COLLATE latin1_spanish_ci NOT NULL,
  PRIMARY KEY (`ruta_id`),
  UNIQUE KEY `ruta_descripcion_UNIQUE` (`ruta_descripcion`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
AUTO_INCREMENT=8 ;

--
-- Volcado de datos para la tabla `rutas`
--

INSERT INTO `rutas` (`ruta_id`, `ruta_descripcion`) VALUES
(1, 'Arequipa - Cusco'),
(2, 'Arequipa - Desaguadero'),
(3, 'Arequipa - Puno'),
(4, 'Desaguadero - Arequipa'),
(5, 'Desaguadero - Puno'),
(6, 'Puno - Arequipa'),
(7, 'Puno - Desaguadero');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `trabajadores`
--

CREATE TABLE IF NOT EXISTS `trabajadores` (
  `trabajador_id` int(11) NOT NULL AUTO_INCREMENT,
  `trabajador_dni` varchar(8) COLLATE latin1_spanish_ci NOT NULL,
  `trabajador_nombres` varchar(145) COLLATE latin1_spanish_ci NOT NULL,
  `trabajador_apellidos` varchar(145) COLLATE latin1_spanish_ci NOT NULL,
  `trabajador_abreviacion` varchar(25) COLLATE latin1_spanish_ci NOT NULL,
  `trabajador_email` varchar(45) COLLATE latin1_spanish_ci DEFAULT NULL,
  `trabajador_telefono` varchar(30) COLLATE latin1_spanish_ci DEFAULT NULL,
  `cargos_cargo_id` int(11) NOT NULL,
  PRIMARY KEY (`trabajador_id`),
  UNIQUE KEY `trabajador_dni_UNIQUE` (`trabajador_dni`),
  UNIQUE KEY `trabajador_abreviacion_UNIQUE` (`trabajador_abreviacion`),
  KEY `fk_trabajadores_cargos1` (`cargos_cargo_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
AUTO_INCREMENT=8 ;

--
-- Volcado de datos para la tabla `trabajadores`
```

```
--

INSERT INTO `trabajadores` (`trabajador_id`, `trabajador_dni`, `trabajador_nombres`, `trabajador_apellidos`, `trabajador_abreviacion`, `trabajador_email`, `trabajador_telefono`, `cargos_cargo_id`) VALUES
(1, '43108106', 'Miguel Angel', 'Mamani Zeballos', 'mamanimiguel01001', 'miguelmzeb@gmail.com', '988980055', 1),
(2, '46203107', 'Milagros Rocio', 'Cardenas Miranda', 'cardenasmilagros02001', 'cmilagros@gmail.com', '054279285', 2),
(3, '45362718', 'Juan Carlos', 'Perez Carpio', 'perezjuan04001', 'juanperez@gmail.com', '987654321', 4),
(4, '29531549', 'Antonio Flavio', 'Mendoza Quispe', 'mendozaantonio03001', 'mqaf@gmail.com', '959123456', 3),
(5, '29493245', 'Sergio Andre', 'Paredes Alvarez', 'paredessergio03002', 'sergiopa@gmail.com', '989842109', 3),
(6, '47334234', 'Jorge Sebastian', 'Manrrique Cazulo', 'manrriquejorge03003', 'jmanrriquec@gmail.com', '054452568', 3),
(7, '43256840', 'Santiago Eduardo', 'Manrique Palomino', 'manrriquesantiago03004', 'smanrique@gmail.com', '984234765', 3);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `trabajadores_en_viajes`
--

CREATE TABLE IF NOT EXISTS `trabajadores_en_viajes` (
  `viajes_viaje_id` int(11) NOT NULL,
  `trabajadores_trabajador_id` int(11) NOT NULL,
  PRIMARY KEY (`viajes_viaje_id`,`trabajadores_trabajador_id`),
  KEY `fk_trabajadores_en_viajes_viajes1` (`viajes_viaje_id`),
  KEY `fk_trabajadores_en_viajes_trabajadores1` (`trabajadores_trabajador_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;

--
-- Estructura de tabla para la tabla `transmisiones`
--

CREATE TABLE IF NOT EXISTS `transmisiones` (
  `transmision_id` int(11) NOT NULL AUTO_INCREMENT,
  `transmision_fecha_hora` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `transmision_latitud` decimal(9,6) NOT NULL,
  `transmision_longitud` decimal(9,6) NOT NULL,
  `transmision_velocidad` decimal(5,2) NOT NULL,
  `transmision_odometro` decimal(9,2) NOT NULL,
  `transmision_referencia` varchar(125) COLLATE latin1_spanish_ci NOT NULL,
  `transmision_evaluada` tinyint(1) NOT NULL DEFAULT '0',
```

```
  `transmision_exceso_velocidad` tinyint(1) DEFAULT NULL,
  `transmision_tipo_velocidad` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL,
  `transmision_dificultad_reaccion` tinyint(1) DEFAULT NULL,
  `transmision_incidente` tinyint(1) DEFAULT NULL,
  `transmision_codificada` varchar(15) NOT NULL,
  `geopuntos_geopunto_id` int(11) DEFAULT NULL,
  `vehiculos_vehiculo_id` int(11) NOT NULL,
  `viajes_viaje_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`transmision_id`),
  UNIQUE KEY `transmision_codificada_UNIQUE` (`transmision_codificada`),
  KEY `fk_transmisiones_geopuntos1` (`geopuntos_geopunto_id`),
  KEY `fk_transmisiones_vehiculos1` (`vehiculos_vehiculo_id`),
  KEY `fk_transmisiones_viajes1` (`viajes_viaje_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
AUTO_INCREMENT=1 ;


-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `usuarios`
--

CREATE TABLE IF NOT EXISTS `usuarios` (
  `usuario_id` int(11) NOT NULL AUTO_INCREMENT,
  `usuario_nombre` varchar(45) COLLATE latin1_spanish_ci NOT NULL,
  `usuario_contrasenia` varchar(45) COLLATE latin1_spanish_ci NOT NULL,
  `trabajadores_trabajador_id` int(11) NOT NULL,
  PRIMARY KEY (`usuario_id`),
  UNIQUE KEY `usuario_nombre_UNIQUE` (`usuario_nombre`),
  KEY `fk_usuarios_trabajadores1` (`trabajadores_trabajador_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
AUTO_INCREMENT=2 ;

--
-- Volcado de datos para la tabla `usuarios`
--

INSERT INTO `usuarios` (`usuario_id`, `usuario_nombre`, `usuario_contrasenia`,
`trabajadores_trabajador_id`) VALUES
(1, 'Administrador', 'ucsm', 1);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `vehiculos`
--
```

```
CREATE TABLE IF NOT EXISTS `vehiculos` (
  `vehiculo_id` int(11) NOT NULL AUTO_INCREMENT,
  `vehiculo_placa` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `vehiculo_marca` varchar(25) COLLATE latin1_spanish_ci NOT NULL,
  `vehiculo_modelo` varchar(35) COLLATE latin1_spanish_ci NOT NULL,
  `vehiculo_anio_fabrica` int(11) NOT NULL,
  `vehiculo_nro_gps` varchar(45) COLLATE latin1_spanish_ci NOT NULL,
  `vehiculo_nro_celular` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `vehiculo_fecha_registro` timestamp NOT NULL DEFAULT
CURRENT_TIMESTAMP,
  PRIMARY KEY (`vehiculo_id`),
  UNIQUE KEY `vehiculo_placa_UNIQUE` (`vehiculo_placa`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
AUTO_INCREMENT=5 ;

--
-- Volcado de datos para la tabla `vehiculos`
--

INSERT INTO `vehiculos` (`vehiculo_id`, `vehiculo_placa`, `vehiculo_marca`,
`vehiculo_modelo`, `vehiculo_anio_fabrica`, `vehiculo_nro_gps`,
`vehiculo_nro_celular`, `vehiculo_fecha_registro`) VALUES
(1, 'A2N961', 'SCANIA', ' K124IB6X2NB400', 2007, ' 1208890', ' 958456098', '2010-
05-24 22:09:00'),
(2, 'A6N960', 'SCANIA', ' K124IB6X2NB360', 2007, ' 475901', ' 972683109', '2010-05-
18 15:42:00'),
(3, 'V4B958', 'SCANIA', 'K124IB6X2NB380', 2008, '373123', '951508321', '2010-06-15
17:01:00'),
(4, 'B7Z961', 'SCANIA', 'K113', 1996, '359587018946 012', '957312210', '2010-06-22
12:27:00'),
(5, 'VAT956', 'VOLVO', 'B430R 6X2', 2016, '1364521', '345901000383251', '2016-08-
23 12:15:29');
-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `viajes`
--

CREATE TABLE IF NOT EXISTS `viajes` (
  `viaje_id` int(11) NOT NULL AUTO_INCREMENT,
  `viaje_fecha_hora_salida` timestamp NOT NULL DEFAULT
CURRENT_TIMESTAMP,
  `viaje_fecha_hora_llegada` timestamp NOT NULL DEFAULT
CURRENT_TIMESTAMP,
  `vehiculos_vehiculo_id` int(11) NOT NULL,
  `rutas_ruta_id` int(11) NOT NULL,
  `viaje_nro_transmisiones` int(11) NOT NULL DEFAULT '0',
```

```
  `viaje_nro_incidentes` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`viaje_id`),
  UNIQUE KEY `viaje_fecha_hora_salida_vehiculo_id_UNIQUE`
(`viaje_fecha_hora_salida`,`vehiculos_vehiculo_id`),
  KEY `fk_viajes_vehiculos1` (`vehiculos_vehiculo_id`),
  KEY `fk_viajes_rutas1` (`rutas_ruta_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
AUTO_INCREMENT=35 ;

--
-- Volcado de datos para la tabla `viajes`
--

INSERT INTO `viajes` (`viaje_id`, `viaje_fecha_hora_salida`,
`viaje_fecha_hora_llegada`, `vehiculos_vehiculo_id`, `rutas_ruta_id`,
`viaje_nro_transmisiones`, `viaje_nro_incidentes`) VALUES
(1, '2015-12-23 23:37:00', '2015-12-24 08:30:00', 1, 4, 0, 0),
(2, '2015-12-24 15:06:00', '2015-12-24 21:22:00', 1, 3, 0, 0),
(3, '2015-12-25 16:39:00', '2015-12-25 22:18:00', 1, 6, 0, 0),
(4, '2015-12-26 14:27:00', '2015-12-26 21:01:00', 1, 3, 0, 0),
(5, '2015-12-27 14:40:00', '2015-12-27 23:30:00', 1, 6, 0, 0),
(6, '2015-12-28 02:01:00', '2015-12-28 10:11:00', 1, 2, 0, 0),
(7, '2015-12-28 23:35:00', '2015-12-29 08:40:00', 1, 4, 0, 0),
(8, '2015-12-29 14:34:00', '2015-12-29 21:14:00', 1, 3, 0, 0),
(9, '2015-12-30 14:37:00', '2015-12-30 23:43:00', 1, 6, 0, 0),
(10, '2015-12-23 14:42:00', '2015-12-23 20:47:00', 2, 6, 0, 0),
(11, '2015-12-24 01:56:00', '2015-12-24 10:31:00', 2, 2, 0, 0),
(12, '2015-12-24 14:06:00', '2015-12-24 22:53:00', 2, 4, 0, 0),
(13, '2015-12-25 14:06:00', '2015-12-25 20:27:00', 2, 3, 0, 0),
(14, '2015-12-26 14:35:00', '2015-12-26 20:24:00', 2, 6, 0, 0),
(15, '2015-12-27 01:56:00', '2015-12-27 09:53:00', 2, 2, 0, 0),
(16, '2015-12-27 23:45:00', '2015-12-28 07:44:00', 2, 4, 0, 0),
(17, '2015-12-28 14:10:00', '2015-12-28 20:34:00', 2, 3, 0, 0),
(18, '2015-12-29 14:32:00', '2015-12-29 20:14:00', 2, 6, 0, 0),
(19, '2015-12-30 01:44:00', '2015-12-30 10:19:00', 2, 2, 0, 0),
(20, '2015-12-30 23:39:00', '2015-12-31 07:42:00', 2, 4, 0, 0),
(21, '2015-12-31 14:38:00', '2015-12-31 20:35:00', 2, 3, 0, 0),
(22, '2015-12-31 21:45:00', '2016-01-01 03:17:00', 2, 6, 0, 0),
(23, '2015-12-23 14:31:00', '2015-12-23 21:25:00', 4, 3, 0, 0),
(24, '2015-12-24 03:03:00', '2015-12-24 09:11:00', 4, 6, 0, 0),
(25, '2015-12-24 13:04:00', '2015-12-24 19:41:00', 4, 3, 0, 0),
(26, '2015-12-25 14:44:00', '2015-12-25 20:40:00', 4, 6, 0, 0),
(27, '2015-12-26 02:05:00', '2015-12-26 10:23:00', 4, 2, 0, 0),
(28, '2015-12-26 23:38:00', '2015-12-27 08:03:00', 4, 4, 0, 0),
(29, '2015-12-27 14:07:00', '2015-12-27 20:38:00', 4, 3, 0, 0),
(30, '2015-12-28 14:34:00', '2015-12-28 20:30:00', 4, 6, 0, 0),
(31, '2015-12-29 01:52:00', '2015-12-29 09:52:00', 4, 2, 0, 0),
(32, '2015-12-29 23:38:00', '2015-12-30 08:18:00', 4, 4, 0, 0),
```

```
(33, '2015-12-30 14:28:00', '2015-12-30 21:08:00', 4, 3, 0, 0),
(34, '2015-12-31 14:39:00', '2015-12-31 20:21:00', 4, 6, 0, 0);

-- --------------------------------------------------------

--
-- Restricciones para tablas volcadas
--

--
-- Filtros para la tabla `geopuntos_en_rutas`
--
ALTER TABLE `geopuntos_en_rutas`
  ADD CONSTRAINT `fk_geopuntos_en_rutas_rutas1` FOREIGN KEY (`rutas_ruta_id`) REFERENCES `rutas` (`ruta_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `fk_geopuntos_en_rutas_geopuntos1` FOREIGN KEY (`geopuntos_geopunto_id`) REFERENCES `geopuntos` (`geopunto_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Filtros para la tabla `infracciones`
--
ALTER TABLE `infracciones`
  ADD CONSTRAINT `fk_infracciones_transmisiones1` FOREIGN KEY (`transmisiones_transmision_id`) REFERENCES `transmisiones` (`transmision_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `fk_infracciones_viajes1` FOREIGN KEY (`viajes_viaje_id`) REFERENCES `viajes` (`viaje_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Filtros para la tabla `trabajadores`
--
ALTER TABLE `trabajadores`
  ADD CONSTRAINT `fk_trabajadores_cargos1` FOREIGN KEY (`cargos_cargo_id`) REFERENCES `cargos` (`cargo_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Filtros para la tabla `transmisiones`
--
ALTER TABLE `transmisiones`
  ADD CONSTRAINT `fk_transmisiones_geopuntos1` FOREIGN KEY (`geopuntos_geopunto_id`) REFERENCES `geopuntos` (`geopunto_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
```

```
  ADD CONSTRAINT `fk_transmisiones_vehiculos1` FOREIGN KEY (`vehiculos_vehiculo_id`) REFERENCES `vehiculos` (`vehiculo_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `fk_transmisiones_viajes1` FOREIGN KEY (`viajes_viaje_id`) REFERENCES `viajes` (`viaje_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Filtros para la tabla `usuarios`
--
ALTER TABLE `usuarios`
  ADD CONSTRAINT `fk_usuarios_trabajadores1` FOREIGN KEY (`trabajadores_trabajador_id`) REFERENCES `trabajadores` (`trabajador_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Filtros para la tabla `viajes`
--
ALTER TABLE `viajes`
  ADD CONSTRAINT `fk_viajes_rutas1` FOREIGN KEY (`rutas_ruta_id`) REFERENCES `rutas` (`ruta_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `fk_viajes_vehiculos1` FOREIGN KEY (`vehiculos_vehiculo_id`) REFERENCES `vehiculos` (`vehiculo_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Filtros para la tabla `trabajadores_en_viajes`
--
ALTER TABLE `trabajadores_en_viajes`
  ADD CONSTRAINT `fk_trabajadores_en_viajes_viajes1` FOREIGN KEY (`viajes_viaje_id`) REFERENCES `viajes` (`viaje_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `fk_trabajadores_en_viajes_trabajadores1` FOREIGN KEY (`trabajadores_trabajador_id`) REFERENCES `trabajadores` (`trabajador_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

ANEXO E

CUESTIONARIO

Previo un atento saludo, le invito a responder el presente cuestionario relacionado con el procedimiento de monitoreo satelital de vehículos. Sus impresiones y opiniones servirán para respaldar la evaluación del procedimiento de monitoreo satelital de vehículos empleado en la actualidad.

Marque con una X su respuesta.

1. ¿Cómo calificaría el procedimiento de monitoreo satelital de vehículos vigente requerido por la SUTRAN en los terminales terrestres?

Bueno		Regular		Malo	

2. La detección de regiones críticas en una ruta de transporte, ¿Nos puede permitir entender la magnitud real de dichos incidentes registrados?

Sí		No	

3. Analizar el desplazamiento vehicular registrado según el porcentaje de regiones críticas o incidentes ocultos detectados es:

Necesario		Recomendable		Innecesario	

4. En base al porcentaje de éxito de clasificación de una nueva transmisión en ruta, ¿Cuál es el nivel de efectividad del sistema propuesto teniendo en cuenta la nueva información obtenida (e.g., regiones críticas en zonas urbanas)?

Muy bueno		Aceptable		Malo	

5. El nivel de integridad de la información contenida en el vector de datos o conjunto de atributos de un registro de transmisión del desplazamiento vehicular monitoreado con el sistema vigente, ¿Es aceptable?

Sí		No	

6. ¿Considera que se necesita agregar nuevos atributos en los registros de transmisión de desplazamiento vehicular, para interpretar mejor los registros de monitoreo satelital de vehículos?

Sí		No	

7. La reevaluación y clasificación de las transmisiones históricas a través del modelo diseñado para detectar las regiones críticas o incidentes ocultos es:

Necesario		Recomendable		Innecesario	

8. Según la información obtenida con el sistema propuesto, ¿Cuál es el nivel de aceptación del modelo de clasificación de transmisiones?

Alto		Regular		Bajo	

9. En el sistema de monitoreo vehicular propuesto, ¿Cómo considera la funcionalidad que permite obtener la información resumen de los viajes registrados?

Ideal		Recomendable		Innecesaria	

10. Como parte de la calidad del servicio de transporte brindado, monitorear los límites máximos permisibles en zonas urbanas, lugares con señalización restringida y curvas peligrosas es:

Necesario		Recomendable		Innecesario	

11. ¿Cuál es el nivel de aceptación de los sistemas de monitoreo satelital de vehículos utilizados en los terminales terrestres?

Alto		Regular		Bajo	

12. ¿Cómo considera el nuevo flujo propuesto para revisar el desplazamiento histórico de un vehículo a nivel de viajes?

Adecuado		Se puede mejorar		Inadecuado	

13. El flujo de trabajo del sistema propuesto en comparación con los sistemas utilizados en los terminales terrestres, ¿Ofrece un mejor control de las transmisiones del desplazamiento vehicular?

Sí		No	

14. ¿Considera que el sistema propuesto puede servir para la toma de decisiones relacionada con el servicio de transporte terrestre?

Sí		No	