UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN
ESPECIALIDAD SISTEMAS ELECTRÓNICOS

"Desarrollo de una solución para la gestión de terminales de consumos propios mediante herramientas multiplataforma open source"

TRABAJO FINAL DE CARRERA

Autor/es:

Manuel Juan Cebrián Ferriols

Director/es:

D. Francisco Sales Castells Ramón

GANDIA, 2011

Dedicado a mis hermanos

Índice

Listado de figuras

Capítulo 1: Introducción y objetivos

El presente proyecto se desarrolla como parte de la realización de prácticas de empresa en la empresa Electroredeval Sistemas en colaboración con la Universidad Politecnica de Valencia. Siendo mi tutor en la empresa D. Jose Antonio Pérez Sánchez y mi tutor en la universidad D. Francisco Sales Castells Ramón, a los cuales he de agradecer la oportunidad de haberme permitido realizar practicas de empresa mientras cursaba mis estudios.

El proyecto escogido para este trabajo de final de carrera posee una parte de desarrollo en hardware y varia partes en software. Debido a que he desarrollado el software de gestión, tome la decisión de utilizar el desarrollo de este software como trabajo de final de carrera.

1.1 Introducción

Cuando una empresa que posee su propia flota de automóviles alcanza un número considerable de vehículos, aparece la posibilidad de que la empresa monte en sus propias instalaciones una estación de repostaje (gasolinera). Esto permite reducir gastos y un mayor control en las operaciones de repostaje de la empresa. Esta estación de repostaje privada recibe la denominación de terminal de consumos propios. Estas terminales también pueden ser utilizadas por empresas dedicadas a gestionar terminales de consumo propio para otras empresas, de forma que funcionarían según el esquema clásico de una gasolinera, pero siendo sus clientes las empresas que previamente han acordado la utilización de estos servicios.

Una terminal de consumos propios está compuesto de uno o varios tanques de combustible conectados a uno o varios surtidores y cada surtidor suele poseer de una a varias mangueras de combustible. Estos surtidores funcionan a través de tarjetas inteligentes para la identificación de conductores y/o vehículos, realizando consultas sobre la autorización de los conductores y vehículos sobre una base de datos. Además, todas las operaciones que se realicen en el terminal de consumos propios deben ser registradas en un base de datos.

Este proyecto diseña y desarrolla una base de datos y software que atacar a esa base de datos. Este software debe tener la funcionalidad de gestionar distintos terminales, tanques de combustible, operaciones en los tanques de combustibles, vehículos, clientes, conductores, operaciones de repostaje y productos (combustibles).

Hemos definido en el esquema de usuarios de nuestro software de gestión a cliente, conductores y vehículos. Esto puede llevar a la confusión por lo que es necesario una descripción mas detallada del concepto de cliente. Denominamos cliente a la empresa o persona física que contrata los servicios de repostaje en un terminal de consumos propios. Cada vehículo y conductor pertenece a un determinado cliente, siendo obvio que un vehículo o conductor solo pueda pertenecer a un único cliente.

La parte del desarrollo hardware de este proyecto consistirá en el diseño e implementación de toda la electrónica de control que posee un surtidor de combustible. Esta electrónica de control se encarga de la gestión de usuarios, control de las operaciones de respotaje y control de las mangueras. Así como la transmisión de todos los datos de las operaciones realizadas a un servidor que almacenara todos estos datos en una base de datos.

La parte software del proyecto consiste en varias partes. Una de las partes consiste en la creación de una base de datos para almacenar toda la información que necesitara el surtidor para su correcto funcionamiento, así como almacenar todas las operaciones. Los datos que se almacenaran en la base de datos serán discutidos con mayor detalle en los siguientes capítulos de esta memoria.

Otra de las partes del desarrollo software es la creación de una aplicación gráfica que permita al operador o administrador de estos surtidores, introducir los datos en la base de datos que necesitara el surtidor para su correcto funcionamiento, así como la consulta de las operaciones realizadas sobre cada uno surtidores. La funcionalidad de este aplicación de gestión se discutirá con mayor detalle en los siguientes capítulos.

Otra parte del desarrollo software consiste en la creación de un software que funcione como un servicio, en entornos Windows o un daemon, en entorno Unix, que se encargue de recibir los datos de los surtidores y almacenarlos en una base de datos. Este servicio y el surtidor se comunican mediante puertos de comunicaciones o mediante una conexión de red, a través de sockets sobre IP.

Aunque he desarrollado todas las partes software que componen este proyecto, solo he utilizado el desarrollo de la base de datos y el software de gestión como trabajo de final de carrera. El software que se ejecuta como un servicio contiene el protocolo que se utiliza para la comunicación de datos y por motivos de confidencialidad no he podido utilizarlo en este trabajo de final de carrera.

1.2 Software de código abierto

Debido a que en la fase de requerimientos el cliente solicitó que el software pudiera ser ejecutado y mantenido en distintas plataformas, además de que no fuera necesario el pago de licencias para las distintas herramientas y librerías de código que se utilizaran para el desarrollo del software, así como una sistema de gestión de base de datos que cumpliera los mismos requisitos.

Para cumplir los requisitos anteriores se decidió utilizar herramientas y librerías con licencias Open Source. Open Source (Código abierto) es el término con el que se conoce al software distribuido y desarrollado libremente. Siendo distinto del denominado software libre, ya que el software de código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

En nuestro proyecto se buscarán herramientas y librerías que posean licencias open source que permitan un uso comercial, así como la posibilidad de no entregar el código fuente de nuestro software.

1.3 Objetivos

El objetivo principal de este trabajo de final de carrera es el diseño e implementación de la base de datos y el software de gestión de la base de datos. Como requisitos del diseño de la base de datos se impuso que el diseño de las tablas fuera lo mas simple posible para evitar un excesivo mantenimiento y permitiera un sistema muy simple para realizar copias de seguridad de los datos contenidos en la base de datos. Además de que se solicito que la base de datos poseyera una excesiva redundancia en el almacenamiento de datos para evitar que un mal manejo del software de gestión por parte de un usuario final pudiera corromper datos críticos como el registro de servicios realizados.

Respecto al software de gestión se solicito que el usuario final pudiera hacer uso del software sin poseer conocimientos de bases de datos, así como que se diseñara con un aspecto visual moderno e intuitivo.

También fue impuesta la restricción en el uso de software privativo para evitar que pagar costes de las licencias, ni tener que realizar una entrega del código fuente desarrollado.

Estos objetivos pueden ser desglosados de la siguiente forma:

- Desarrollo de la base de datos
  - Diseño que permita bajo mantenimiento y un forma simple de realizar las copias de seguridad
  - Diseño redundante para evitar que un fallo cometido por el usuario final al introducir los datos pueda corromper el registro de operaciones.

- Desarrollo de el software de gestión
  - Diseño orientado a usuarios finales sin conocimientos de bases de datos
  - Interfaz visualmente atractiva e intuitiva.

Capítulo 2: Revisión del estado del arte

En este capitulo analizaremos las principales herramientas de código abierto, motor de la base de datos y librerías gráficas, que a priori puedan ser utilizadas en nuestro proyecto.

2.1 Motor de la base de datos

Los sistemas de gestión de bases de datos (en inglés database management system, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Los objetivos a cumplir por un DBMS son:

- Abstracción de la información.
  Los DBMS ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Por ejemplo, el número de archivos, así como su organización o jerarquía son transparentes para el usuario. Así, se definen varios niveles de abstracción.

- Independencia.
  La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- Consistencia.
  En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad determinada que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los DBMS existen herramientas que facilitan la programación de este tipo de condiciones.

- Seguridad.
  La información almacenada en una base de datos puede llegar a tener un gran valor. Los DBMS deben garantizar que esta información se encuentra segura de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

- Manejo de transacciones.
  Una transacción es un programa que se ejecuta como una sola operación. Esto quiere decir que el resultado de una ejecución en la que se produce una falla es el mismo que se obtendría si el programa no se hubiera ejecutado. Los DBMS proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

- Tiempo de respuesta.
  Lógicamente, es deseable minimizar el tiempo que el DBMS tarda en proporcionar la información solicitada y en almacenar los cambios realizados.

En este proyecto debemos buscar un DBMS con licencia open source que pueda ser ejecutado en varias plataformas. Después de realizar una búsqueda en internet, se ha llegado a la conclusión que los posibles candidatos para este proyecto son: MySQL, PostgreSQL y Firebird.

2.1.1 MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con varios millones de instalaciones. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios.

MySQL es muy utilizado en aplicaciones web, como Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Características de la última versión:

- Un amplio subconjunto de ANSI SQL 99, y varias extensiones.
- Soporte multiplataforma.
- Procedimientos almacenados
- Disparadores (triggers).
- Vistas actualizables.
- Soporte a VARCHAR
- Soporte X/Open XA de transacciones distribuidas; transacción en dos fases como parte de esto, utilizando el motor InnoDB de Oracle.
- Motores de almacenamiento independientes (MyISAM para lecturas rápidas, InnoDB para transacciones e integridad referencial).
- Transacciones con los motores de almacenamiento InnoDB, BDB Y Cluster; puntos de recuperación (savepoints) con InnoDB.
- Soporte para SSL.
- Query caching
- Sub-SELECTs (o SELECTs anidados).
- Réplica con un maestro por esclavo, varios esclavos por maestro, sin soporte automático para múltiples maestros por esclavo.
- indexing y búsqueda de campos de texto completos usando el motor de almacenamiento MyISAM.
- Soporte completo para Unicode.
- Conforme a las reglas ACID usando los motores InnoDB, BDB y Cluster.
- Shared-nothing clustering through MySQL Cluster.

MySQL funciona sobre múltiples plataformas, incluyendo: AIX, BSD, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, Novell Netware, OpenBSD, OS/2 Warp, QNX, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, eBD, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista, Windows 7 y Windows Server y OpenVMS.

La licencia GNU GPL de MySQL obliga a que la distribución de cualquier producto derivado (aplicación) se haga bajo esa misma licencia. Si un desarrollador desea incorporar MySQL en su producto pero desea distribuirlo bajo otra licencia que no sea la GNU GPL, puede adquirir una licencia comercial de MySQL que le permite hacer justamente eso.

Estos son algunos de los usuarios mas destacados que hacen uso de MySQL: Amazon.com, Cox Communications, Craigslist, CNET Networks, Digg, Flickr, Google, Joomla!, phpBB, LiveJournal, NASA, NetQOS, Nokia, Omniture, Sabre, Slashdot, Wikipedia, WordPress, Yahoo!.

2.1.2 PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

La licencia BSD (Berkeley Software Distribution) es una licencia de software libre permisiva. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de formas desinteresadas, altruistas, libres y/o apoyadas por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Algunas de sus principales características son, entre otras:

- Alta concurrencia
  Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último que poseía la tabla antes de que se ejecutara los comandos SQL. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

- Amplia variedad de tipos nativos
- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas).
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.
- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).
- Disparadores (triggers)

Un disparador o trigger se define como una acción específica que se realiza de acuerdo a un evento,

cuando éste ocurra dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

En general, cualquier plataforma moderna tipo Unix debe ser capaz de ejecutar PostgreSQL. PostgreSQL también corre de forma nativa en sistemas operativos basados en Microsoft Windows NT como Win2000 SP4, WinXP y Win2003.

Estos son algunos de los usuarios mas destacados que hacen uso de PostgreSQL: La American Chemical Society, BASF, IMDb, Skype, TiVo, Penny Arcade, Sony Online, U.S. Departamento de Trabajo, USPS, VeriSign, Pictiger.com, Wisconsin Circuit Court Access, OpenACS y IFE.

2.1.3 Firebird

Firebird es un sistema de administración de base de datos relacional de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en 2000. Su código fue reescrito de C a C++.

A finales de la década de 1990, Borland decidió liberar el código de Interbase. Diversos integrantes de la plantilla crearon una nueva empresa denominada IBPhoenix, y junto a otros desarrolladores independientes, crearon el fork ahora conocido como Firebird. Más tarde, Borland decidiría volver a privatizar Interbase y comercializar sus licencias. Sin embargo, Firebird sigue siendo un proyecto de código abierto bajo una licencia similar a la MPL (Mozilla Public License).

Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre. En este caso, Firebird puede enlazado y distribuido junto a una aplicación de código cerrado sin tener que pagar ninguna licencia ni deber licenciar la aplicación bajo una licencia de software libre.

Características:

- Es multiplataforma, y actualmente puede ejecutarse en los sistemas operativos:
- Ejecutable pequeño, con requerimientos de hardware bajos.
- Arquitectura Cliente/Servidor sobre protocolo TCP/IP y otros (embedded).
- Soporte de transacciones ACID y claves foráneas.
- Buena seguridad basada en usuarios/roles.
- Bases de datos de sólo lectura, para aplicaciones que corran desde dispositivos sin capacidad de escritura, como cd-roms.
- Existencia de controladores ODBC, OLEDB, JDBC, PHP, Perl, .net, etc.
- Requisitos de administración bajos, siendo considerada como una base de datos libre de mantenimiento, al margen de la realización de copias de seguridad.
- Pleno soporte del estándar SQL-92, tanto de sintaxis como de tipos de datos.
- Completo lenguaje para la escritura de disparadores y procedimientos almacenados denominado PSQL.
- Capacidad de almacenar elementos BLOB (Binary Large OBjects).
- Soporte de User-Defined Functions (UDFs).
- Versión autoejecutable, sin instalación, excelente para la creación de catálogos en CD-Rom y para crear versiones de evaluación de algunas aplicaciones.

Firebird funciona sobre múltiples plataformas como Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows.

2.2 Bibliotecas gráficas

Previamente a la definición de biblioteca gráfica debemos definir lo que es una biblioteca. Una biblioteca (del inglés library) es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código que implementa funciones y define estructuras de datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables.

La mayoría de los sistemas operativos modernos proporcionan bibliotecas que implementan la mayoría de los servicios del sistema. De esta manera, estos servicios se convierten en una "materia prima" que cualquier aplicación moderna espera que el sistema operativo ofrezca. Como tal, la mayor parte del código utilizado por las aplicaciones modernas se ofrece en estas bibliotecas.

Al referirnos a biblioteca gráfica, nos referimos a una biblioteca que contiene todo el código y funcionalidad para el desarrollo de interfaces gráficas de usuario. Las bibliotecas gráficas mas famosas en entornos de software libre son wxWidgets, QT ó GTK.

Aunque existen una gran cantidad de bibliotecas gráficas que cumplan los requisitos necesarios para nuestro proyecto, hemos decidido utilizar unas de las tres librerías mas populares que se han citado anteriormente. Esta decisión ha sido tomada debido a que el hecho de recibir mayor popularidad que otras bibliotecas gráficas ha propiciado que para estas librerías exista una mayor cantidad de documentación.

2.2.1 wxWidgets

Las wxWidgets son unas bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++. Están publicadas bajo una licencia LGPL, similar a la GPL con la excepción de que el código binario producido por el usuario a partir de ellas, puede ser propietario, permitiendo desarrollar aplicaciones empresariales sin coste de licencias.

Las wxWidgets proporcionan una interfaz gráfica basada en las bibliotecas ya existentes en el sistema (nativas), con lo que se integran de forma óptima y resultan muy portables entre distintos sistemas operativos. Están disponibles para Windows, MacOS, GTK+, Motif, OpenVMS y OS/2.

También pueden ser utilizadas desde otros lenguajes de programación, aparte del C++: Java, Javascript, Perl, Python, Smalltalk, Ruby.

Fue diseñado por Julian Smart en la universidad de Edinburgo 1992. Julian diseñaba la herramienta meta-CASE llamada Hardy que necesitaba correr en Windows, así como en estaciones de trabajo de X-Unix, las herramientas existentes y comerciales multiplataforma eran costosas para un proyecto experimental, así que su única alternativa era crear su propia herramienta. Inicialmente se llamaba wxWindows pero tuvo que cambiar al nombre por wxWidgets debido a que la empresa Microsoft interpuso una demanda a finales de 2003 por una posible confusión con el nombre de su sistema operativo.

wxWidgets (W para Windows y la X para X-Unix) es un Framework parecido a Microsoft Foundation Classes, especializado en el desarrollo de aplicaciones multiplataforma en lenguaje C++ aunque también existen bindings para Python y Perl, es multiplataforma, soporta Windows, Linux, Mac OS X , Unix y sus variantes, Solaris, Plataformas Embedded (inicios de investigación),

también en plataformas móviles como Microsoft Pocket PC, y Palm OS; se distribuye bajo licencia wxWindows License (compatible con Open Source y GNU LGPL (Lesser General Public License)) permitiendo utilizarla para desarrollos comerciales, siempre y cuando estos desarrollos no usen código distribuido bajo alguna licencia GNU.

Cuenta con una parte denominada wxBase que incluye clases como wxString, clases para el manejo de archivos y directorios de manera independiente del sistema, funcionalidades como: gráficos 2D, 3D con OpenGL, Bases de Datos (ODBC), Redes, Impresión, Hilos, visión e impresión del HTML, un sistema de archivos virtual y cuenta con algunos IDEs.

En la figura 1 se puede observar una captura del programa WebsitePainter, el cual utiliza las wxWidgets.

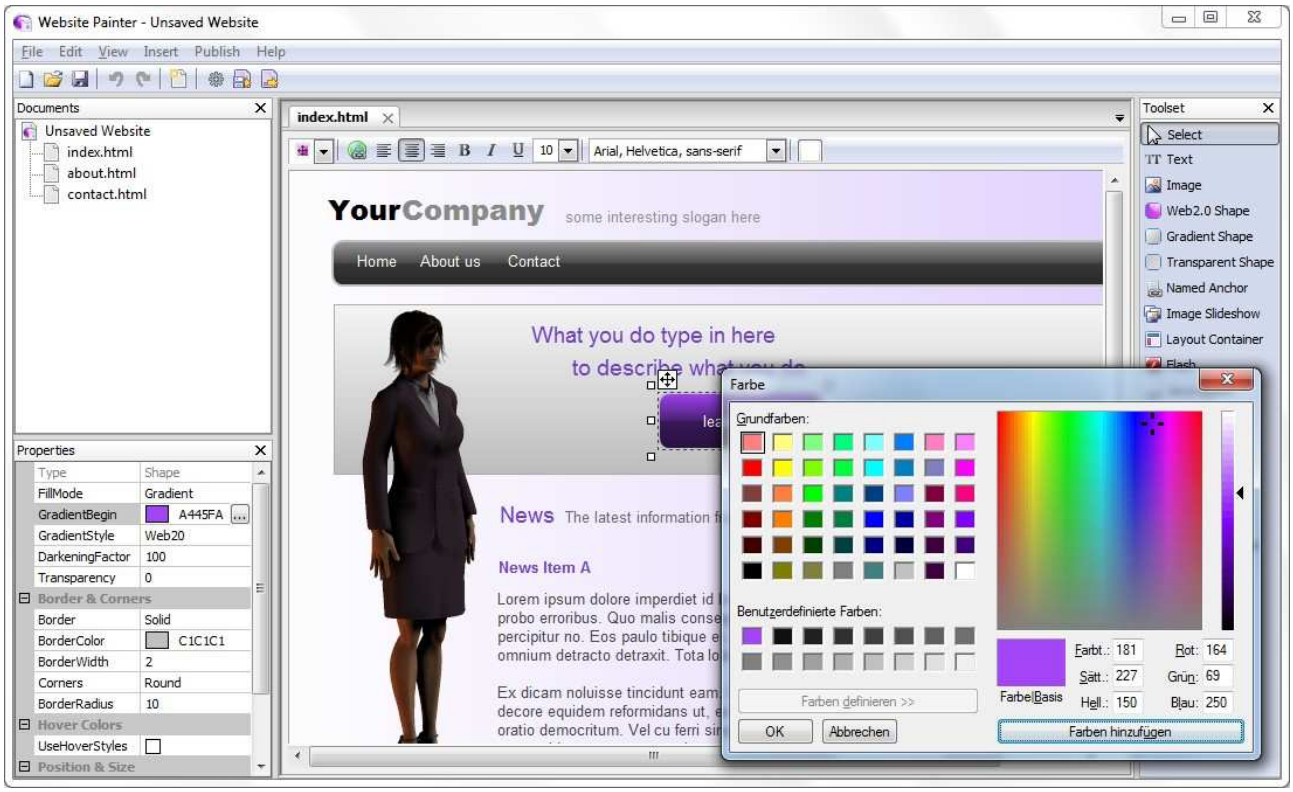

Figura 1. Captura de pantalla del programa WebsitePainter

2.2.2 GTK

GTK+ o The GIMP Toolkit es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario (GUI), principalmente para los entornos gráficos GNOME, XFCE y ROX aunque también se puede usar en el escritorio de Windows, MacOS y otros.

Inicialmente fueron creadas para desarrollar el programa de edición de imagen GIMP, sin embargo actualmente se usan bastante por muchos otros programas en los sistemas GNU/Linux. Junto a Qt es una de las bibliotecas más populares para X Window System.

GTK+ se ha diseñado para permitir programar con lenguajes como C, C++, C#, Java, Ruby, Perl, PHP o Python. Licenciado bajo los términos de LGPL, GTK+ es software libre y es parte del proyecto GNU.

GTK es un API orientado a objetos. Aunque está completamente escrita en C, soporta la idea de clases y funciones de respuesta (es decir punteros a funciones).

Existe un tercer componente llamado glib que proporciona un sustituto a algunas llamadas que podríamos denominar estándar. También incorpora funciones adicionales para manejar listas enlazadas, etc... Las funciones sustituto son usadas para aumentar la portabilidad de GTK, ya que algunas de las funciones incluidas no se encuentran disponibles en otros entornos Unix, como es el caso de g_strerror(). Otras simplemente son versiones mejoradas de las que proporciona libc. Por ejemplo g_malloc() proporciona métodos de depuración que no se encuentran la versión de libc.

En la figura 2 puede observarse una captura del programa GIMP, el cual hace uso de las librerías GTK.

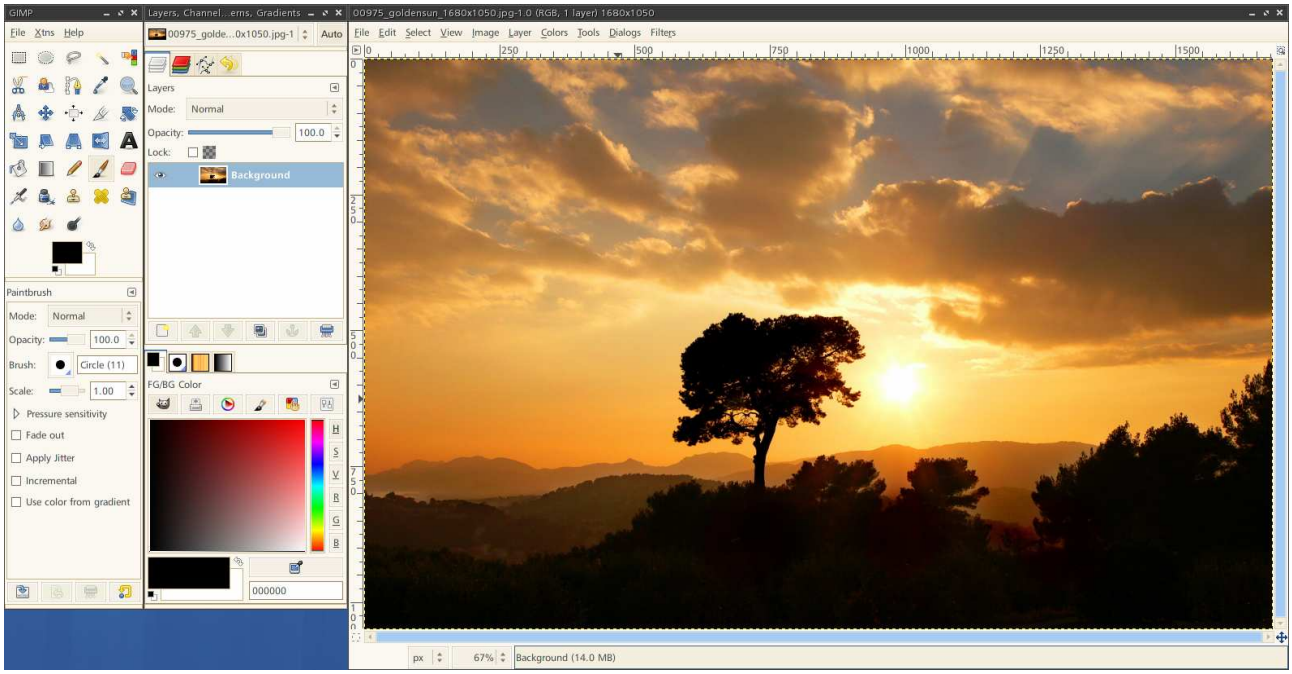

Figura 2. Captura de pantalla del programa GIMP

2.2.3 Qt

Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores.

Qt es utilizada principalmente en Autodesk Maya, Dassault DraftSight, Google Earth, KDE, Adobe Photoshop Album, la Agencia Espacial Europea, Opie, Siemens, Volvo, Walt Disney Animation Studios, Skype, Qt Extended, VLC media player, Samsung, Philips, Panasonic, VirtualBox y Mathematica.

Es producido por la división de software Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008.

Qt es utilizada en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de bindings.

Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales. Distribuida bajo los términos de GNU Lesser General Public License (y otras), Qt es software libre y de código abierto.

En la figura 3, se muestra una captura del programa Qt Designer, el cual es parte de las herramientas que son incluidas con la librerías gráficas Qt.

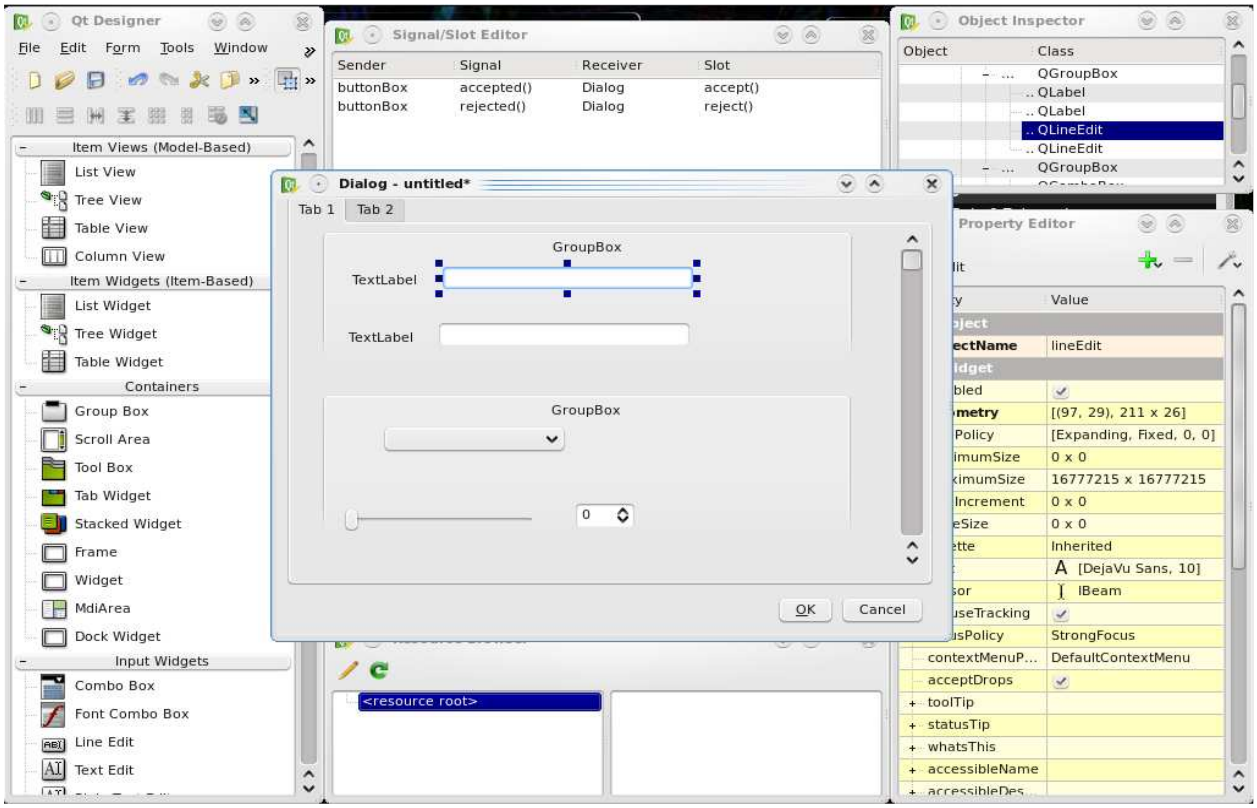

Figura 3. Captura de pantalla del programa Qt Designer

Capítulo 3: Desarrollo de la base de datos

Después de analizar las características de los tres DBMS comentados durante el capitulo 2, se ha optado por utilizar el motor de base de datos PostgreSQL, debido a su licencia así como sus características. MySQL ha debido ser descartada como candidata debido a las restricciones que imponía su licencia. Firebird poseía una licencia y unas características idóneas para el proyecto, pero el hecho de que la documentación de PostgreSQL es muy superior en calidad y cantidad a la que puede encontrarse de Firebird, ha hecho que quedara descartada como candidata.

En este capitulo definiremos los datos que deben ser almacenados en cada una de las tablas. Estos datos fueron determinados de acuerdo a las necesidades de los clientes.

3.1 Conceptos

Antes de empezar a definir la base de datos, debemos definir dos conceptos propios de las base de datos relacionales: la clave primaria y la clave foránea.

Definimos como clave primaria a un campo o a una combinación de campos que identifica de forma única a cada fila de una tabla. Una clave primaria comprende de esta manera una columna o conjunto de columnas. No puede haber dos filas en una tabla que tengan la misma clave primaria.

Una clave primaria es un caso especial de clave única. La mayor diferencia es que para claves únicas, no se impone automáticamente la restricción implícita NOT NULL, mientras que para claves primarias, sí. Así, los valores en columnas de clave única pueden o no ser NULL. Otra diferencia es que las claves primarias deben definirse por medio de otra sintaxis.

Mientras que una clave foránea (o Foreign Key FK) es una limitación referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla (tabla hija o referendo) que se refiere a una columna o grupo de columnas en otra tabla (tabla maestra o referenciada). Las columnas en la tabla referendo deben ser la clave primaria u otra clave candidata en la tabla referenciada.

Los valores en una fila de las columnas referendo deben existir solo en una fila en la tabla referenciada. Así, una fila en la tabla referendo no puede contener valores que no existen en la tabla referenciada. De esta forma, las referencias pueden ser creadas para vincular o relacionar información. Esto es una parte esencial de la normalización de base de datos. Múltiples filas en la tabla referendo pueden hacer referencia, vincularse o relacionarse a la misma fila en la tabla referenciada.

3.2 Desarrollo

La bases de datos de este proyecto se estructura en siete tablas principales. Estas siete tablas son: productos, tanques, operaciones en tanques, clientes, vehículos, conductores y operaciones.

3.2.1 Productos

Los productos es el nombre bajo el cual nos referiremos a cada uno de los combustibles que se disponen.

La tabla debe guardar los siguientes datos:

- Número del producto
  Este campo es del tipo numérico y podría tomar el valor de entre 1 y 100. Este número es del tipo único para cada producto.

- Nombre del producto
  Este campo es del tipo texto y es único, de forma que ningún producto mas pudiera recibir el mismo nombre.

- Precio/litro
  En este campo se guardaría el precio en euros por litro de producto. Este campo es numérico de forma que en representación decimal posee un parte entera de 4 dígitos y una parte decimal de 3 dígitos.

- Porcentaje de biodiesel
  En este campo se guarda el porcentaje de biodiesel que contiene el producto. Este campo es numérico y tiene un valor comprendido entre 0 y 100, ambos valores incluidos.

En la figura 4 se muestra el esquema gráfico de esta tabla.

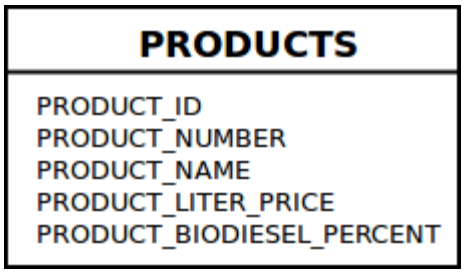

Figura 4. Esquema de la tabla productos

3.2.2 Tanques

Los tanques son los tanques de combustible donde se almacena en producto que utiliza el terminal.

La tabla debe guardar los siguientes datos:

- Número de tanque
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 100. Este número es único para cada tanque.

- Nombre del tanque
  Este campo es del tipo texto y es único, de forma que ningún tanque mas pudiera recibir el mismo nombre.

- Producto
  Este campo es una clave foránea que apunta a un elemento de la tabla productos. En este campo se indica cual es el producto que esta almacenado dentro del tanque.

- Capacidad total
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 6 dígitos. En este campo se indica cual la capacidad total del tanque.

- Nivel mínimo
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 5 dígitos. En este campo se indica cual es el nivel mínimo del combustible para mostrar un aviso al usuario.

- Nivel actual
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 6 dígitos y una parte decimal de 2 dígitos. En este campo se indica cual es el nivel actual de combustible del tanque.

En la figura 5 se muestra el esquema gráfico de esta tabla.

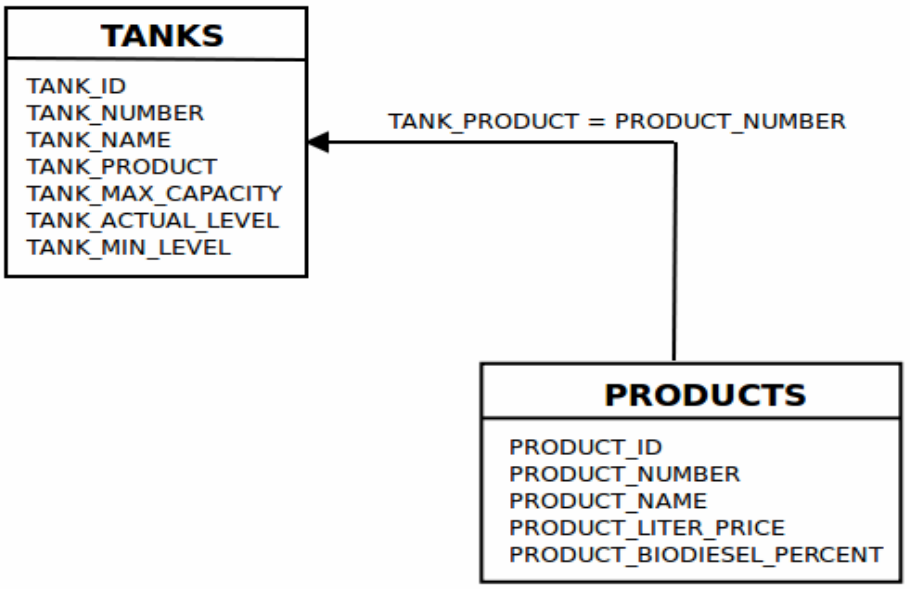

Figura 5. Esquema de la tabla tanques

3.2.3 Operaciones en los tanques

Las operaciones en los tanques es la tabla bajo la cual se guardan las operaciones que modifican el nivel actual de un determinado tanque. Estas operaciones pueden ser de medida (varillado) o de rellenado (compra). En las operaciones de medida se realiza una medición del nivel actual de un tanque y se introduce manualmente en la base de datos para corregir las perdidas que hayan podido producirse en el tanque, debido a que esta operación se realiza mediante una varilla graduada, esta operación recibe el nombre de varillado. En las operaciones de rellenado del tanque se introduce mas combustible en el tanque haciendo que su nivel varié, esta operación recibe el nombre de compra, debido a que desde el punto de vista de la gestión del terminal de consumos propios se ha realizado una compra de material.

Los datos que deben ser almacenados son:

- Número de operación
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 9999. Este número seria único para cada operación.

- Tipo de operación
  Este campo solo puede ser Varillado o Compra.

- Fecha
  Este campo es un campo con un formato específico para guardar la fecha de la operación.

- Hora
  Este campo es un campo con un formato específico para guardar la hora en la que se realizo la operación.

- Tanque
  Este campo es una clave foránea que apunta a un elemento de la tabla tanques. En este campo se indica cual es el tanque sobre el que se ha realizado la operación.

- Cantidad
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 6 dígitos. En este campo se indica cual la cantidad de litros que se han introducido en el tanque en una operación de compra, o la cantidad de litros que hay en el tanque después de una operación de varillado.

- Nombre del producto
  Este campo es del tipo texto. No es necesario que sea introducido por el usuario, ya que su valor será el nombre del producto que esta guardado en el tanque.

- Descripción
  Este campo es del tipo texto. Este campo es utilizado por el usuario para informar de incidencias o otros datos que puedan ser relevantes en esta operación.

- Proveedor
  Este campo es del tipo texto. Este campo es utilizado por definir el proveedor al que se le ha realizado la compra del producto.

- Número CAE del proveedor
  Este campo es del tipo texto. Este campo es utilizado por definir el número CAE del proveedor.

- Número de documento circulación vehículo
  Este campo es del tipo texto. Este campo es utilizado para almacenar el documento de circulación del vehículo que ha efectuado la operación de rellenado del tanque.

- Número de documento IIE
  Este campo es del tipo de texto. Este campo es utilizado para almacenar el documento IIE del proveedor.

En la figura 6 se muestra el esquema gráfico de esta tabla.

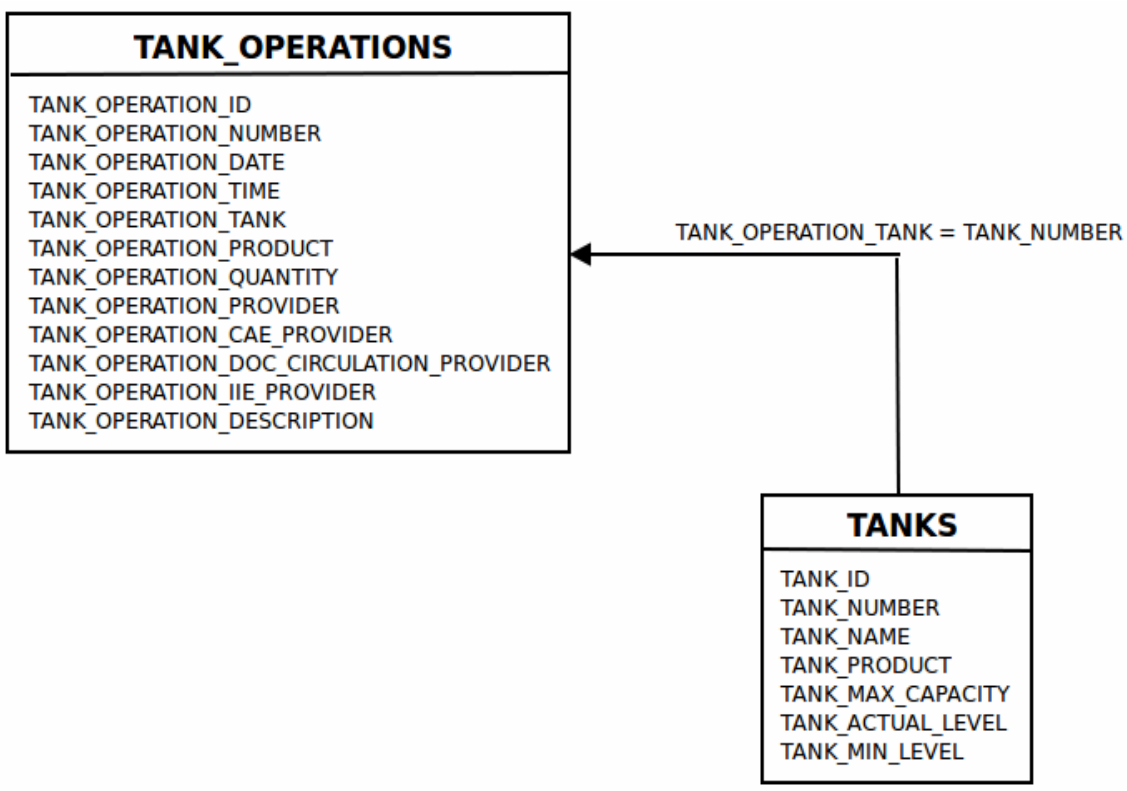

Figura 6. Esquema de la tabla de operaciones en los tanques

3.2.4 Clientes

La tabla debe guardar los siguientes datos:

- Número de cliente
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 10000. Este número es único para cada cliente.

- Nombre del cliente
  Este campo es del tipo de texto y es único, de forma que ningún cliente mas pudiera recibir el mismo nombre.

- NIF/CIF
  Este campo es del tipo de texto y único, de forma que ningún cliente mas pudiera recibir el mismo NIF/CIF.

- CAE
  Este campo es del tipo de texto. En este campo se utilizada por almacenar el número CAE del cliente.

- Dirección
  Este campo es del tipo de texto. En este campo se utilizada por almacenar la dirección del cliente.

- Código postal
  Este campo es del tipo numérico y podría tomar el valor de entre 1 y 99999. En este campo se utilizada por almacenar el código postal del cliente.

- Población
  Este campo es del tipo de texto. En este campo se utilizada por almacenar la población del cliente.

- Provincia
  Este campo es del tipo de texto. En este campo se utilizada por almacenar la provincia del cliente.

- Teléfono
  Este campo es del tipo de texto. En este campo se utilizada por almacenar el teléfono del cliente.

- Correo electrónico
  Este campo es del tipo de texto. En este campo se utilizada por almacenar el correo electrónico del cliente.

- Control de consumo
  Este campo es del tipo booleano. En este campo se define si al cliente se le aplicara el control de consumo mensual.

- Crédito mensual
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 6 dígitos. En este campo se define el consumo máximo en euros que puede efectuar mensualmente el cliente.

- Saldo consumido mensual
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 6 dígitos y una parte decimal de 2 dígitos. En este campo se define el saldo consumido por el cliente en el periodo transcurrido de mes, este campo es puesto a cero automáticamente a día uno del mes.

- Bloqueado
  Este campo es del tipo booleano. En este campo se define si el cliente o cualquiera de los activos del cliente se le permitirán realizar operaciones de repostaje.

En la figura 7 se muestra el esquema gráfico de esta tabla.

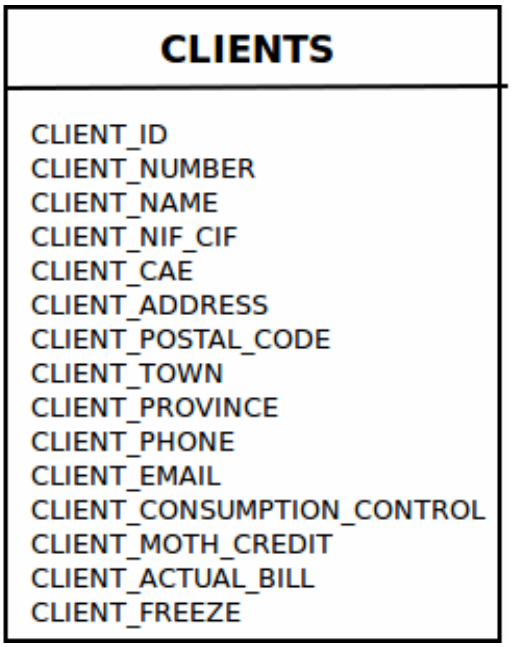

Figura 7. Esquema de la tabla clientes

3.2.5 Conductores

La tabla debe guardar los siguientes datos:

- Número de conductor
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 10000. Este número es único para cada conductor.

- Nombre
  Este campo es del tipo de texto y es único, de forma que ningún conductor mas pudiera recibir el mismo nombre.

- NIF
  Este campo es del tipo de texto y es único, de forma que ningún conductor mas pudiera recibir el mismo nif/cif.

- Identificador de tarjeta
  Este campo es del tipo de texto y único, de forma que ningún conductor mas pudiera recibir el mismo identificador de tarjeta. Aquí se introduce el número de tarjeta para la identificación del conductor en el terminal.

- Solicitar Pin
  Este campo es del tipo booleano. En este campo se define si al conductor necesitara introducir su número personal de identificación (PIN) en el terminal.

- Pin
  Este campo es del tipo numérico y puede tomar el valor de entre 0 y 9999.

- Cliente
  Este campo es una clave foránea que apunta a un elemento de la tabla clientes. En este campo se indica cual es el cliente al que pertenece el conductor del vehículo.

- Bloqueado
  Este campo es del tipo booleano. En este campo se define si el cliente o cualquiera de los activos del cliente se le permitirán realizar operaciones de repostaje.

En la figura 8 se muestra el esquema gráfico de esta tabla.

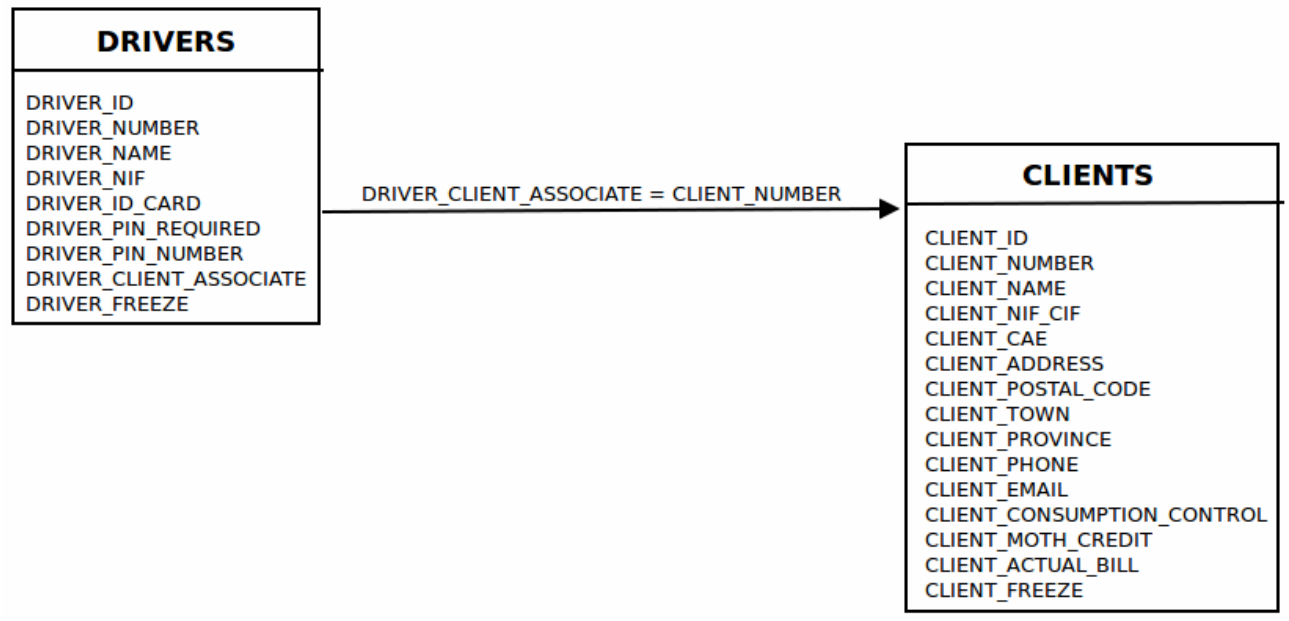

Figura 8. Esquema de la tabla conductores

3.2.6 Terminales

La tabla debe guardar los siguientes datos:

- Número de terminal
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 100. Este número es único para cada terminal.

- Nombre
  Este campo es del tipo de texto y es único, de forma que ningún terminal mas pudiera recibir el mismo nombre.

- Tipo de comunicación
  Este campo solo puede tomar el valor Serie o Red. Estos son los dos métodos de conexión con el terminal a través de un cable Serie o a través de dirección IP.

- Dirección
  Este campo es del tipo texto. En este campo se introduce la dirección IP que se utilizara para conectar con el terminal.

- Reconocimiento de matrícula
  Este campo es del tipo booleano. En este campo se define si el terminal hará uso de elementos externos para el reconocimiento de la matricula del vehículo.

- Detector de presencia de vehículo
  Este campo es del tipo booleano. En este campo se define si el terminal hará uso de elementos externos para el detectar que un vehículo a estacionado en la proximidad del terminal.

- Ticket
  Este campo es del tipo booleano. En este campo se define si el terminal entregara al usuario del terminal un ticket después de la operación de repostaje.

- Encabezado del ticket línea 1
  Este campo es del tipo texto. En este campo se define la primera línea de texto que será mostrada en el ticket.

- Encabezado del ticket línea 2
  Este campo es del tipo texto. En este campo se define la segunda línea de texto que será mostrada en el ticket.

- Encabezado del ticket línea 3
  Este campo es del tipo texto. En este campo se define la tercera línea de texto que será mostrada en el ticket.

- Encabezado del ticket línea 4
  Este campo es del tipo texto. En este campo se define la cuarta línea de texto que será mostrada en el ticket.

- Pie ticket
  Este campo es del tipo texto. En este campo se define la ultima línea de texto que será mostrada en el ticket.

- Número de intentos PIN
  Este campo es del tipo numérico y puede tomar el valor de entre 0 y 10. En este campo se especifica el número de intentos fallidos en la introducción del PIN, antes de bloquear al usuario.

- Temporizador PIN
  Este campo es del tipo numérico y puede tomar el valor de entre 0 y 100. En este campo se especifica el tiempo en minutos que debe de esperar el usuario que ha consumido el numero máximo de intentos en la introducción del pin.

- Temporizador descuelgue
  Este campo es del tipo numérico y puede tomar el valor de entre 0 y 99. En este campo se especifica el tiempo máximo en minutos desde que el terminal a autorizado la operación de repostaje y el comienzo de repostaje por parte del usuario, transcurrido este tiempo la operación se invalidara.

- Temporizador servicio
  Este campo es del tipo numérico y puede tomar el valor de entre 0 y 99. En este campo se especifica el tiempo máximo que puede durar la operación de repostaje.

- Temporizador pulsos
  Este campo es del tipo numérico y puede tomar el valor de entre 0 y 99. En este campo se especifica el tiempo máximo que puede durar la operación de repostaje.

- Polaridad descuelgue manguera 1
  Este campo puede tomar el valor NA (Normalmente abierto) o NC (Normalmente cerrado).

- Pulsos/litro manguera 1
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 1000. En este campo se especifica el número de pulsos que el terminal recibe por litro de combustible.

- Tanque manguera 1
  Este campo es una clave foránea que apunta a un elemento de la tabla tanques. En este campo se indica a cual de los tanques esta conectada la manguera.

- Polaridad descuelgue manguera 2
  Este campo puede tomar el valor NA (Normalmente abierto) o NC (Normalmente cerrado).

- Pulsos/litro manguera 2
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 1000. En este campo se especifica el número de pulsos que el terminal recibe por litro de combustible.

- Tanque manguera 2
  Este campo es una clave foránea que apunta a un elemento de la tabla tanques. En este campo se indica a cual de los tanques esta conectada la manguera.

- Polaridad descuelgue manguera 3
  Este campo puede tomar el valor NA (Normalmente abierto) o NC (Normalmente cerrado).

- Pulsos/litro manguera 3
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 1000. En este campo se especifica el numero de pulsos que el terminal recibe por litro de combustible.

- Tanque manguera 3
  Este campo es una clave foránea que apunta a un elemento de la tabla tanques. En este campo se indica a cual de los tanques esta conectada la manguera.

- Polaridad descuelgue manguera 4
  Este campo puede tomar el valor NA (Normalmente abierto) o NC (Normalmente cerrado).

- Pulsos/litro manguera 4
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 1000. En este campo se especifica el número de pulsos que el terminal recibe por litro de combustible.

- Tanque manguera 4
  Este campo es una clave foránea que apunta a un elemento de la tabla tanques. En este campo se indica a cual de los tanques esta conectada la manguera.

- Bloqueado
  Este campo es del tipo booleano. En este campo se define si el cliente o cualquiera de los activos del cliente se le permitirán realizar operaciones de repostaje.

En la figura 9 se muestra el esquema gráfico de esta tabla.

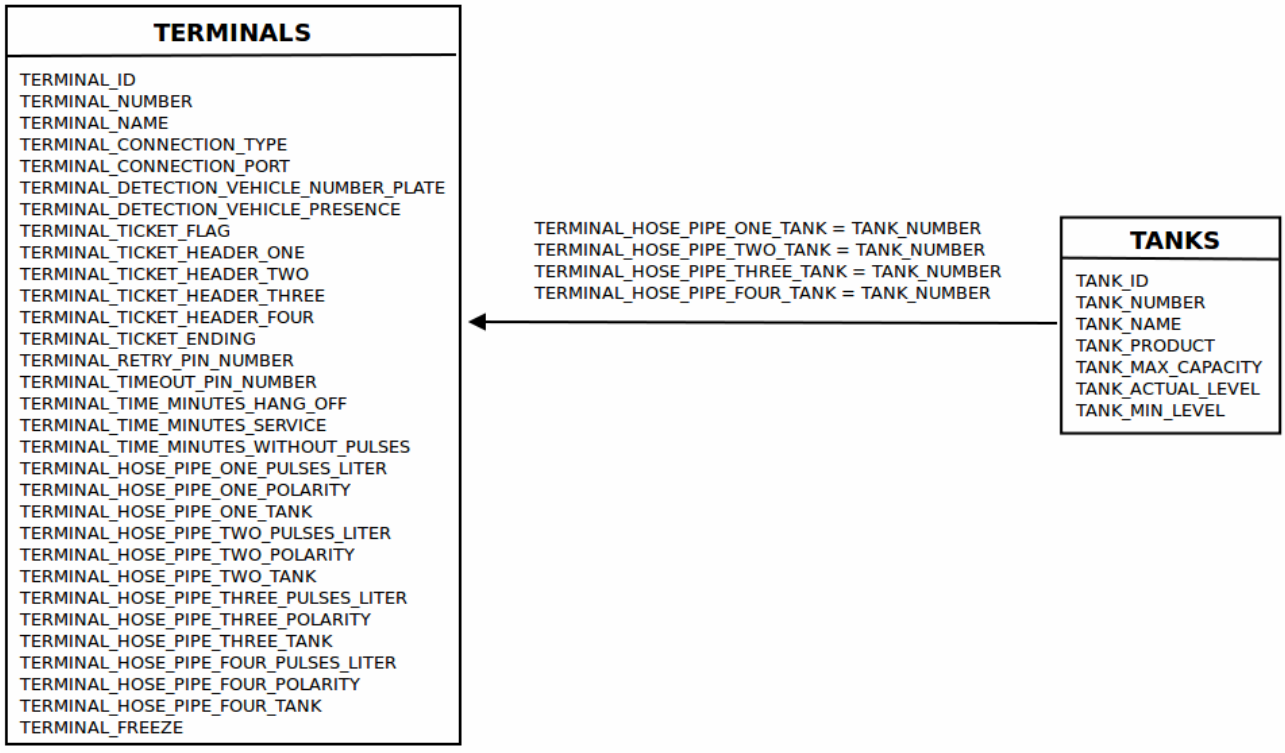

Figura 9. Esquema de la tabla terminales

3.2.7 Vehículo

La tabla debe guardar los siguientes datos:

- Número de vehículo
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 10000. Este número es único para cada conductor.

- Matrícula del vehículo:
  Este campo es del tipo de texto y es único, de forma que ningún vehículo mas pudiera recibir la misma matricula.

- Fabricante del vehículo
  Este campo es del tipo de texto.

- Modelo del vehículo
  Este campo es del tipo de texto.

- Identificador de tarjeta
  Este campo es del tipo de texto y único, de forma que ningún vehículo mas pudiera recibir el mismo identificador de tarjeta. Aquí se introduce el número de tarjeta para la identificación del vehículo en el terminal.

- Solicitar Pin
  Este campo es del tipo booleano. En este campo se define si el vehículo necesitara introducir su número personal de identificación (PIN) en el terminal.

- Pin
  Este campo es del tipo numérico y podría tomar el valor de entre 0 y 9999.

- Control de consumo
  Este campo es del tipo booleano. En este campo se define si al vehículo se le aplicara el control de consumo mensual.

- Crédito mensual
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 6 dígitos. En este campo se define el consumo máximo en euros que puede efectuar mensualmente el vehículo.

- Saldo consumido mensual
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 6 dígitos y una parte decimal de 2 dígitos. En este campo se define el saldo consumido por el cliente en el periodo transcurrido de mes, este campo es puesto a cero automáticamente a día uno del mes.

- Control de km
  Este campo es del tipo booleano. En este campo se define si al vehículo se le aplicara el control de kilometraje.

- Km actuales
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 6 dígitos. En este campo se define los kilómetros que ha realizado el vehículo, este campo es actualizado en cada operación a través del terminal, mediante la introducción por parte del conductor de los kilómetros actuales del vehículo.

- Margen de Km
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 5 dígitos. En este campo se define el margen de error que se permita al introducir en el terminal, los kilómetros que ha realizado el vehículo.

- Gasóleo profesional
  Este campo es del tipo booleano. En este campo se define si al vehículo se le permitirá realizar operación de repostaje de gasóleo profesional.

- Requiere identificación del conductor
  Este campo es del tipo booleano. En este campo se define si al repostar es necesario la identificación del conductor después de la identificación del vehículo

- Cliente
  Este campo es una clave foránea que apunta a un elemento de la tabla clientes. En este campo se indica cual es el cliente al que pertenece el conductor del vehículo.

- Producto 1
  Este campo es una clave foránea que apunta a un elemento de la tabla productos. En este campo se indica uno de los productos de los cuales el vehículo puede repostar.

- Producto 2
  Este campo es una clave foránea que apunta a un elemento de la tabla productos. En este campo se indica uno de los productos de los cuales el vehículo puede repostar.

- Producto 3
  Este campo es una clave foránea que apunta a un elemento de la tabla productos. En este campo se indica uno de los productos de los cuales el vehículo puede repostar.

- Producto 4
  Este campo es una clave foránea que apunta a un elemento de la tabla productos. En este campo se indica uno de los productos de los cuales el vehículo puede repostar.

- Autorización de terminales
  Este campo es del tipo booleano. En este campo se define si se llevara un control del los terminales en los cuales el vehículo puede repostar.

- Terminal 1
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 2
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 3
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 4
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 5
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 6
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 7
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 8
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 9
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Terminal 10
  Este campo es una clave foránea que apunta a un elemento de la tabla terminales. En este campo se indica uno de los terminales de los cuales el vehículo puede utilizar.

- Bloqueado
  Este campo es del tipo booleano. En este campo se define si el cliente o cualquiera de los activos del cliente se le permitirán realizar operaciones de repostaje.

En la figura 10 se muestra el esquema gráfico de esta tabla.

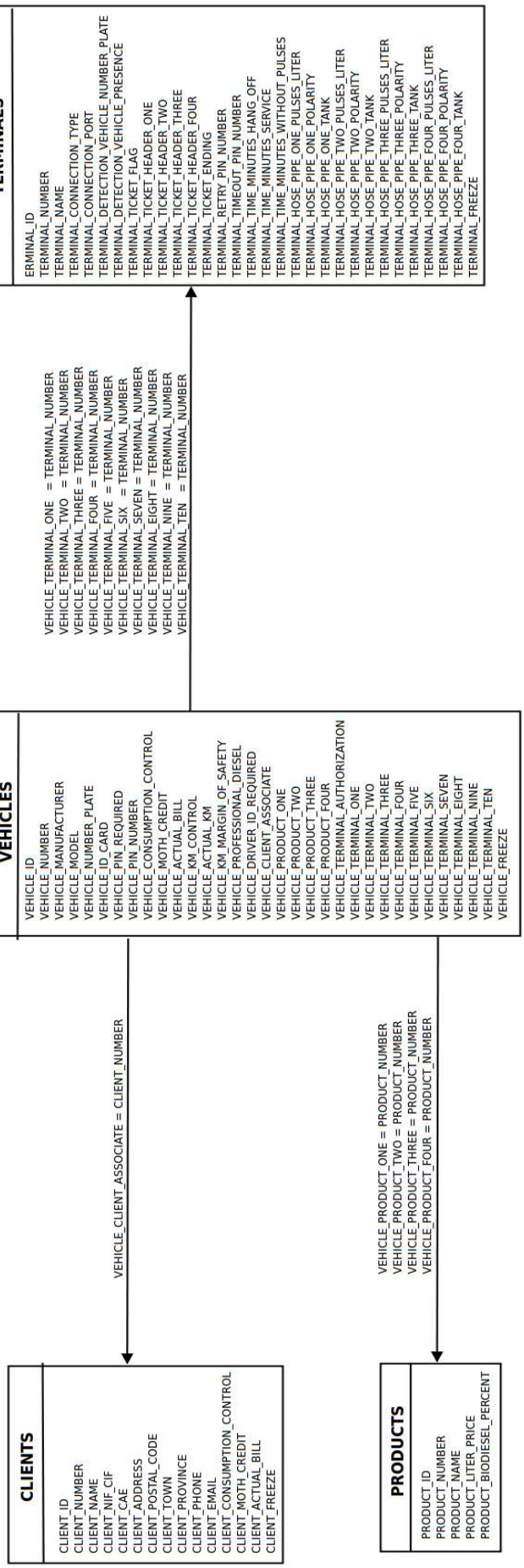

Figura 10. Esquema de la tabla vehículos

3.3.8 Operaciones

La tabla debe guardar los siguientes datos:

- Tipo de operación
  Este campo solo puede tomar dos valores: automática o manual. Siendo una operación automática aquella que ha sido registrada a través del terminal y una operación manual, aquella que ha sido introducida a través del programa.

- Número de operación
  Este campo es del tipo numérico y puede tomar el valor de entre 1 y 9999999. Este número es único para cada operación.

- Fecha
  Este campo es un campo con un formato específico para guardar la fecha de la operación.

- Hora
  Este campo es un campo con un formato específico para guardar la hora en la que se realizo la operación.

- Número de terminal
  Este campo es de tipo numérico. En este campo se almacena el número del terminal donde se ha realizado la operación.

- Número de cliente
  Este campo es de tipo numérico. En este campo se almacena el número del cliente al que pertenece el vehículo que ha realizado la operación.

- Nombre del cliente
  Este campo es de tipo texto. En este campo se almacena el nombre del cliente al que pertenece el vehículo que ha realizado la operación.

- Número del conductor
  Este campo es de tipo numérico. En este campo se almacena el número del conductor que ha realizado la operación.

- Nombre del conductor
  Este campo es de tipo texto. En este campo se almacena el nombre del conductor que ha realizado la operación.

- Nombre del producto
  Este campo es de tipo texto. En este campo se almacena el nombre del producto que ha sido repostado en la operación.

- Cantidad
  Este campo seria numérico de forma que en representación decimal posee un parte entera de 4 dígitos y la parte decimal 2 dígitos. En este campo se indica cual es la cantidad de litros que se han repostado en el vehículo.

- Precio/litro
  Este campo es numérico de forma que en representación decimal posee un parte entera de 4 dígitos y una parte decimal de 3 dígitos. En este campo se guardaría el precio en euros por litro de producto.

- Porcentaje de biodiesel
  En este campo se guarda el porcentaje de biodiesel que contiene el producto. Este campo es numérico y tiene un valor comprendido entre 0 y 100, ambos valores incluidos.

- Importe total
  Este campo es numérico de forma que en representación decimal posee un parte entera de 5 dígitos y una parte decimal de 2 dígitos. En este campo se almacena el importe total de la operación.

- Número de vehículo
  Este campo es del tipo numérico. En este campo se almacena el número del vehículo que ha realizado la operación.

- Matrícula del vehículo
  Este campo es del tipo de texto. En este campo se almacena la matricula del vehículo que ha realizado la operación.

- Km. del vehículo
  Este campo es del tipo numérico de forma que en representación decimal posee un parte entera de 7 dígitos. En este campo se define los kilómetros que ha realizado el vehículo en el momento de la operación de repostaje.
- Gasóleo profesional
  Este campo es del tipo booleano. En este campo se define si en la operación se ha utilizado gasóleo profesional.

- Corregido manualmente
  Este campo es del tipo booleano. Si en una operación automática, se ha modificado los valores a después de la operación automática, este campo pasa a estar activo.

- Estado
  En este campo se almacena el estado en que acabo la operación. Este campo puede tomar los siguientes valores:
  0 Correcto.
  1 Vehículo no autorizado en terminal.
  2 Reconocimiento de matrícula.
  3 PIN del vehículo incorrecto.
  4 PIN del conductor incorrecto.
  5 Producto no autorizado.
  6 Saldo insuficiente.
  7 Temporización descuelgue.
  8 Temporización servicio.
  9 Temporización pulsos.

En la figura 11 se muestra el esquema gráfico de esta tabla.

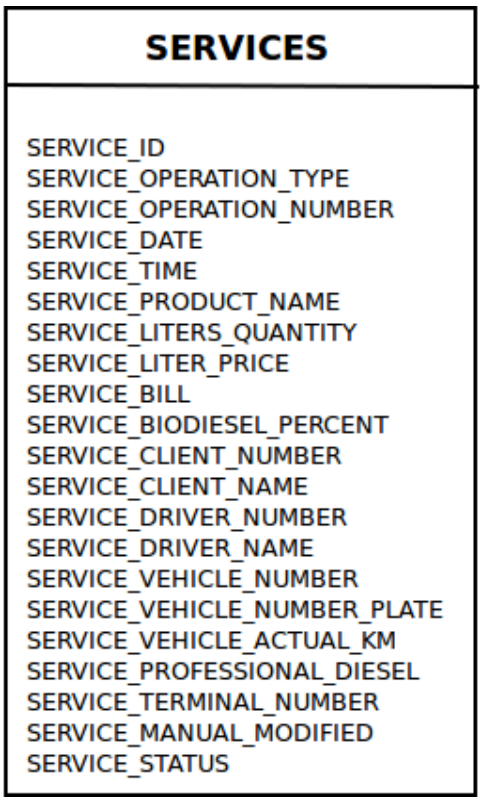

Figura 11. Esquema de la tabla servicios

Capítulo 4: Desarrollo del software

En el capitulo 2, hemos analizado las tres principales librerías gráficas que podían ser candidatas a ser elegidas para el desarrollo del software. Después de evaluarlas se ha tomado la decisión de utilizar como librería gráfica del software de gestión a Qt. Se ha descartado el uso de GTK debido a las restricciones de su licencia y aunque wxwidgets y Qt cumplían los requerimientos, nos hemos decantado por Qt debido a que la empresa donde se ha realizado este proyecto poseía experiencia previa en Qt.

Debido a que este trabajo de final de carrera ha sido realizado como practicas de empresa no puedo publicar el código fuente. Por lo que en este capitulo analizare la jerarquía de clases y mostrare el modo de funcionamiento de la aplicación a través de capturas de pantalla.

4.1 Pecularidades de las librerias Qt

Antes de empezar a describir la jerarquía de clases que conforma la aplicación, algunas de las peculiaridades de Qt , deben ser descritas para comprender algunas decisiones de diseño.

4.1.1 Signals & Slots

En una librería gráficas, normalmente cada elemento de la interfaz gráfica genera un evento al realizar una acción sobre el. El programador solo debe escribir una función que procese los eventos y a partir del evento que llame a una función o otra. Este seria el enfoque clásico en las librerías gráficas.

Aunque en Qt se puede programar utilizando de esta manera y algunas partes poco frecuentes deben ser programadas de esta manera, Qt aporta un enfoque diferente. La idea es, cuando se produce un evento en un objeto, éste puede emitir una señal. Esa señal se puede conectar con un slot (que no deja de ser un método de otro objeto) que responde a esa señal.

Para poder hacer uso de esta técnica, las clases que emitan señales o contengan slots, deben indicar que son objetos Qt, incluyendo la directiva Q_OBJECT en su declaración.

Así, por ejemplo, supongamos un objeto botón típico de aplicación, QPushButton. Cuando el usuario pulsa el botón, se produce una señal, clicked, que podemos conectar con un slot en el que respondemos a esa pulsación. Las conexiones se pueden establecer y desestablecer en tiempo de ejecución.

Además, es posible conectar una señal a varios slots. También es posible crear nuestros propios objetos que emitan nuestras propias señales (para emitir una señal basta emit(miSeñal()), definiendo miSeñal dentro de la declaración de la clase, bajo "signals:". Así se pueden extender los controles o los objetos adecuados fácilmente.

Entre las críticas que recibe este sistema es que no es un método compatible con Ansi C++, por lo que antes de compilar los fuentes, éstos deben ser analizados con una herramienta que proporcionan las Qt, que se llama "moc" y que se encarga de traducir esa jerga especial de "signals, slots, Q_OBJECT" en código intermedio que realiza la comunicación. Sin embargo, hay que tener en cuenta que este sistema no implica que deban realizarse modificaciones en el compilador, incluyendo directivas de compilador.

4.1.2 Modulo Interview

A partir de la versión 4 de Qt, se le añadieron mejoras que no poseían las versiones anteriores. Estas mejoras fueron separadas en módulos y denominadas con nombre clave. Varios de estos módulos son:

- Tulip, es un nuevo conjunto de contenedores que sustituyen al antiguo QCollection.
- Interview, proporciona un nuevo modelo de vista para las aplicaciones Qt basadas en la arquitectura Modelo/Vista/Controlador
- Arthur, es un Framework de Qt4 para realizar operaciones de dibujo, basado en las principales clases tales como QPainter, QPaintDevice y QPaintEngine.
- Scribe, un procesador de texto Unicode con una API pública para la manipulación de texto plano.

En nuestro caso, vamos a utilizar la clase interview por lo cual es interesante comentar un poco sobre ella.

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

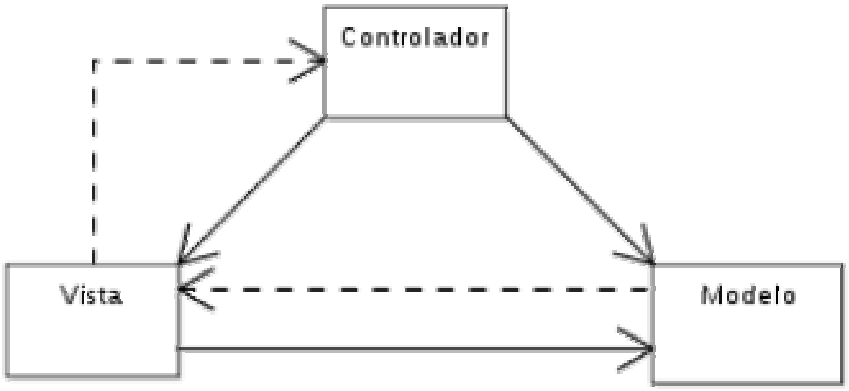

Figura 12. Modelo Vista Controlador

Analizando cada elemento por separado:

- Modelo
  Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

- Vista
  Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

- Controlador
  Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

En Qt existen clases para definir modelos de datos que pueden ser del tipo listas, consultas SQL, tablas SQL, lista de IPs, listados de ficheros, etc. Para la representaciones gráficas se utilizan, clases el tipo vista que pueden ser tablas, arboles o listas sencillas. Cuando un clase del tipo vista va a representar un elemento de la clase modelo no accede directamente a la clase modelo sino que solicita el dato a una clase Controlador que le dice como debe representar el dato. En Qt estas clases reciben el nombre de Delegates y efectúan la acción anterior.

En nuestra aplicación utilizaremos en modulo interview el cual nos facilitara la representación en forma de tablas de los datos de la base de datos.

4.2 Jerarquía de clases

Como se puede observar en la figura 13, la clase central de la aplicación es mainWindow, la cual define y gestiona la ventana principal de la aplicación, en ella están incluidas por composición todas las demás clases que conforman el programa menos la clase globalSettings.

La clase globalSettings es una clase del tipo singleton (instancia única), esta clase está diseñada para restringir la creación de objetos pertenecientes a ella con intención de garantizar que sólo exista una instancia y proporcionar un punto de acceso global a ella. De esta forma al almacenar la configuración global de la aplicación en esta clase, todos los accesos a la configuración serán a través del mismo objeto.

La clase dataBaseSession tiene como función encapsular todas las funciones de conexión y mantenimiento de la base de datos, de forma que la única clase en todo la aplicación que realiza operaciones de escritura en la base de datos esta en esta clase.

En la clase preferencesDialog, crea y gestiona una ventana que permite gestionar al usuario toda la configuración global del programa y almacenarla luego en la clase globalSettings.

Como se puede observar se ha creado una clase interface (en C++, se denomina clase virtual pura) para los distintas ventanas que deben formar parte del programa: productos, vehículos, clientes, conductores, operaciones, terminales, tanques y operaciones en tanques; Cada una de estas ventanas implementa la clase interfaceWidget, de forma que la clase mainWindow siempre ve una clase interfaceWidget.

Cada uno de las ventanas esta compuesta por una clase que crea y gestiona una ventana, así como de varias clases del tipo delegate para mostrar los datos de forma amigable. Así como de una clase dialogo para que el usuario pueda modificar e introducir los datos en la base de datos. Todas las ventanas poseen la misma estructura, por ello y para que fuera mas sencilla la compresión de la jerarquía de clases, no he puesto los demás ventanas en el esquema.

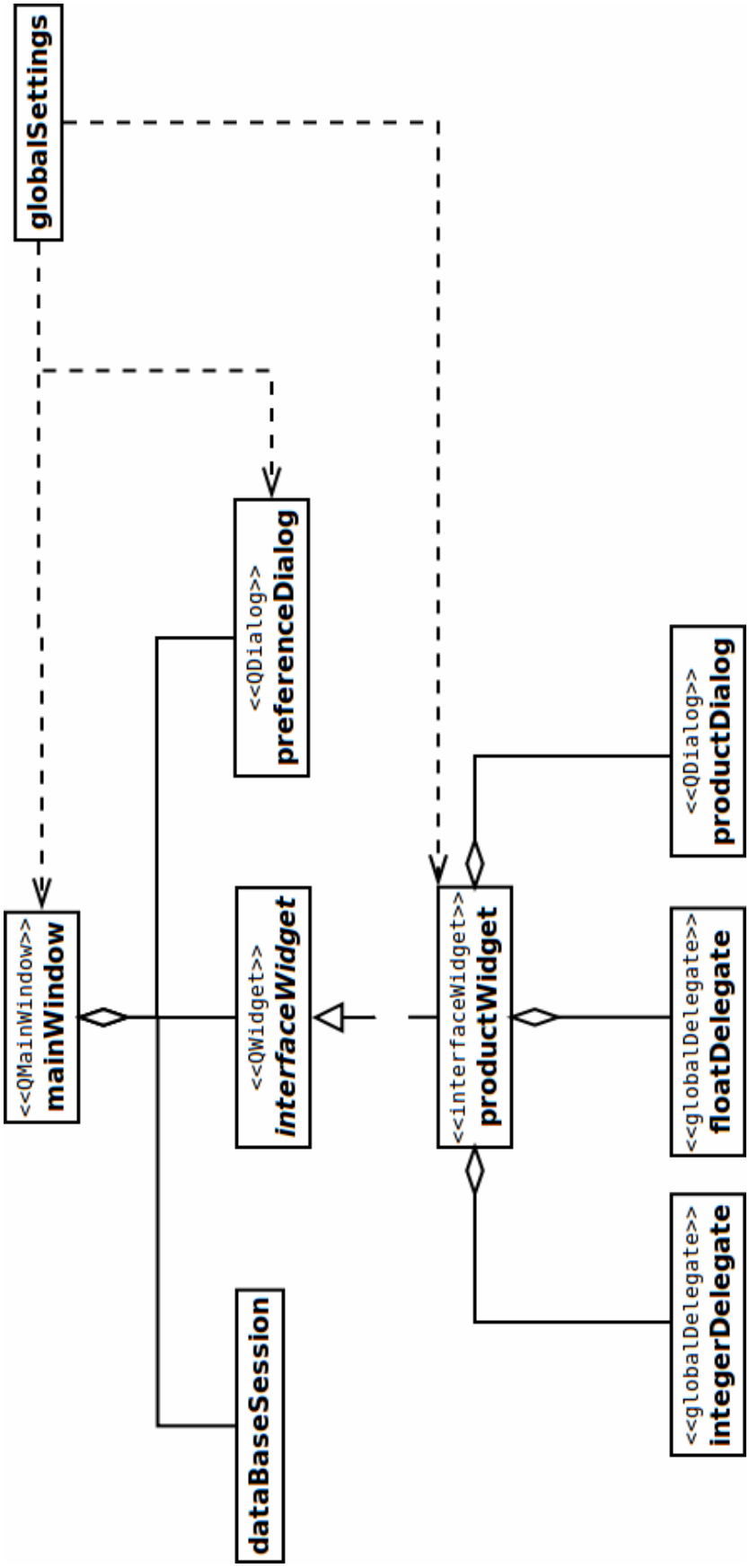

Figura 13. Jerarquía de clases

4.3 Modo de funcionamiento

Al iniciar el programa, nos encontramos con la pantalla mostrada en la figura 14, en la que se nos solicita en usuario y la contraseña.

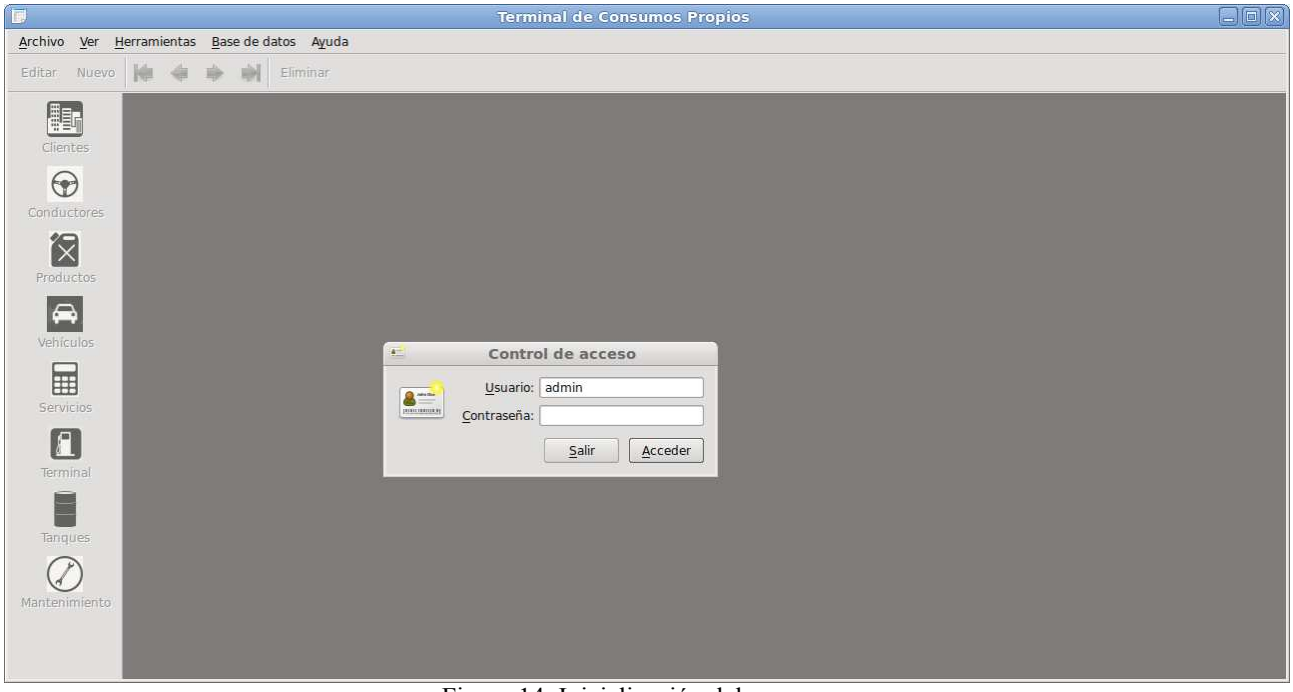

Figura 14. Inicialización del programa

En la figura 15 se muestra el dialogo de control de acceso. En este dialogo se solicita el usuario y contraseña que serán utilizados para iniciar una sesión en la base de datos. Una vez introducido un usuario y contraseñas correctos, los diferentes menús de la aplicación pasan a estar activos.

Figura 15. Dialogo de control de acceso

4.3.1 Menús y barra de Herramientas

En la figura 16 puede observarse la barra del menú de la aplicación junto a la barra de Herramientas.

Figura 16. Barra de menú y herramientas

En la barra del menú se cumple la siguiente jerarquía:

- Archivo
  - Iniciar Sesión
    Cuando existe una conexión a la base de datos esta acción esta desactivada, sino existe conexión se activa. Al ejecutarla muestra el dialogo de control de acceso y luego gestiona la conexión con la base de datos.

  - Cerrar Sesión
    Si existe conexión con la base de datos, permanece activa. Si no existe conexión permanece desactivada. Al ejecutar esta acción, se realiza la desconexión de la aplicación a la base de dato.

  - Preferencias
    Esta acción muestra en pantalla en dialogo de preferencias de la aplicación.

  - Salir
    Esta acción cierra la aplicación.

- Ver
  - Mostrar Barra de Acceso Rápido
    Esta acción muestra o oculta, dependiendo del estado previo, la barra de Acceso Rápido. Esta barra, que en la figura donde se muestra el inicio de la aplicación se puede observar a la derecha de la ventana principal, muestra iconos con los lanzadores para ventanas de gestión de las tablas.

  - Mostrar Barra de Herramientas
    Esta acción muestra o oculta, dependiendo del estado previo, la barra de Acceso Rápido. Esta barra, que en la figura donde se muestra el inicio de la aplicación se puede observar justo debajo de la barra de menús, muestra los iconos necesarios para tratar con la tablas, que se muestran en las ventanas de gestión.

- Navegación
    - Editar
      Esta acción hace que se muestre el dialogo para modificar un elemento de la tabla.

    - Nuevo
      Esta acción hace que se muestre el dialogo para introducir los datos para un nuevo elemento de la tabla.

    - Primero
      Esta acción hace que el elemento seleccionado de la tabla pase a ser el primero de la lista.

    - Anterior
      Esta acción hace que el elemento seleccionado de la tabla pase a ser el anterior al seleccionado.

    - Siguiente
      Esta acción hace que el elemento seleccionado de la tabla pase a ser el siguiente al seleccionado.

    - Último
      Esta acción hace que el elemento seleccionado de la tabla pase a ser el último de la tabla.

    - Eliminar
      Esta acción hace que se elimine el elemento seleccionado de la lista.

- Base de datos
    - Clientes
      Esta acción muestra la ventana para la gestión de los clientes.

    - Conductores
      Esta acción muestra la ventana para la gestión de los conductores.

    - Productos
      Esta acción muestra la ventana para la gestión de los productos.

    - Vehículos
      Esta acción muestra la ventana para la gestión de los vehículos.

    - Servicios
      Esta acción muestra la ventana para la gestión de los servicios.

    - Terminales
      Esta acción muestra la ventana para la gestión de los terminales.

    - Tanques
      Esta acción muestra la ventana para la gestión de los tanques.

- Mantenimiento
  Esta acción muestra la ventana para la gestión de las operaciones de mantenimiento de los tanques.

- Ayuda
  - Acerca
    Esta acción muestra el acerca de la aplicación, mostrando el nombre y la version de la aplicación.

  - Acerca QT
    Esta acción muestra el acerca de las librerías Qt, donde puede apreciarse bajo que licencia están registradas.

4.3.2 Barra de Acceso Rápido

En la figura 17 se muestra una captura de la barra de Acceso Rápido. Esta barra tiene los lanzadores para las distintas ventanas de gestión que posee el programa.

Figura 17. Barra de acceso rápido

4.3.3 Preferencias

En la figura 18 se muestra el dialogo de preferencias de la aplicación. En este dialogo se muestran las opciones para gestionar la aplicación. Como se puede observar el dialogo posee un sistema de pestañas, donde se puede seleccionar otra pestaña para la gestión de las fuentes de la aplicación.

Las diferentes opciones de configuración son:

- Confirmar antes de salir
  Si esta opción esta activa, se le solicitara una confirmación al usuario antes de cerrar la aplicación.

- Mostrar icono en la bandeja de notificaciones
  Si esta opción esta activa, se muestra un icono en la bandeja de notificaciones. Desde este icono el usuario puede maximizar o minimizar la aplicación, así como gestionar la sesión.

- Recordar último usuario
  Si esta opción esta activa, al mostrarse la ventana de control de acceso, en el campo de usuario se incluirá automáticamente el último usuario que inicio una sesión.

- Mostrar ventana de inicio
  Si esta opción esta activa, durante la inicialización de la aplicación se mostrar una ventana con el logotipo de la empresa.

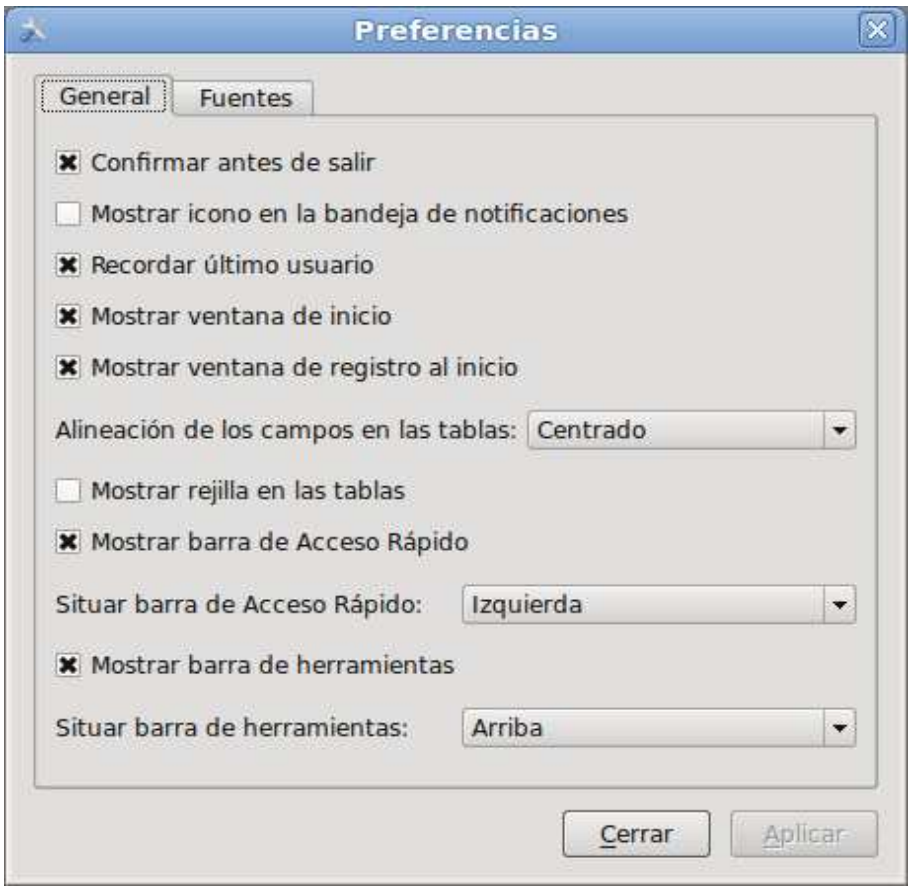

Figura 18. Dialogo de preferencias

- Mostrar ventana de registro al inicio
  Si esta opción esta activa, se mostrara al inicio de la aplicación la ventana de control de acceso.

- Alineación de los campos en las tablas
  A través de esta opción es posible indicar la alineación que mostraran cada una de las columnas en las tablas. Las posibles alineaciones son centrado, derecha e izquierda.

- Mostrar rejillas en las tablas
  Si esta opción esta activa, la tabla presentaran una rejilla para separar las distintas columnas.

- Mostrar barra de Acceso Rápido
  Si esta opción esta activa, se mostrara en pantalla la barra de Acceso Rápido.

- Situar barra de Acceso Rápido
  A través de esta opción es posible indicar donde esta colocada la barra de Acceso Rápido. Las posibles opciones de esta opción son: arriba, abajo, derecha e izquierda.

- Mostrar barra de Herramientas
  Si esta opción esta activa, se mostrara en pantalla la barra de Herramientas.

- Situar barra de Herramientas
  A través de esta opción es posible indicar donde esta colocada la barra de Herramientas. Las posibles opciones de esta opción son: arriba, abajo, derecha e izquierda.

En la figura 19 se muestra la pestaña de fuentes de las aplicación, con esta pestaña podemos definir las fuente que serán utilizadas en la aplicación como el las tablas.

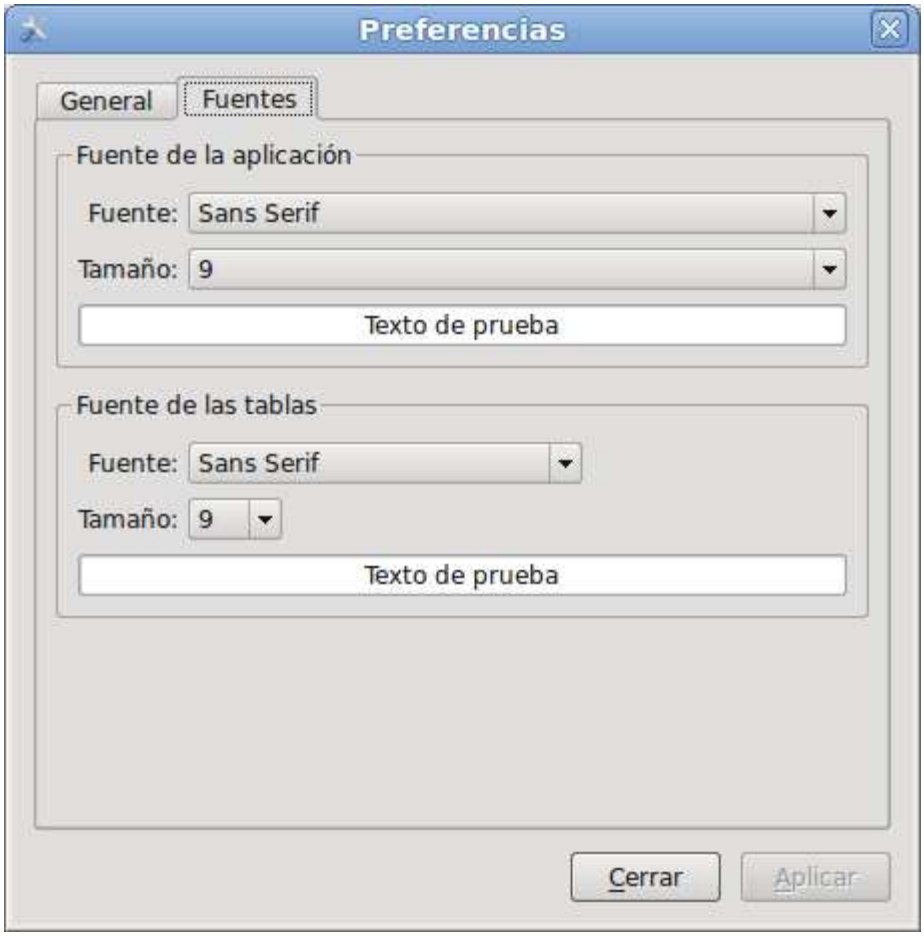

Figura 19. Dialogo de fuentes

4.3.4 Tablas

En la figura 20 se muestra la ventana de gestión de clientes, la cual posee como elemento central la tabla de clientes.

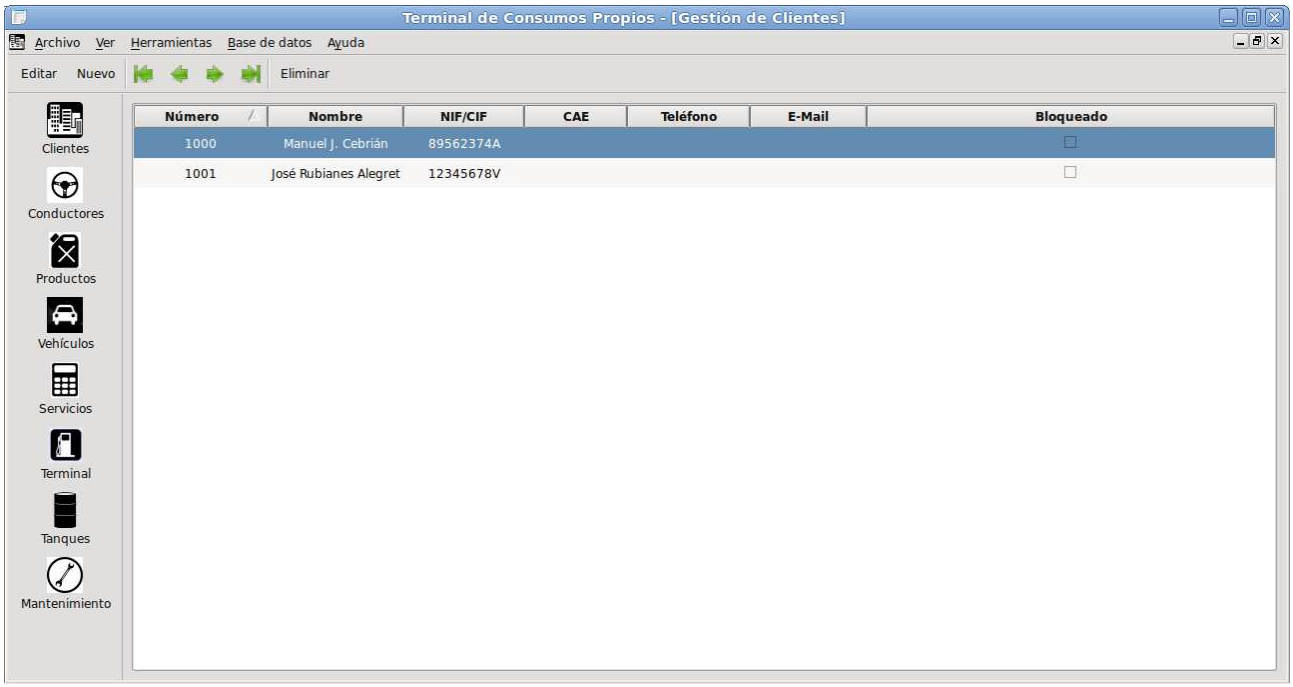

Figura 20. Gestión de clientes

A través de los menús de navegación o con el puntero del ratón o utilizando las flechas del teclado se puede desplazar a través de los distintos elementos de la tabla. Al pulsar a editar o hacer doble click en cualquier elemento se muestra el dialogo con el formulario para la modificación de los datos.

En la captura mostrada en la figura 21 puede observar con mayor detalle parte de la tabla.

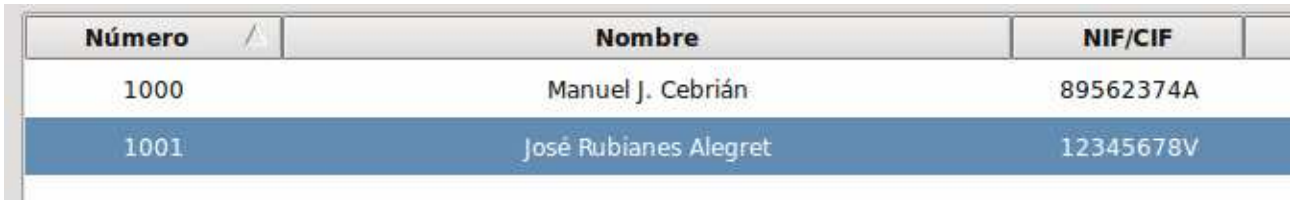

Figura 21. Tabla clientes

En la figura 22 se puede apreciar como a través de una clase delegado se ha utilizado un iconos gráficos en la tabla para representar valores booleanos.

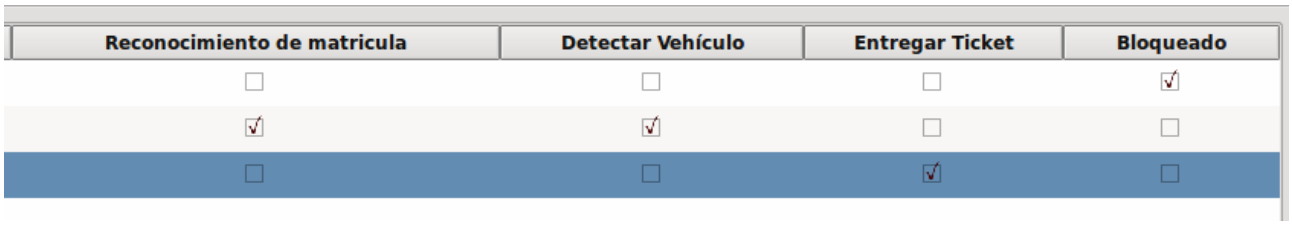

Figura 22. Valores booleanes en las tablas

En la figura 23 se muestra el uso de clases delegado para mostrar variables de tiempo en un formato cómodo de leer para el usuario.

Número	Operación	Fecha	Hora	Tanque	Producto	Cantidad
01	Compra	2010/07/21	12:59:00	1 - Deposito 01	Gasoleo A	5000
02	Compra	2010/07/22	13:29:00	1 - Deposito 01	Gasoleo A	5000
03	Compra	2010/07/13	22:59:00	2 - Deposito 02	Gasoleo A	5073

Figura 23. Representación de fecha y hora en la tabla

En la figura 24 se muestra el menú que se despliega al hacer click con el botón secundario sobre la tabla.

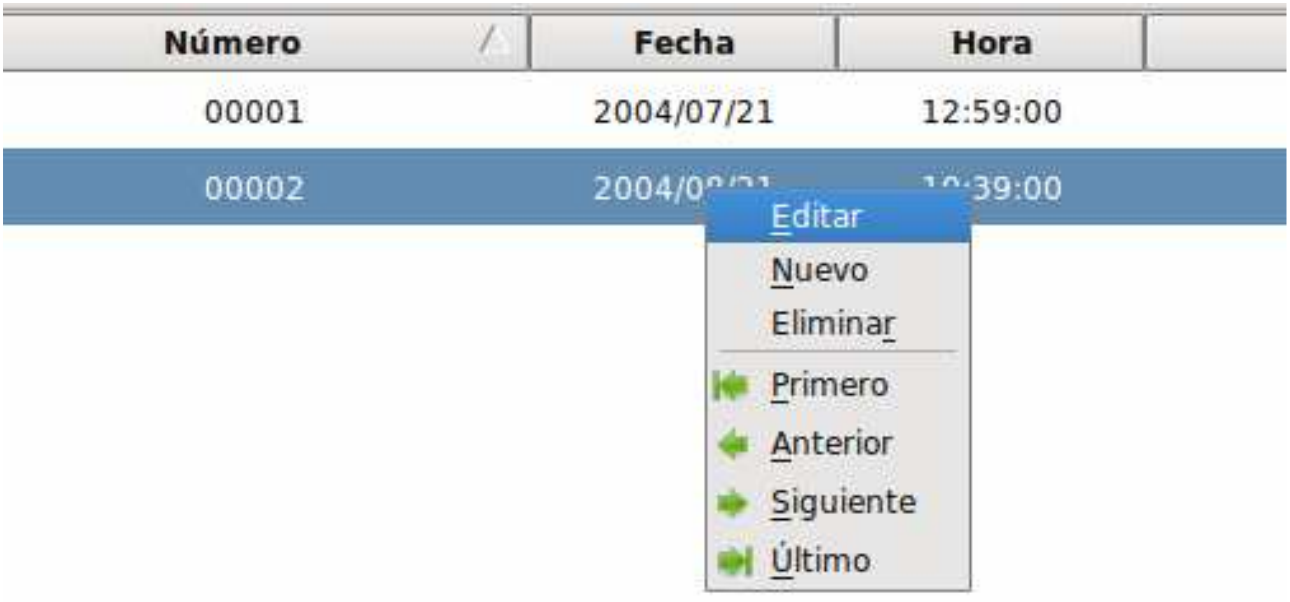

Figura 24. Menú desplegable

4.3.5 Clientes

En la figura 25 se muestra el dialogo de modificación de los datos de los clientes.

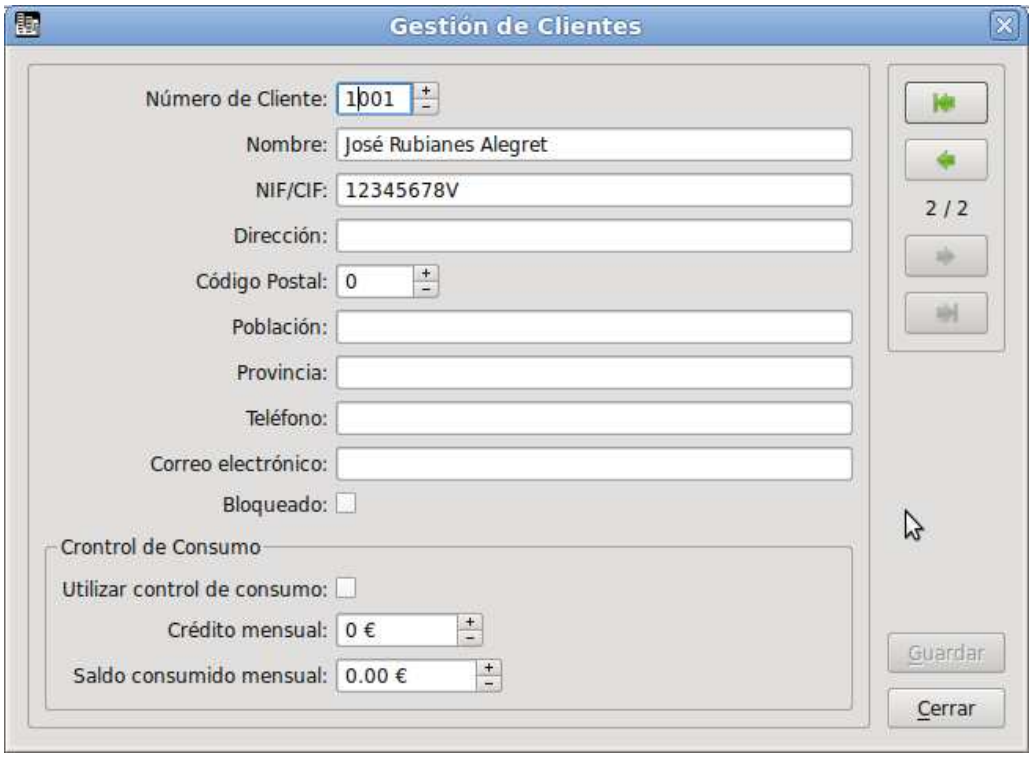

Figura 25. Dialogo de modificación de los datos de los clientes

El botón de Guardar, permanece desactivado mientras no se produzcan cambios en el formulario, una vez introducidos cambios en el formulario, se activa permitiendo guardarlos en la base de datos.

Si se introduce un dato no valido o después de escribir los datos no se guardan, a través de diálogos se le comunica al usuario para que haga las oportunas modificaciones.

Figura 26. Dialogo de advertencia de datos no validos introducidos

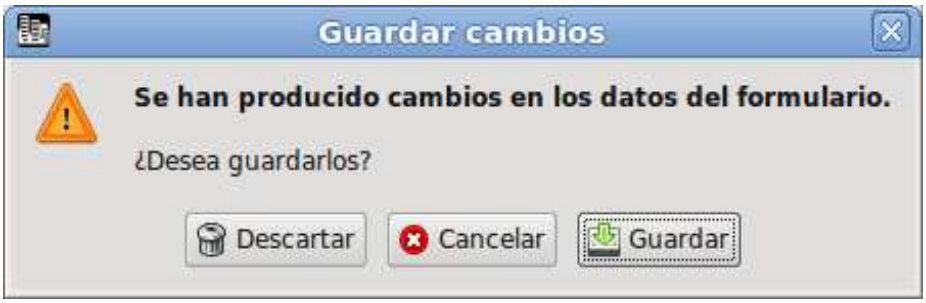

Figura 27. Dialogo de advertencia de cambios sin guardar

4.3.6 Productos

En la figura 28 se muestra el dialogo de modificación de los datos de los productos.

Figura 28. Dialogo de modificación de los datos de los productos

4.3.7 Tanques

En la figura 29 se muestra el dialogo de modificación de los datos de los tanques.

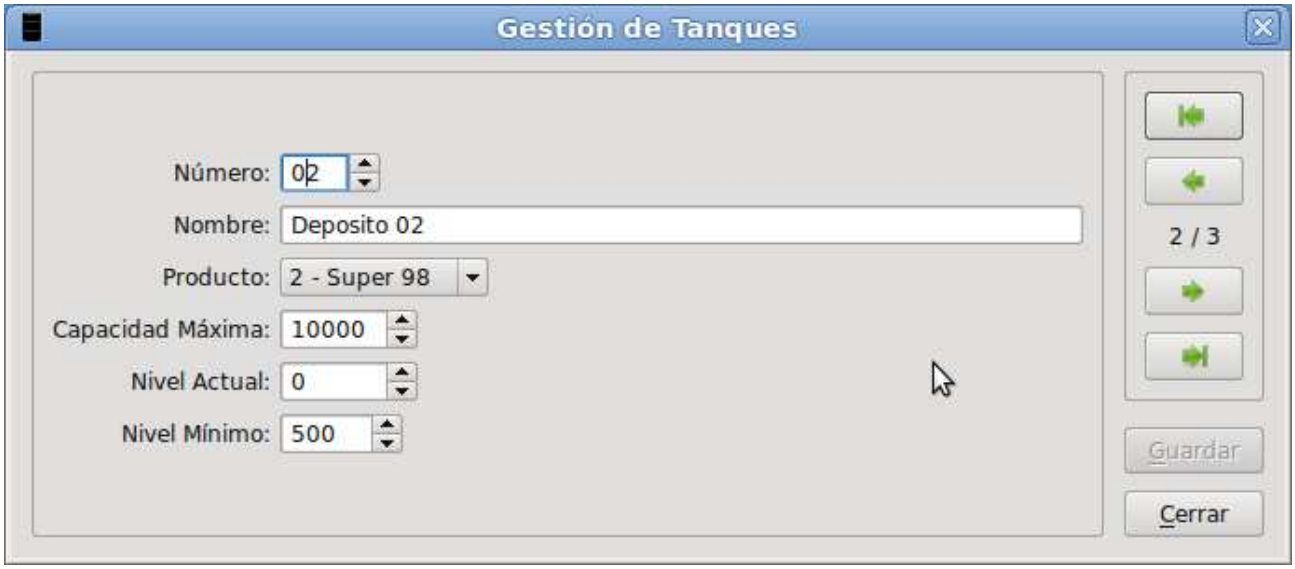

Figura 29. Dialogo de modificación de los datos de los tanques

4.3.8 Operaciones en los tanques

En la figura 30 se muestra el dialogo de modificación de las operaciones en los tanques.

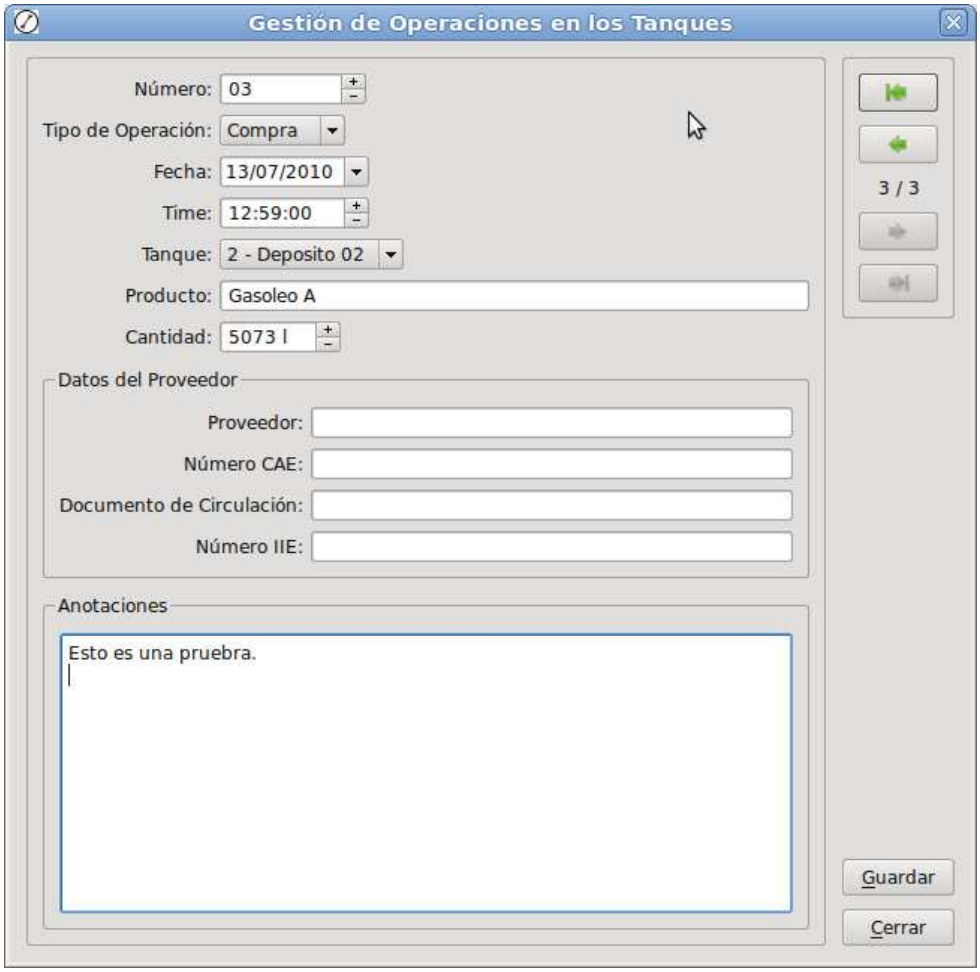

Figura 30. Dialogo de modificación de los datos de las operaciones en los tanques

4.3.9 Conductores

En la figura 31 se muestra el dialogo de modificación de los conductores.

Figura 31. Dialogo de modificación de los datos de los conductores

4.3.10 Vehículos

En la figura 32 se muestra el dialogo de modificación de los vehículos.

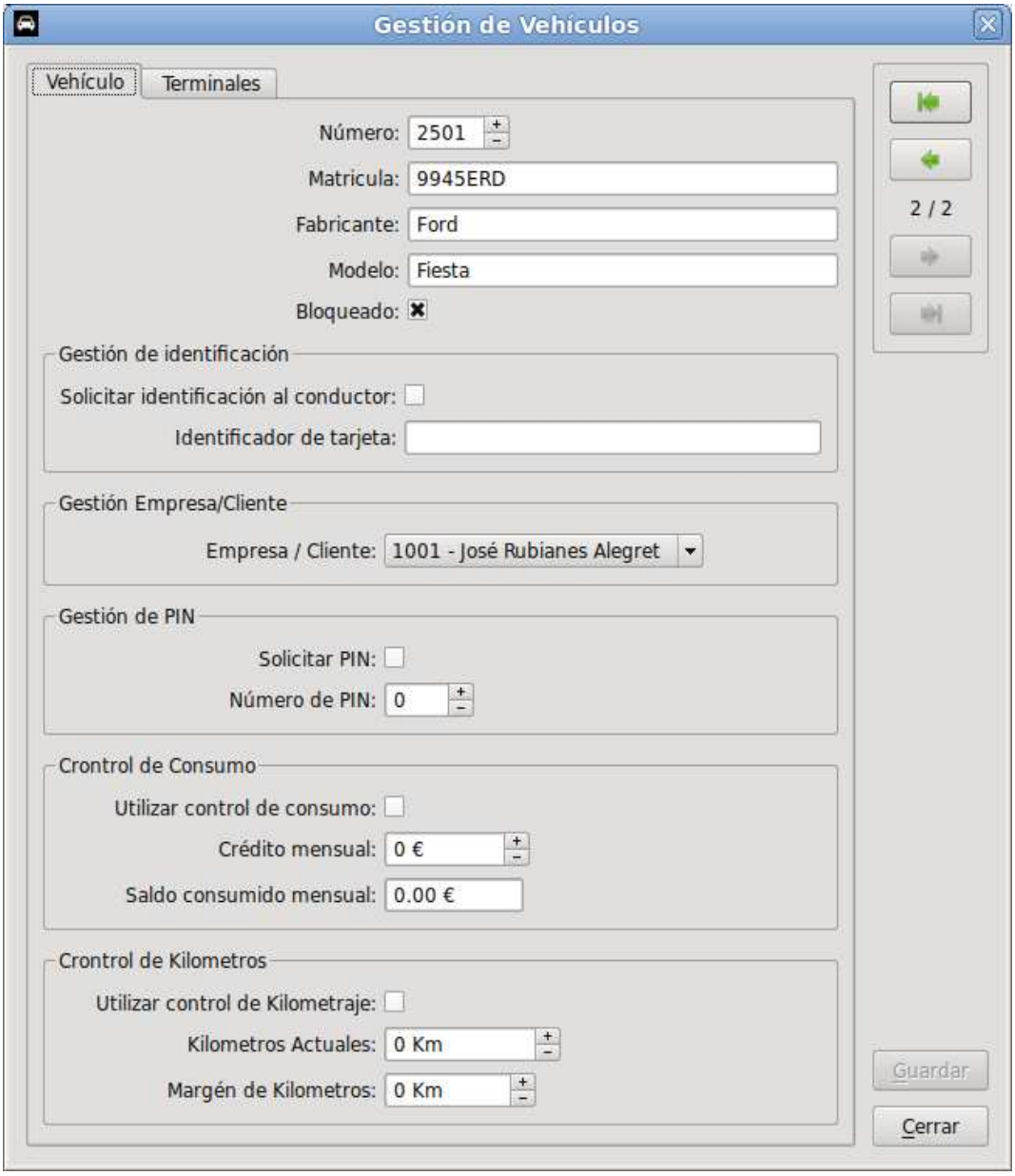

Figura 32. Dialogo de modificación de los datos de los vehículos

Como se puede observar en la figura 32, el dialogo de modificación de los datos de los vehículos esta estructurado en dos pestañas. En la primera pestaña (mostrada en la figura 32) se accede a los principales datos del vehículo. En la otra pestaña se accede a los datos de la gestión de terminales para el vehículo.

En la figura 33 se muestra la otra pestaña del dialogo de modificación de los datos de los vehículos.

Figura 33. Dialogo de modificación de los datos de los vehículos. Segunda pestaña

4.3.11 Terminales

En la figura 34 se muestra el dialogo de modificación de datos de los Terminales.

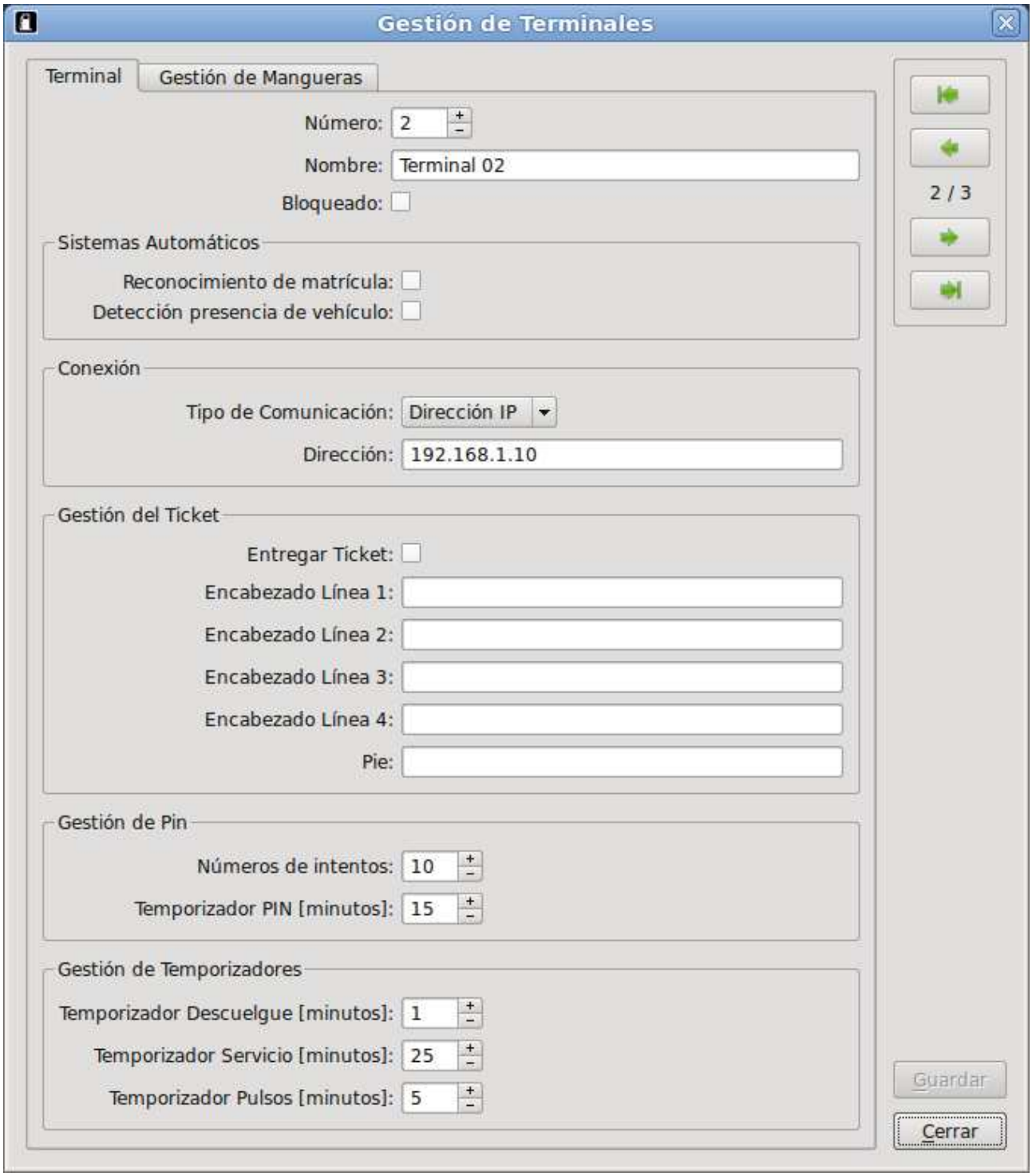

Figura 34. Dialogo de modificación de los datos de los terminales

Como se puede observar en la figura 34, este dialogo esta estructurado en dos pestañas. En la primera pestaña (mostrada en la figura 34) se accede a los principales datos del terminal. En la otra pestaña se accede a los datos de la gestión de mangueras del terminal.

En la figura 35 se muestra la otra pestaña del dialogo de modificación de los datos de los terminales.

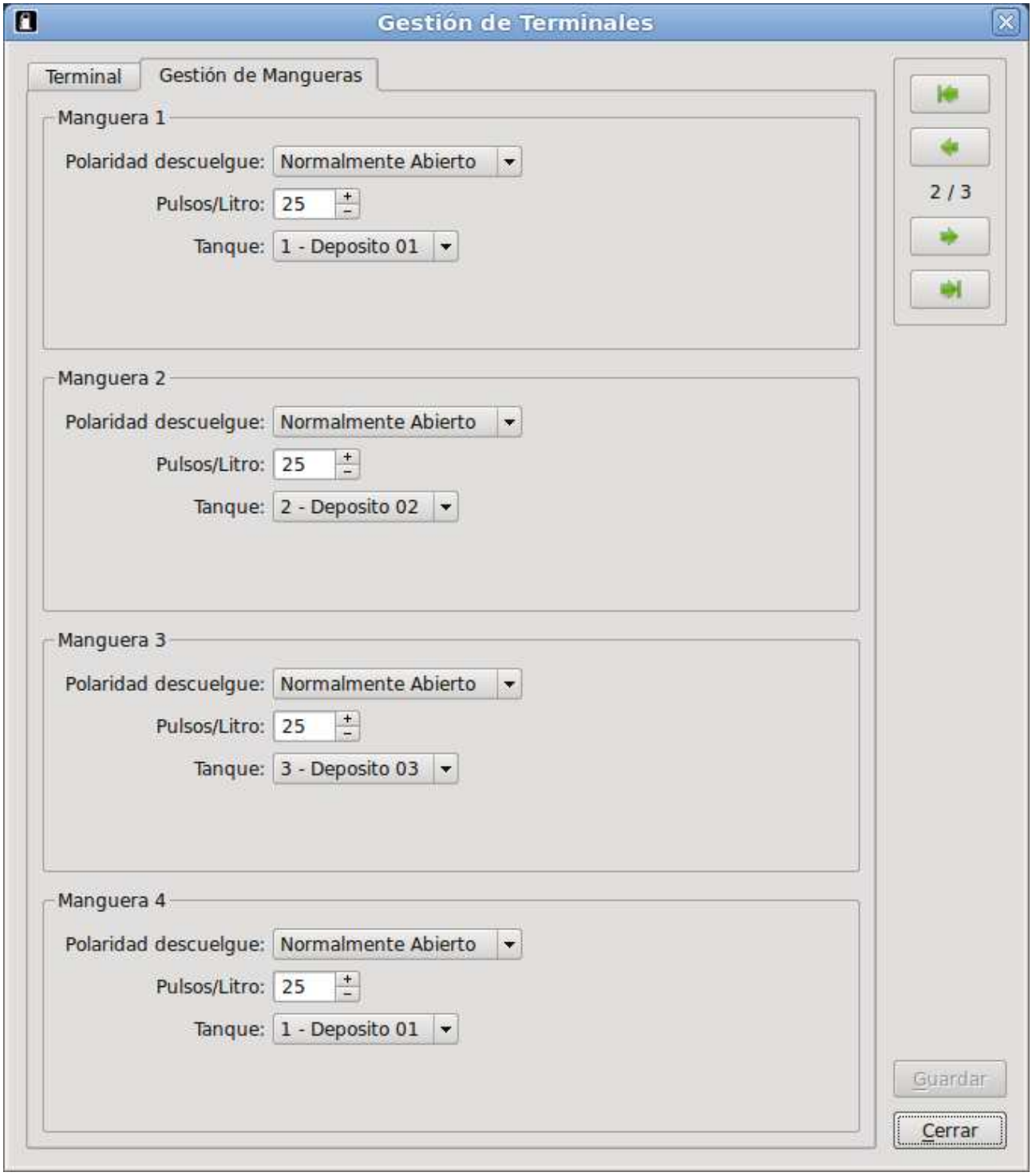

Figura 35. Dialogo de modificación de los datos de los terminales. Segunda pestaña

4.3.12 Servicios

En la figura 36 se muestra el dialogo de modificación de datos de los Servicios.

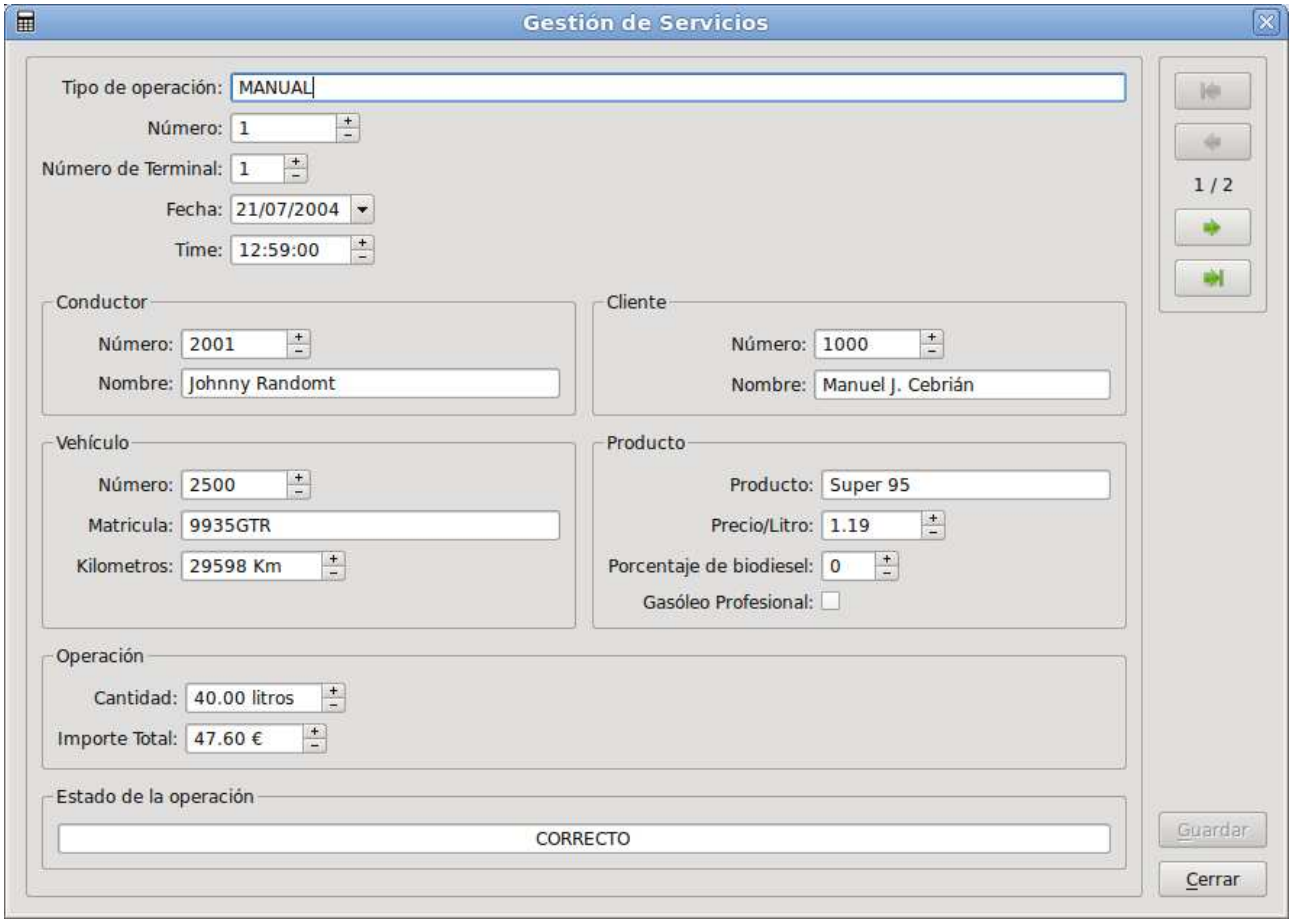

Figura 36. Dialogo de modificación de los datos de los servicios

Capítulo 5: Conclusiones

Recordando el desglose de los objetivos principales de este proyecto:

- Desarrollo de la base de datos
  - Diseño que permita bajo mantenimiento y un forma simple de realizar las copias de seguridad
  - Diseño redundante para evitar que un fallo cometido por el usuario final al introducir los datos pueda corromper el registro de operaciones.

- Desarrollo de el software de gestión
  - Diseño orientado a usuarios finales sin conocimientos de bases de datos
  - Interfaz visualmente atractiva e intuitiva.

Se puede afirmar que hemos logrado encontrar realizar un desarrollo de la base de datos que cumple los requisitos solicitados.

Respecto al software de gestión debido a la dificultad de mostrar el funcionamiento del software a través de capturas de pantalla, no se ha podido apreciar bien si se cumplian con los requisitos que se le solicitaban. Aunque en opinión de la empresa y del cliente que contrato este proyecto con la empresa si se cumplen las expectativas en cuanto a sencillez y facilidad de uso, pudiendo ser utilizado por usuarios finales sin conocimientos en bases de datos.

5.1 Líneas futuras

En cuanto a líneas futuras, en esta versión inicial del software no se ha implementado la gestión de impresión y las herramientas para generar informes y albaranes, así como el desarrollo de un instalador que facilite la instalación por parte del usuario final del software. Estos punto por que quedan en el desarrollo serán implementados durante la fase de pruebas del proyecto y bajo los criterios que indique el cliente.

Bibliografía

Paul Deitel, Harvey M. Deitel; C++ How to Program. Prentice Hall

Andrew Koenig, Barbara E. Moo; Accelerated C++: Practical Programming by Example. Addison-Wesley Professional

Julian Smart, Kevin Hock, Stefan Csomor; Cross-Platform GUI Programming with wxWidgets. Prentice Hall

Mark Summerfield; Advanced Qt Programming: Creating Great Software with C++ and Qt 4. Prentice Hall

Mark Summerfield; C++ GUI Programming with Qt 4. Prentice Hall

Neil Matthew, Richard Stones; Beginning Databases with PostgreSQL: From Novice to Professional. Apress

Grant Allen, Mike Owens; The Definitive Guide to SQLite. Apress

Seyed M.M. Tahaghoghi, Hugh Williams; Learning MySQL. O'Reilly.

Recursos web

http://www.mysql.com/
http://www.postgresql.org.pe/
http://es.wikipedia.org
http://qt.nokia.com/products/
http://doc.qt.nokia.com/
http://www.firebirdsql.org/
http://www.wxwidgets.org/
http://www.qtcentre.org/forum/
http://www.qtforum.org/index.html

Anexos A. Licencias

• Licencia LGPL

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:
    - 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
    - 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

- **Licencia BSD**

The BSD License

> The following is a BSD license template. To generate your own license, change the values of OWNER, ORGANIZATION and YEAR from their original values as given here, and substitute your own.
>
> *Note: You may optionally omit clause 3 and still be OSD-conformant.* On January 9th, 2008 the OSI Board approved the "Simplified BSD License" variant used by FreeBSD and others, which omits the final "no-endorsement" clause and is thus roughly equivalent to the MIT License.
>
> *Historical Note*: The original license used on BSD Unix had four clauses. The advertising clause (the third of four clauses) required you to acknowledge use of U.C. Berkeley code in your advertising of any product using that code. It was officially rescinded by the Director of the Office of Technology Licensing of the University of California on July 22nd, 1999. He states that clause 3 is "hereby deleted in its entirety." The four clause license has not been approved by OSI. The license below does not contain the advertising clause.
>
> This prelude is not part of the license.

<OWNER> = Regents of the University of California
<ORGANIZATION> = University of California, Berkeley
<YEAR> = 1998

In the original BSD license, both occurrences of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".

Here is the license template:

Copyright (c) <YEAR>, <OWNER>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,

DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Anexo B. Comandos SQL para la creación de la base de datos

```
SET CLIENT_ENCODING TO 'UTF8';
SET DATESTYLE TO 'European, ISO';
```

B.1 Creación tabla productos

```
CREATE TABLE products (
  product_id                serial,
  product_number            smallint       UNIQUE NOT NULL CHECK(product_number
> 0 AND product_number <= 100),
  product_name              varchar(128)    NOT NULL,
  product_literprice        numeric(7,3)    NOT NULL DEFAULT 0,
  product_biodiesel_percent  smallint       NOT NULL DEFAULT 0
CHECK(product_biodiesel_percent >= 0 AND product_biodiesel_percent <= 100),

  CONSTRAINT product_pk PRIMARY KEY(product_id)
);
```

B.2 Creación tabla clientes

```
CREATE TABLE clients (
  client_id                         serial,

  client_number                     smallint       UNIQUE NOT NULL
CHECK(client_number >= 1 AND client_number <= 10000),
  client_name                       varchar(128)    UNIQUE NOT NULL,
  client_nif_cif                    varchar(9)      UNIQUE NOT NULL,
  client_cae                        varchar(15)     UNIQUE DEFAULT NULL,

  client_address                    varchar(512)    DEFAULT NULL,
  client_postal_code                integer         NOT NULL DEFAULT 0
CHECK(client_postal_code >= 0 AND client_postal_code < 100000),
  client_town                       varchar(128)    DEFAULT NULL,
  client_province                   varchar(128)    DEFAULT NULL,
  client_phone                      varchar(32)     DEFAULT NULL,
  client_email                      varchar(128)    DEFAULT NULL,

  client_consumption_control        boolean         DEFAULT FALSE,
  client_moth_credit                integer         NOT NULL DEFAULT 0
CHECK(client_moth_credit >= 0 AND client_moth_credit < 1000000),
  client_actual_bill                integer         NOT NULL DEFAULT 0
CHECK(client_actual_bill >= 0 AND client_actual_bill < 1000000),

  client_freeze                     boolean         NOT NULL DEFAULT FALSE,

  CONSTRAINT client_pk PRIMARY KEY(client_id)
);
```

B.3 Creación tabla conductores

```
CREATE TABLE drivers (
  driver_id                         serial,

  driver_number                      smallint       UNIQUE NOT NULL CHECK
(driver_number > 0 AND driver_number < 10000),
  driver_name                       varchar(128)    UNIQUE NOT NULL,
  driver_nif                        varchar(9)      UNIQUE DEFAULT NULL,
  driver_id_card                    varchar(64)     UNIQUE DEFAULT NULL,

  driver_pin_required               boolean         NOT NULL DEFAULT TRUE,
  driver_pin_number                 smallint        NOT NULL DEFAULT 0
CHECK(driver_pin_number >= 0 AND driver_pin_number < 10000),

  driver_client_associate           smallint        NOT NULL,

  driver_freeze                     boolean         NOT NULL DEFAULT FALSE,

  CONSTRAINT driver_pk PRIMARY KEY(driver_id),
  CONSTRAINT driver_client_associate_fk  FOREIGN KEY(driver_client_associate)
REFERENCES clients(client_number)
);
```

B.4 Creación tabla tanques

```
CREATE TABLE tanks (
  tank_id                           serial,
  tank_number                       smallint       UNIQUE NOT NULL
CHECK(tank_number > 0 AND tank_number < 100),
  tank_name                         varchar(128)    UNIQUE NOT NULL,
  tank_product                      smallint       NOT NULL,
  tank_max_capacity                    integer         NOT NULL
CHECK(tank_max_capacity >= 0 AND tank_max_capacity < 1000000),
  tank_actual_level                 integer         NOT NULL CHECK(tank_actual_level
>= 0 AND tank_actual_level < 1000000),
  tank_min_level                    integer         NOT NULL CHECK(tank_min_level
>= 0 AND tank_min_level < 100000),

  CONSTRAINT tank_pk PRIMARY KEY(tank_id),
  CONSTRAINT tank_product_fk FOREIGN KEY(tank_product) REFERENCES
products(product_number)
);
```

B.5 Creación tabla operaciones en los tanques

```
CREATE TABLE tank_operations (
  tank_operation_id                    serial,
  tank_operation_number                  smallint       UNIQUE NOT NULL,
  tank_operation_type                   smallint       NOT NULL DEFAULT 0 CHECK
(tank_operation_type >= 0 AND tank_operation_type <= 1),  /* 0 -> Varillado, 1 ->
Compra */
  tank_operation_date                   date           NOT NULL,
  tank_operation_time                   time           NOT NULL,
  tank_operation_tank                   smallint       NOT NULL,
  tank_operation_product                 varchar(128)    NOT NULL,
  tank_operation_quantity                integer        NOT NULL
CHECK(tank_operation_quantity >= 0 AND tank_operation_quantity < 1000000),
  tank_operation_provider                varchar(256)    DEFAULT NULL,
  tank_operation_cae_provider             varchar(128)    DEFAULT NULL,
  tank_operation_doc_circulation_provider   varchar(128)    DEFAULT NULL,
  tank_operation_iie_provider            varchar(128)    DEFAULT NULL,
  tank_operation_description             varchar(2048)   DEFAULT NULL,

  CONSTRAINT tank_operation_pk PRIMARY KEY(tank_operation_id),
  CONSTRAINT tank_operation_tank_fk FOREIGN KEY(tank_operation_tank)
REFERENCES tanks(tank_number)
);
```

B.6 Creación tabla servicios

```
CREATE TABLE services (
  service_id                          serial,

  service_operation_type                 smallint       NOT NULL DEFAULT 0
CHECK(service_operation_type >= 0 AND service_operation_type <= 1), /* 0 -> auto
; 1 -> manual */
  service_operation_number               integer        UNIQUE NOT NULL
CHECK(service_operation_number >= 0),

  service_date                         date           NOT NULL,
  service_time                         time           NOT NULL,

  service_product_name                   varchar(128)    NOT NULL,
  service_liters_quantity                numeric(6,2)    NOT NULL,
  service_liter_price                   numeric(7,3)    NOT NULL,
  service_bill                         numeric(7,2)    NOT NULL DEFAULT 0,
  service_biodiesel_percent              smallint       NOT NULL DEFAULT 0
CHECK(service_biodiesel_percent >= 0 AND service_biodiesel_percent <= 100),

  service_client_number                  integer        NOT NULL
CHECK(service_client_number > 0 AND service_client_number < 10000),
  service_client_name                   varchar(128)    NOT NULL,

  service_driver_number                  integer        NOT NULL
CHECK(service_driver_number > 0 AND service_driver_number < 10000),
  service_driver_name                   varchar(128)    NOT NULL,
```

```
  service_vehicle_number                integer         NOT NULL
CHECK(service_vehicle_number > 0 AND service_vehicle_number < 10000),
  service_vehicle_number_plate            varchar(8)      NOT NULL,
  service_vehicle_actual_km               integer         NOT NULL DEFAULT 0
CHECK(service_vehicle_actual_km < 10000000),
  service_professional_diesel             boolean         NOT NULL DEFAULT FALSE,

  service_terminal_number                 smallint        NOT NULL DEFAULT 0
CHECK(service_terminal_number > 0 AND service_terminal_number < 100),

  service_manual_modified                 boolean         NOT NULL DEFAULT FALSE,

  service_status                          smallint        NOT NULL DEFAULT 0
CHECK(service_status >= 0 AND service_status <= 9),

  CONSTRAINT service_pk PRIMARY KEY(service_id)
);
```

B.7 Creación tabla terminales

```
CREATE TABLE terminals (

  terminal_id                           serial,
  terminal_number                       smallint        UNIQUE NOT NULL
CHECK(terminal_number > 0 AND terminal_number < 100),
  terminal_name                         varchar(128)    UNIQUE NOT NULL,

  terminal_connection_type              smallint        NOT NULL
CHECK(terminal_connection_type >= 0 AND terminal_connection_type <= 1), /* 0 ->
COM, 1 -> IP */
  terminal_connection_port              varchar(64),

  terminal_detection_vehicle_number_plate   boolean         NOT NULL DEFAULT
FALSE,
  terminal_detection_vehicle_presence       boolean         NOT NULL DEFAULT FALSE,

  terminal_ticket_flag                  boolean         NOT NULL DEFAULT FALSE,

  terminal_ticket_header_one            varchar(128)    DEFAULT NULL,
  terminal_ticket_header_two            varchar(128)    DEFAULT NULL,
  terminal_ticket_header_three          varchar(128)    DEFAULT NULL,
  terminal_ticket_header_four           varchar(128)    DEFAULT NULL,
  terminal_ticket_ending                varchar(128)    DEFAULT NULL,

  terminal_retry_pin_number             smallint        NOT NULL DEFAULT 0
CHECK(terminal_retry_pin_number >= 0 AND terminal_retry_pin_number < 11),
  terminal_timeout_pin_number           smallint        NOT NULL DEFAULT 0
CHECK(terminal_timeout_pin_number >= 0 AND terminal_timeout_pin_number <=
100),

  terminal_time_minutes_hang_off        smallint        NOT NULL DEFAULT 0
CHECK(terminal_time_minutes_hang_off >= 0 AND terminal_time_minutes_hang_off
< 100),
```

```
  terminal_time_minutes_service             smallint        NOT NULL DEFAULT 0
CHECK(terminal_time_minutes_service >= 0 AND terminal_time_minutes_service <
100),
  terminal_time_minutes_without_pulses      smallint        NOT NULL DEFAULT 0
CHECK(terminal_time_minutes_without_pulses >= 0 AND
terminal_time_minutes_without_pulses < 100),

  terminal_hose_pipe_one_pulses_liter       smallint        NOT NULL DEFAULT 1
CHECK(terminal_hose_pipe_one_pulses_liter > 0 AND
terminal_hose_pipe_one_pulses_liter < 1001),
  terminal_hose_pipe_one_polarity           smallint        NOT NULL DEFAULT 0
CHECK(terminal_hose_pipe_one_polarity >= 0 AND terminal_hose_pipe_one_polarity
<= 1),
  terminal_hose_pipe_one_tank               smallint        DEFAULT NULL,

  terminal_hose_pipe_two_pulses_liter       smallint        NOT NULL DEFAULT 1
CHECK(terminal_hose_pipe_two_pulses_liter > 0 AND
terminal_hose_pipe_two_pulses_liter < 1001),
  terminal_hose_pipe_two_polarity           smallint        NOT NULL DEFAULT 0
CHECK(terminal_hose_pipe_two_polarity >= 0 AND terminal_hose_pipe_two_polarity
<= 1),
  terminal_hose_pipe_two_tank               smallint        DEFAULT NULL,

  terminal_hose_pipe_three_pulses_liter     smallint        NOT NULL DEFAULT 1
CHECK(terminal_hose_pipe_three_pulses_liter > 0 AND
terminal_hose_pipe_three_pulses_liter < 1001),
  terminal_hose_pipe_three_polarity         smallint        NOT NULL DEFAULT 0
CHECK(terminal_hose_pipe_three_polarity >= 0 AND
terminal_hose_pipe_three_polarity <= 1),
  terminal_hose_pipe_three_tank             smallint        DEFAULT NULL,

  terminal_hose_pipe_four_pulses_liter      smallint        NOT NULL DEFAULT 1
CHECK(terminal_hose_pipe_four_pulses_liter > 0 AND
terminal_hose_pipe_four_pulses_liter < 1001),
  terminal_hose_pipe_four_polarity          smallint        NOT NULL DEFAULT 0
CHECK(terminal_hose_pipe_four_polarity >= 0 AND terminal_hose_pipe_four_polarity
<= 1),
  terminal_hose_pipe_four_tank              smallint        DEFAULT NULL,

  terminal_freeze                           boolean         NOT NULL DEFAULT FALSE,

  CONSTRAINT terminal_pk PRIMARY KEY(terminal_id),
  CONSTRAINT terminal_hose_pipe_one_tank_fk FOREIGN
KEY(terminal_hose_pipe_one_tank) REFERENCES tanks(tank_number),
  CONSTRAINT terminal_hose_pipe_two_tank_fk FOREIGN
KEY(terminal_hose_pipe_two_tank) REFERENCES tanks(tank_number),
  CONSTRAINT terminal_hose_pipe_three_tank_fk FOREIGN
KEY(terminal_hose_pipe_three_tank) REFERENCES tanks(tank_number),
  CONSTRAINT terminal_hose_pipe_four_tank_fk FOREIGN
KEY(terminal_hose_pipe_four_tank) REFERENCES tanks(tank_number)
);
```

B.8 Creación tabla vehículos

```
CREATE TABLE vehicles (
  vehicle_id                        serial,
  vehicle_number                    smallint       UNIQUE NOT NULL CHECK
(vehicle_number > 0 AND vehicle_number < 10000),
  vehicle_manufacturer              varchar(64)    DEFAULT NULL,
  vehicle_model                     varchar(64)    DEFAULT NULL,
  vehicle_number_plate              varchar(8)     UNIQUE NOT NULL,

  vehicle_id_card                   varchar(64)    UNIQUE DEFAULT NULL,

  vehicle_pin_required              boolean        NOT NULL DEFAULT FALSE,
  vehicle_pin_number                smallint       NOT NULL DEFAULT 0
CHECK(vehicle_pin_number >= 0 AND vehicle_pin_number < 10000),

  vehicle_consumption_control       boolean        DEFAULT FALSE,
  vehicle_moth_credit               integer        NOT NULL DEFAULT 0
CHECK(vehicle_moth_credit >= 0 AND vehicle_moth_credit < 1000000),
  vehicle_actual_bill               numeric(8,2)   NOT NULL DEFAULT 0
CHECK(vehicle_actual_bill >= 0 AND vehicle_actual_bill < 1000000),

  vehicle_km_control                boolean        NOT NULL DEFAULT FALSE,
  vehicle_actual_km                 integer        NOT NULL DEFAULT 0
CHECK(vehicle_actual_km >= 0 AND vehicle_actual_km < 10000001),
  vehicle_km_margin_of_safety       integer        NOT NULL DEFAULT 0
CHECK(vehicle_km_margin_of_safety >= 0 AND vehicle_km_margin_of_safety <
100000),

  vehicle_professional_diesel       boolean        NOT NULL DEFAULT FALSE,

  vehicle_driver_id_required        boolean        NOT NULL DEFAULT FALSE,

  vehicle_client_associate          smallint       NOT NULL,

  vehicle_product_one               smallint       DEFAULT NULL,
  vehicle_product_two               smallint       DEFAULT NULL,
  vehicle_product_three             smallint       DEFAULT NULL,
  vehicle_product_four              smallint       DEFAULT NULL,

  vehicle_terminal_authorization    boolean        DEFAULT FALSE,
  vehicle_terminal_one              smallint       DEFAULT NULL,
  vehicle_terminal_two              smallint       DEFAULT NULL,
  vehicle_terminal_three            smallint       DEFAULT NULL,
  vehicle_terminal_four             smallint       DEFAULT NULL,
  vehicle_terminal_five             smallint       DEFAULT NULL,
  vehicle_terminal_six              smallint       DEFAULT NULL,
  vehicle_terminal_seven            smallint       DEFAULT NULL,
  vehicle_terminal_eight            smallint       DEFAULT NULL,
  vehicle_terminal_nine             smallint       DEFAULT NULL,
  vehicle_terminal_ten              smallint       DEFAULT NULL,


  vehicle_freeze                    boolean        NOT NULL DEFAULT FALSE,
```

```
  CONSTRAINT vehicle_pk PRIMARY KEY(vehicle_id),
  CONSTRAINT vehicle_client_fk        FOREIGN KEY(vehicle_client_associate)
REFERENCES clients(client_number),
  CONSTRAINT vehicle_product_one_fk   FOREIGN KEY(vehicle_product_one)
REFERENCES products(product_number),
  CONSTRAINT vehicle_product_two_fk   FOREIGN KEY(vehicle_product_two)
REFERENCES products(product_number),
  CONSTRAINT vehicle_product_three_fk FOREIGN KEY(vehicle_product_three)
REFERENCES products(product_number),
  CONSTRAINT vehicle_product_four_fk  FOREIGN KEY(vehicle_product_four)
REFERENCES products(product_number),

  CONSTRAINT vehicle_terminal_one_fk   FOREIGN KEY(vehicle_terminal_one)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_two_fk   FOREIGN KEY(vehicle_terminal_two)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_three_fk FOREIGN KEY(vehicle_terminal_three)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_four_fk  FOREIGN KEY(vehicle_terminal_four)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_five_fk  FOREIGN KEY(vehicle_terminal_five)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_six_fk   FOREIGN KEY(vehicle_terminal_six)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_seven_fk FOREIGN KEY(vehicle_terminal_seven)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_eight_fk FOREIGN KEY(vehicle_terminal_eight)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_nine_fk  FOREIGN KEY(vehicle_terminal_nine)
REFERENCES terminals(terminal_number),
  CONSTRAINT vehicle_terminal_ten_fk   FOREIGN KEY(vehicle_terminal_ten)
REFERENCES terminals(terminal_number)
);
```