ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

INFORME DE MATERIA DE GRADUACIÓN

"BASE DE DATOS CENTRALIZADA PARA SISTEMAS DE SEGURIDAD"

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN ESPECIALIZACIÓN SISTEMAS TECNOLÓGICOS

PRESENTADA POR:

BRUNO OSWALDO MACIAS VELASCO
ROLANDO FERNANDO QUIJIJE IDUARTE

GUAYAQUIL - ECUADOR

2009

AGRADECIMIENTO

A Dios, a nuestros padres, a nuestros maestros y a todas aquellas personas que siempre confiaron en nosotros.

DEDICATORIA

Dedico este proyecto a mi familia.
Bruno Macías

A la mujer que amo, y que amaré por siempre.
A Vanessa mi única Princesa.

Rolando Quijije

TRIBUNAL DE GRADO

Ing. Hugo Villavicencio	Dr. Ing. Carlos Valdivieso A.
MIEMBRO DEL TRIBUNAL	DIRECTOR DE TESIS

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, nos corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Bruno Oswaldo Macías Velasco

Rolando Fernando Quijije Iduarte

RESUMEN

Las Base de Datos Centralizadas son muy utilizadas debido a las grandes garantías y bondades que proporcionan, el hecho de que permiten almacenar los datos en un solo sitio, conservando su integridad y consistencia es sin duda una de sus más importantes características.

El presente Proyecto de tesis tiene como objetivo principal diseñar e implementar una Base de Datos Centralizada para administrar los datos capturados por ciertos Sistema de Seguridades, esta Base de Datos será administrada por una interfaz gráfica desarrollada en LabVIEW.

La organización de este documento se detalla a continuación:

El primer capítulo incluye los antecedentes, identificación del problema, el alcance, los objetivos, las ventajas y limitantes de este Proyecto además de un breve estudio de una soluciones similar.

En el segundo capítulo de este Proyecto se detalla toda la información referente a las Bases de Datos, y sus fortalezas, así mismo el modelo a seguir el cual es el de tener una Base de Datos Centralizada, el modelo Cliente - Servidor. En este capítulo también se explicarán las bondades de LabVIEW, la forma de programar en lenguaje G, como una herramienta de

programación gráfica y su forma de interactuar con las Bases de Datos, todo esto para lograr un mayor entendimiento de la base teórica de las herramientas y modelos usados. Además se incluye información adicional de MySQL que será el motor de Base de Datos que usaremos, las librerías de manejo de Base de Datos e Imágenes de LabVIEW.

El tercer capítulo explica el diseño e implementación de la interfaz gráfica en LabVIEW, y también el de la Base de Datos Centralizada.

En el cuarto y último capítulo se expone la simulación de la interfaz que interactúa con la Base de Datos, las pruebas y los resultados obtenidos.

INDICE GENERAL

INDICE DE FIGURAS

INTRODUCCIÓN

Las primeras redes informáticas se desarrollaron alrededor de un equipo central llamado "sistema central" u “ordenador central”.

Por este motivo el sistema central es un ordenador central de alto rendimiento que controla las sesiones del usuario en los diferentes terminales que están conectados a él. Gracias a esta arquitectura puede consolidarse, es decir, administrar de manera centralizada todas las aplicaciones de un sistema.

Sin embargo, en un modelo centralizado, el rendimiento del sistema depende íntegramente de la capacidad de procesamiento del ordenador central.

Las Bases de Datos son una parte esencial de todo sistema de información, para nuestro caso diseñaremos e implementaremos una Base de Datos Centralizada capaz de administrar, desde una interfaz gráfica creada en LabVIEW los datos de varios Sistemas de Seguridades, todos estos sistemas son Proyectos de la Materia de Graduación MICROCONTROLADORES AVANZADOS.

En el mercado global existen muchos sistemas de seguridades, por citar un ejemplo tenemos los CCTV circuitos cerrado de televisión los cuales mediante cámaras de vigilancia monitorean lugares estratégicos en tiempo real, las imágenes obtenidas por estas cámaras son analizadas desde terminales por uno o varios operadores, la información de estas cámaras de video son almacenadas en cintas magnéticas u otros medios de almacenamiento, en algunos casos no se almacena esta información y estos sistemas de seguridades se convierten en sólo de monitoreo lo cual no es muy útil en el caso que se quiera revisar algún evento en el pasado que haya sido observado por las cámaras

Otro de los ejemplos que podemos mencionar son los sistemas de seguridad para control de salida de los artículos en un supermercado estos sistemas son de alerta aunque también existen los CCTV, en este caso cada artículo del supermercado tiene adherido un dispositivo de seguridad el cual si no ha sido removido o desactivado, por el cajero del supermercado, al salir del supermercado activa el Sistema de Seguridad del establecimiento alertando al personal de seguridad.

El Motor de Base de Datos a utilizar es MySQL versión 5.0 y la interfaz gráfica de administración será desarrollada en LabVIEW 8.5 de National Instrument.

CAPÍTULO I

ALCANCE DEL PROYECTO Y SOLUCIONES SIMILARES

1.1 Identificación del Problema

El problema principal que se tiene es que en un Sistema de Seguridades muchas veces los datos no son almacenados en ningún lugar debido a que estos sistemas son solo de monitoreo o de alerta, en otros casos si se almacenan datos pero cada terminal de monitoreo del Sistema de Seguridades tiene su propio medio de almacenamiento o Base de Datos localmente, esto significa que si se quiere revisar algún evento o acción que el Sistema de Seguridad haya detectado o ejecutado respectivamente tenemos que dirigirnos al PC terminal donde se almacenaron esos datos.

Al tener una Base de Datos Centralizada este problema es solucionado ya que todas las PC's terminales accesarían a una sola Base de Datos la cual estaría ubicada estratégicamente en un lugar con las respectivas medidas seguridad que amerita este tipo de medio de almacenamiento. Otro problema que se soluciona al tener una Base de Datos Centralizada es el del respaldar los datos, en lugar de respaldar muchas Bases de

Datos Locales ahora solo se hace el respaldo de una sola Base de Datos.

1.2 Antecedentes

Aprovechando la versatilidad de las herramientas actuales de programación, específicamente la programación en LabVIEW aprendida en esta Materia de Graduación, y el conocimiento adquirido durante nuestra vida estudiantil en diseño y manejo de Bases de Datos, se construirá una Base de Datos Centralizada la cual almacenará datos de varios Sistemas de Seguridades, esta Base de Datos será administrada por un programa creado en LabVIEW.

Hace algunos años el implementar una Base de Datos Centralizada era visto como una implementación muy costosa debido a los costos elevados de hardware, pero dado que actualmente los precios del hardware tienden a bajar considerablemente debido a su rápida evolución y producción en masa esto ya no es un problema tan grave.

1.3 Alcance del Proyecto

Administrar una Base de Datos Centralizada mediante una interfaz gráfica desarrollada en LabVIEW.

Específicamente nos concentramos en una Base de Datos Centralizada que almacenara los datos capturados por algunos Sistemas de Seguridades, estos Sistemas mediante sensores, actuadores y cámaras de video capturaran datos de importancia que serán transmitidos vía Ethernet para luego ser almacenados en la Base de Datos Centralizada.

La lista de los Sistemas de Seguridades soportados por la Base de Datos Centralizada es la siguiente:

- Sistema de Seguridad Domiciliaria
- Sistema de Seguridad Industrial
- Sistema de Seguridad para un Laboratorio
- Sistema de Seguridad para equipos de Laboratorio
- Sistema de Seguridad para Vehículos
- Sistema para Control de Personal
- Sistema de Seguridad para entrada principal y corredores

Todos estos Sistemas de Seguridades son Proyectos de la Materia de Graduación “MICROCONTROLADORES AVANZADOS”

1.4 Objetivos

El presente proyecto tiene la meta de lograr los siguientes objetivos:

1.4.1 Objetivo general

- Diseñar e implementar una Base de Datos Centralizada para administrar los datos capturados por Sistemas de Seguridades mediante una interfaz desarrollada en LabVIEW.

1.4.2 Objetivos específicos

- Tener una Base de Datos Centralizada, flexible pero al mismo tiempo confiable y robusta, para el uso de los diferentes Sistemas de Seguridades.
- Facilitar consultas como ayuda para el análisis de los datos almacenados.
- Diseñar e implementar una interfaz gráfica en LabVIEW para la administración de una Base de Datos.

1.5 Limitaciones del Proyecto

- La metodología de desarrollo de LabVIEW usando iconos lo cual se conoce como lenguaje G, cambia la forma tradicional de programación mediante líneas de código como se lo hace en C#, C++, JAVA, etc. Esto trae un fuerte cambio de costumbre en programación.
- El modelo de la Base de Datos obtenido, está enfocado para soportar los proyectos de esta materia de Graduación, puede que el mismo modelo soporte otros sistemas de seguridades pero no se da garantía de ello.

1.6 Ventajas del proyecto

- A pesar de que estamos sujetos al uso de LabVIEW para el desarrollo de la interfaz gráfica que administrara una Base de Datos centralizada, cabe destacar que LabVIEW nos ofrece una gran gama de “iconos” para programar cualquier tipo de aplicación.
- Para sistemas de seguridades diferentes a los soportados y donde usen sensores, actuadores y agentes que la Base de Datos Centralizada soporta, se puede dar garantía que esos sistemas de seguridades podrán ser soportados.

1.7 Estudio de Soluciones Similares

Una solución similar la cual analizaremos es la CENTRALIZACIÓN DEL SISTEMA DE INFORMACIÓN DE ATENCIÓN PRIMARIA: HISTORIA CLÍNICA ELECTRÓNICA INTEGRADA.

Esta solución nace en España – Madrid donde la Consejería de Sanidad, consciente de las crecientes necesidades y demandas de sus Centros de Salud, trabajaron para cumplir el objetivo prioritario de poner al alcance de los más de 6 millones de ciudadanos y de los más de 12 mil profesionales de la red asistencial de Atención Primaria de la Comunidad de Madrid toda los datos del historial médico de los pacientes disponibles en el ámbito sanitario con plenas garantías de calidad y seguridad. Esto llevo a la Integración de los Sistemas de Información de la Sanidad de la Comunidad de Madrid, para lo cual se estableció como paso fundamental la Centralización del Sistema que almacena los datos de Atención Primaria enmarcados en el Plan Estratégico de este organismo.

En la situación anterior de sistemas distribuidos existía un servidor en cada centro de salud o consultorio, lo que supone la existencia de más de 400 equipos, (246 Centros de Salud y 167 Consultorios), que albergan los datos clínicos y administrativos de los pacientes asignados a cada centro y donde las tareas para garantizar la integridad, disponibilidad, conservación y protección de datos se realizaba de forma local.

Esta solución centralizada, basada en una única Base de Datos, que integra la totalidad de los Datos, es accesible desde cualquier equipo del centro de Atención Primaria y da respuesta a las necesidades actuales y futuras, garantizando, además, la continuidad del planteamiento funcional y operativo, con objeto de limitar los riesgos del cambio. De esta forma, se avanza decididamente hacia la Historia Clínica Única Electrónica HCUE.

Concretamente, los objetivos estratégicos de esta solución similar son:

- Situar al ciudadano en el foco de máximo interés:

- Integrando y consolidando la información.
- Posibilitando el acceso del paciente a su propia información.
- Mejorando el circuito asistencial (integrando ámbitos y niveles).
- Facilitando el acceso a servicios integrados: Cita centralizada, Receta electrónica.
- Garantizar la integración con otros Sistemas de Información.
- Cumplimiento de las normas básicas de respecto a protección de datos personales.
- Evolucionar tecnológicamente y simplificar el mantenimiento, la gestión y la evolución.

Como paso previo a la implantación de esta solución, fue necesaria la definición de un **marco tecnológico** perfectamente estructurado que garantice la incorporación paulatina de todos los Sistemas de Información del sistema sanitario. Este marco tecnológico tiene como pilares fundamentales:

- La **mejora de la Red de Comunicaciones**, que dispondrá antes de la puesta en marcha del nuevo sistema, de enlaces de fibra óptica para todos los Centros de Salud de la Comunidad de Madrid.

- La **ampliación de los Centros de Procesos de Datos**, dotándolos de un equipamiento adecuadamente dimensionado, con redundancia de los elementos críticos aumentando la fiabilidad y disponibilidad de los sistemas.
- La creación de los **mecanismos de Gestión de Identidades**, que faciliten el acceso de los profesionales a los diferentes sistemas, así como la administración de los usuarios.
- La disponibilidad de los datos comunes a los diferentes Sistemas de Información de forma compartida en un **Centro de Información Básica Estratégica para los Entornos Sanitarios (CIBELES)**. Algunos de los datos que residirán en este sistema son: Población, Mapa Sanitario, Centros, Servicios, Profesionales, Nomenclátor, etc.

Por otra parte, para la consecución de los objetivos perseguidos, se establecieron un conjunto de **principios funcionales** que rigen esta solución, y los cuales son:

- Continuidad del planteamiento funcional y operativo del sistema de partida, unido a un proceso de renovación tecnológica que facilite el resto de los objetivos del proyecto.
- Garantía de integridad, confidencialidad y disponibilidad de la Información.
- Homogeneidad de servicios de los Centros de Atención Primaria.

- Utilización de mejores prácticas.
- Procesos Asistenciales integrados y comunes.
- Estructuración organizada de los servicios informáticos.
- Flexibilidad total de adaptación ante nuevas necesidades o cambios en los Centros de Salud de la Ciudad de Madrid.
- Orientación hacia una **única Historia Clínica Electrónica** por paciente en el Sistema Sanitario.
- Mejora Continua de la solución partiendo de la solución actual y minimizando el impacto del cambio.

Luego de revisar la CENTRALIZACIÓN DEL SISTEMA DE INFORMACIÓN DE ATENCIÓN PRIMARIA: HISTORIA CLÍNICA ELECTRÓNICA INTEGRADA podemos apreciar que la Centralización de la Base de Datos mejora muchos aspectos como lo son el mantenimiento de los datos las seguridades y acceso a los datos, además que el tener el HCUE se elimina la necesidad de tener Bases de Datos locales en cada Centro de Salud.
A pesar que esta solución no es un Sistema de Seguridades se puede apreciar claramente las ventajas de tener una Base de Datos Centralizada.

De igual manera también podemos mencionar otra solución similar la cual en cambio monitorea sucesos en redes informáticas. GFI Event Manager es la otra solución que revisaremos brevemente. Esta aplicación también posee una Base de Datos Centralizada donde los Datos son almacenados para luego ser accesados cuando se requiera hacer una consulta histórica o algún reporte.

Dado que los registros de sucesos son una valiosa herramienta para monitorizar la seguridad y el rendimiento de la red que son a menudo infrautilizados debido a su complejidad y volumen. Como las organizaciones crecen, necesitan una aproximación más estructurada junto con la administración y retención de registros de sucesos. Una reciente encuesta llevada a cabo por SANS Institute descubrió que el 44% de los administradores de sistemas no mantiene los logs más de un mes. La apropiada administración de registros de sucesos le ayuda a cumplir varios objetivos, incluyendo:

- Sistema de información y seguridad de red: detecta intrusos y brechas de seguridad
- Monitorización de la salud del sistema: monitoriza constantemente sus terminales.
- Cumplimiento legal y regulador (SOX, PCI DSS, HIPAA): una ayuda para cumplir con las regulaciones.
- Investigación forense: un punto de referencia cuando algo va mal.

GFI EventsManager recoge datos de todos los dispositivos que utilizan registros de sucesos Windows, W3C, y Syslog y aplica las mejores reglas y filtros de la industria para identificar los datos clave. Esto le permite hacer un seguimiento cuando el personal abusa, descuelga el teléfono para llamar a casa, enciende su PC, qué hacen en sus PCs y a qué archivos acceden durante su trabajo diario. GFI EventsManager también le proporciona alertas en tiempo real cuando aparecen sucesos críticos de seguridad y de sistema y sugiere acciones correctoras.

Entre los beneficios que oferta GFI Event Manager tenemos los siguientes:

- Centraliza los sucesos Syslog, W3C y Windows generados por cortafuegos, servidores, enrutadores, switches, sistemas telefónicos, PCs y más
- Configuración por asistentes que simplifica la operación y mantenimiento del usuario final
- Rendimiento sin igual de escaneo de sucesos de hasta 6 millones de sucesos por hora
- Reglas pre configuradas de procesamiento de sucesos para una eficaz clasificación y administración de sucesos instantánea
- Monitorización y alertas automatizadas de la actividad de sucesos 24/7
- Potente generación de informes para la monitorización eficaz de la actividad de red y para un inmediato ROI.

Utilizando el generador de informes de GFI EventsManager, usted puede crear o personalizar informes incluyendo informes estándar tales como:

- Informes de uso de cuentas

- Informes administrativos de cuentas
- Informes de cambios de directiva
- Informes de acceso a objetos
- Informes administrativos de aplicaciones
- Informes de servidor de impresión
- Informes de sistema del registro de sucesos Windows
- Informes de tendencias de eventos

Al igual que la aplicación que desarrollaremos en LabVIEW notamos que el GFI Event Manager tiene una Base de Datos Centralizada y una interfaz de usuario que permite consultar ciertos tipos de reportes.

CAPÍTULO II

BASES DE DATOS CENTRALIZADAS Y LABVIEW

2.1 Introducción a las Bases de Datos Centralizadas

Una Base de Datos Centralizada es una Base de Datos que almacena grandes cantidades de datos en un solo punto, estos datos son accesados por todos los usuarios de los diferentes sistemas que interactúan con ella.

Por sus características y seguridades una Base de Datos es uno de los elementos primordiales en todo sistema de información, a diferencia de otros medios de almacenamientos, como lo son archivos, las Bases de Datos poseen estructuras y componentes internos que organizan de una manera eficaz y eficiente los datos, el acceso a los datos almacenados en una Base de Datos se lo realiza mediante el lenguaje SQL (Structure Query Languaje).

Las Bases de Datos son administradas por un Motor de Base de Datos o Sistema de Administración de Bases de Datos DBMS por sus siglas en ingles, estos sistemas manejan todos los requerimientos que hacen los usuarios a una Base de Datos.

Los usuarios de una Base de Datos se dividen por lo general en tres grandes grupos:

- Los usuarios finales: son quienes interactúan con la Base de Datos desde sus terminales de trabajo, por lo general los usuarios finales son quienes ingresan, consultan, modifican y eliminan registros mediante programas o sistemas instalados en sus terminales de trabajo.
- Los programadores: son los responsables de escribir los programas o sistemas que interactúan con la Base de Datos en algún lenguaje de programación.
- Los Administradores de Bases de Datos (DBA): son los encargados de supervisar la consistencia e integridad de los datos almacenados en una Base de Datos, también son los responsables de migrar datos y hacer respaldos para evitar perdida de datos.

Las Bases de Datos están compuestas por objetos, entre ellos tenemos las entidades o tablas, las tablas están asociadas entre si mediante relaciones y de estas asociaciones nace lo que se conoce como el Modelo Entidad Relación que no es nada más que una representación gráfica de cómo se relacionan las tablas de una Base de Datos.

Cada entidad o tabla posee campos de diferentes tipos de datos y un campo en especial que se conoce como "*primary key*" de la tabla, los valores de este campo no se pueden duplicar y tampoco pueden ser nulos, por ejemplo: una tabla muy sencilla puede ser "continente", en esta tabla se pueden tener los siguientes campos, "*id" y* nombre.

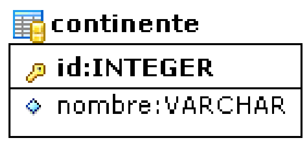


Figura 2.1.1 - Tabla continente

Los nombres y los tipos de datos de cada campo son colocados dependiendo de los requerimientos del modelo de negocios que se desea representar.

Como indicamos anteriormente en las Bases de Datos existen tablas que se mantienen ligadas mediante una relación, esta relación se lleva a cabo a través de un campo en común que comparten ambas tablas, es así que si en nuestro ejemplo tenemos otra tabla llamada "pais" con los campos "*id"* y nombre.

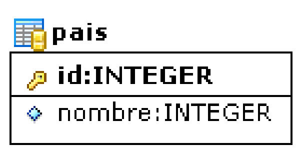


Figura 2.1.2 - Tabla pais

Esta tabla “pais” podemos relacionarla con la tabla “continente” partiendo de la hipótesis de que cada país pertenece a un continente. El campo en común que unirá a estas tablas mediante una relación será el “*id*”, el cual deberá también ser colocado en la tabla “pais” para establecer la relación, este campo se lo conoce como “*foreign key*”

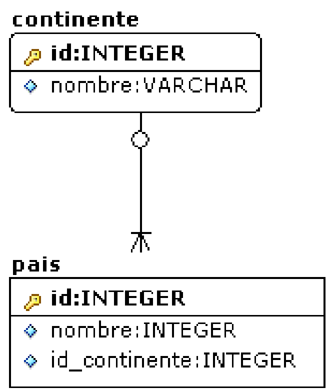


Figura 2.1.3 – Ejemplo de Modelo Entidad Relación

Pero por qué colocamos el campo “*id*” de la tabla “continente” en la tabla “pais” en lugar de poner el “*id*” de la tabla “pais” en la tabla “continente”. La explicación es la siguiente: sabemos que un continente tiene muchos países, pero un país pertenece solo a un continente.

Para facilitar la comprensión de este ejemplo se seleccionaron las tablas “continente” y “pais”, debido a que al llenar la tabla “continente” sabemos que esta tendrá a lo mucho cinco registros ya que existen solo cinco continentes en el globo terráqueo, mientras que la tabla “pais” tendrá más de cien registros. Si colocáramos el campo “id” en la tabla "continente" esto implicaría que tengamos la misma cantidad de registros que en la tabla "pais", esto no es permitido ya que provocaría redundancia de datos.

2.2 Características de una Base de Datos

Una Bases de Datos debe tener las siguientes características:

- **Independencia de los Datos.** Es decir, que los datos no dependan de los programas y por tanto cualquier aplicación puede hacer uso de los datos.
- **Reducción de la Redundancia.** Se llama *redundancia* a la existencia de duplicación de los datos; al reducir ésta al máximo se consigue un mayor aprovechamiento del espacio y además se evita que existan *inconsistencias* entre los datos. Las *inconsistencias* se dan cuando se encuentran datos contradictorios.
- **Seguridad.** Una Base de Datos debe permitir que se tenga un estricto control sobre la seguridad de los datos.

2.3 Ventajas de las Bases de Datos Centralizadas

Entre las ventajas de utilizar Bases de Datos Centralizadas se pueden mencionar las siguientes:

- Se evita la redundancia.

En sistemas que no usan *Bases de Datos Centralizadas*, cada aplicación tiene sus propios archivos privados o se encuentran en diferentes localidades. Esto a menudo origina enorme redundancia en los datos almacenados, así como desperdicio resultante del espacio de almacenamiento; por ejemplo, una aplicación de personal y otra de registros educativos pueden poseer cada una un archivo que contenga información de departamento de los empleados. Estos dos archivos pueden integrarse (para eliminar la redundancia) si el Administrador de la Base de Datos (DBA) está consciente de los requerimientos de información para ambas aplicaciones, es decir, si el DBA tiene el control global necesario.

- Pueden hacerse cumplir las normas establecidas.

  Con un control central de la base de datos, el DBA puede garantizar que se cumplan todas las formas aplicables a la representación de los datos. Las normas aplicables pueden comprender la totalidad o parte de lo siguiente: normas de la compañía, de instalación, departamentales, industriales, nacionales o internacionales. Es muy deseable unificar los formatos de los datos almacenados como ayuda para el intercambio o migración de datos entre sistemas.

- Pueden aplicarse restricciones de seguridad.

Al tener jurisdicción completa sobre los datos de operación, el DBA puede:

- Asegurar que el único medio de acceder la base de datos sea a través de los canales establecidos y, por tanto,
- Definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles. Diferentes controles pueden establecerse para cada tipo de acceso (recuperación, modificación, supresión, etc.) a cada parte de la información de la base de datos.

- Puede conservarse la integridad.

  El problema de la integridad es garantizar que los datos de la Base de Datos sean exactos. El control centralizado de la base de datos, es decir, que los datos se encuentren en una sola máquina, ayuda a evitar la inconsistencia de los datos, por el mismo hecho de encontrarse en una sola máquina. Es conveniente señalar que la integridad de los datos es un aspecto muy importante en una Base de Datos, porque los datos almacenados se comparten y porque sin procedimientos de validación adecuados es posible que un programa con errores genere datos incorrectos que afecten a otros programas que utilicen esa información.

- Pueden equilibrarse los requerimientos contradictorios.

Cuando se conocen los requerimientos globales de la empresa, en contraste con los requerimientos de cualquier usuario individual, el DBA puede estructurar el sistema de Base de Datos para brindar un servicio que sea el mejor para la empresa en términos globales. Por ejemplo, puede elegirse una representación de los datos almacenados que ofrezca rápido acceso a las aplicaciones más importantes a costa de un desempeño de menor calidad en algunas otras aplicaciones.

2.4 Arquitectura Cliente/Servidor

Diversas aplicaciones se ejecutan en un entorno Cliente/Servidor. Esto significa que los equipos clientes (equipos que forman parte de una red) contactan a un servidor, un equipo generalmente muy potente en materia de capacidad de entrada/salida, que proporciona servicios a los equipos clientes. Estos servicios pueden ser programas, archivos, Bases de Datos, etc. Dado este entorno un cliente que necesite hacer una consulta o actualización en una Base de Datos centralizada, envía una petición al servidor de la Base de Datos y este le devuelve los datos solicitados.

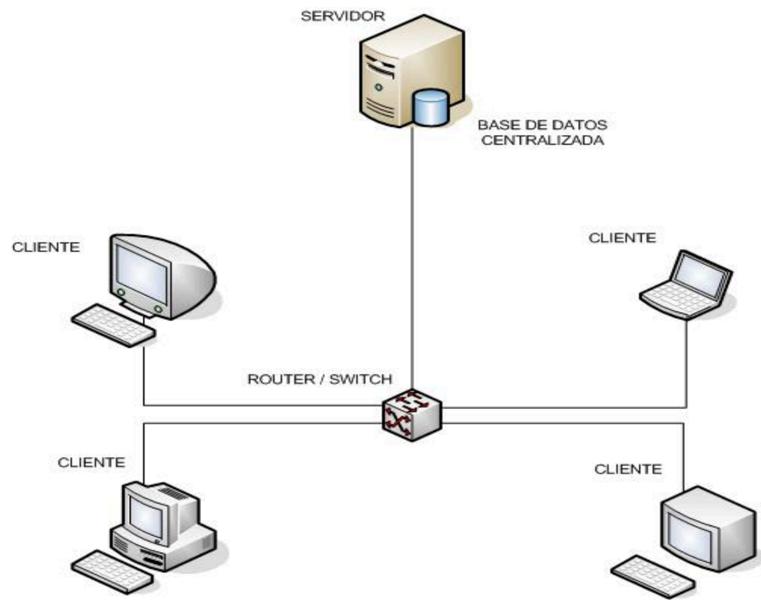


Figura 2.4.1 – Arquitectura Cliente / Servidor

En el pasado los servidores se instalaban para controlar las impresiones y el acceso a los archivos, pero hoy en día la mayoría son Servidores de Base de Datos, Servidores Web, mientras los clientes son los que manipulan terminales con una Interfaz Gráfica del Usuario (GUI).

2.4.1 Ventajas de la arquitectura Cliente/Servidor

El modelo cliente/servidor se recomienda, en particular, para redes que requieran un alto grado de fiabilidad. Las principales ventajas son:

- **Recursos centralizados**: debido a que el servidor es el centro de la red, puede administrar los recursos que son comunes a todos los usuarios, por ejemplo: una base de datos centralizada se utilizaría para evitar problemas provocados por datos contradictorios y redundantes.
- **Seguridad mejorada**: ya que la cantidad de puntos de entrada que permite el acceso a los datos no es importante.
- **Administración al nivel del servidor**: ya que los clientes no juegan un papel importante en este modelo, requieren menos administración.
- **Red escalable**: gracias a esta arquitectura, es posible quitar o agregar clientes sin afectar el funcionamiento de la red y sin la necesidad de realizar mayores modificaciones.
- **Tolerancia a fallos:** Afortunadamente, el servidor es altamente tolerante **a los fallos (principalmente gracias al sistema RAID).**

2.4.2 Desventajas de la arquitectura Cliente/Servidor

La arquitectura cliente/servidor también tiene las siguientes desventajas:

- **Costo elevado:** debido a la complejidad técnica del servidor.
- **Un eslabón débil:** el servidor es el único eslabón débil en la red de cliente/servidor, debido a que toda la red está construida en torno a él.

2.4.3 Funcionamiento de la arquitectura Cliente/Servidor

El cliente envía una solicitud al servidor mediante su dirección IP y el puerto, que está reservado para un servicio en particular que se ejecuta en el servidor.

El servidor recibe la solicitud y responde con la dirección IP del equipo cliente y su puerto.

2.5 El Motor de Bases de Datos MySQL

MySQL es un motor de Base de Datos relacional, multihilo y multiusuario que en su mayor parte está desarrollado en ANSI C. Oracle Corporation desde abril de 2009 desarrolla MySQL como software libre en un esquema de licenciamiento dual.

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.

A diferencia de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propietario y está 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. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

2.6 Introducción a LabVIEW

LabVIEW (*Laboratory Virtual Instrumentation Engineering Workbench)*, es un entorno de desarrollo de aplicaciones para adquisición de datos, instrumentación y control automático, y es ampliamente utilizado por ingenieros y científicos de todo el mundo.

Hay que recalcar que LabVIEW implementa un lenguaje de programación grafico denominado lenguaje “G”, es decir, que a diferencia de lenguajes tradicionales que utilizan líneas de código para el desarrollo de una aplicación, se usan iconos con entradas y salidas para desarrollar cualquier tipo de aplicación.

Los programas de LabVIEW se denominan instrumentos virtuales o VI, debido a que su apariencia y operación imita a los instrumentos físicos, tales como osciloscopios y multímetros. LabVIEW contiene un número comprensible de herramientas para adquisición, análisis, despliegue, y almacenamiento de datos, así como herramientas que le ayudan a resolver su código de ejecución. En todo VI existen dos áreas de trabajo muy importantes, el Front Panel (Panel Frontal) y el Block Diagram (Diagrama de Bloques).

El Panel Frontal es la interfaz que el usuario está viendo y puede ser totalmente parecido al instrumento del cual se están recogiendo los datos, de esta manera el usuario sabe de manera precisa cual es el estado actual de dicho instrumento y los valores de las señales que se están midiendo.

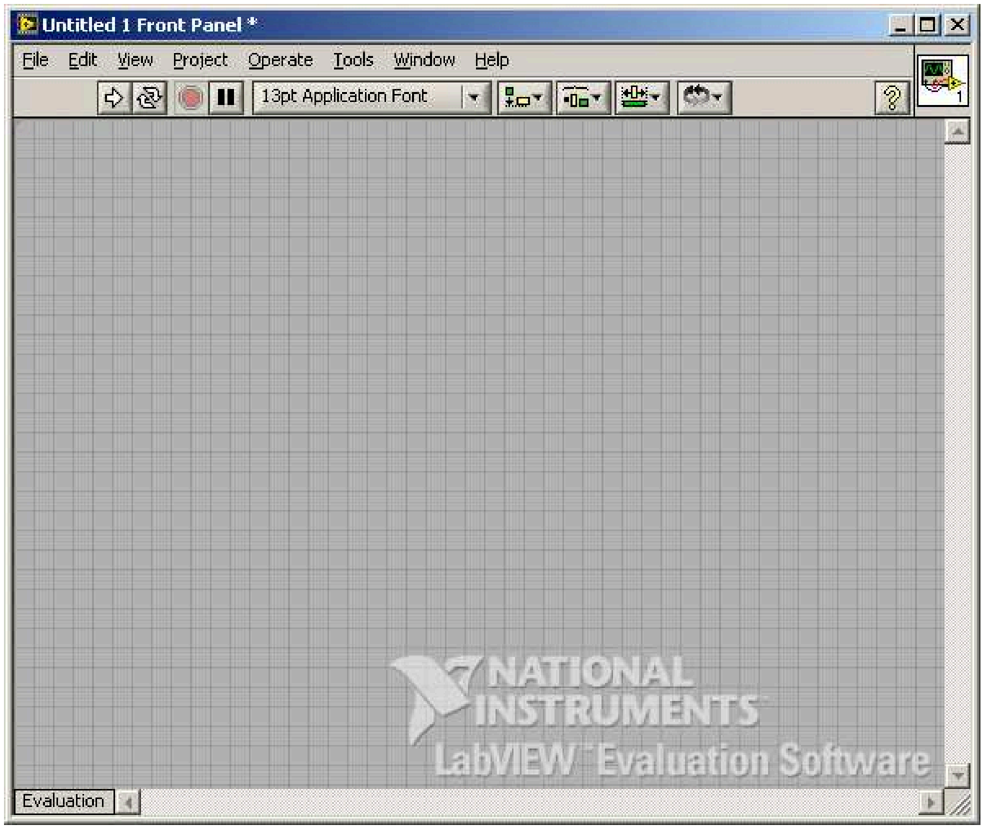


Figura 2.6.1 - Panel Frontal de LabVIEW

Los controles simulan instrumentos de entrada y entregan los respectivos datos al diagrama de bloques del VI. Entre los controles tenemos perillas, botones y otros dispositivos de entrada

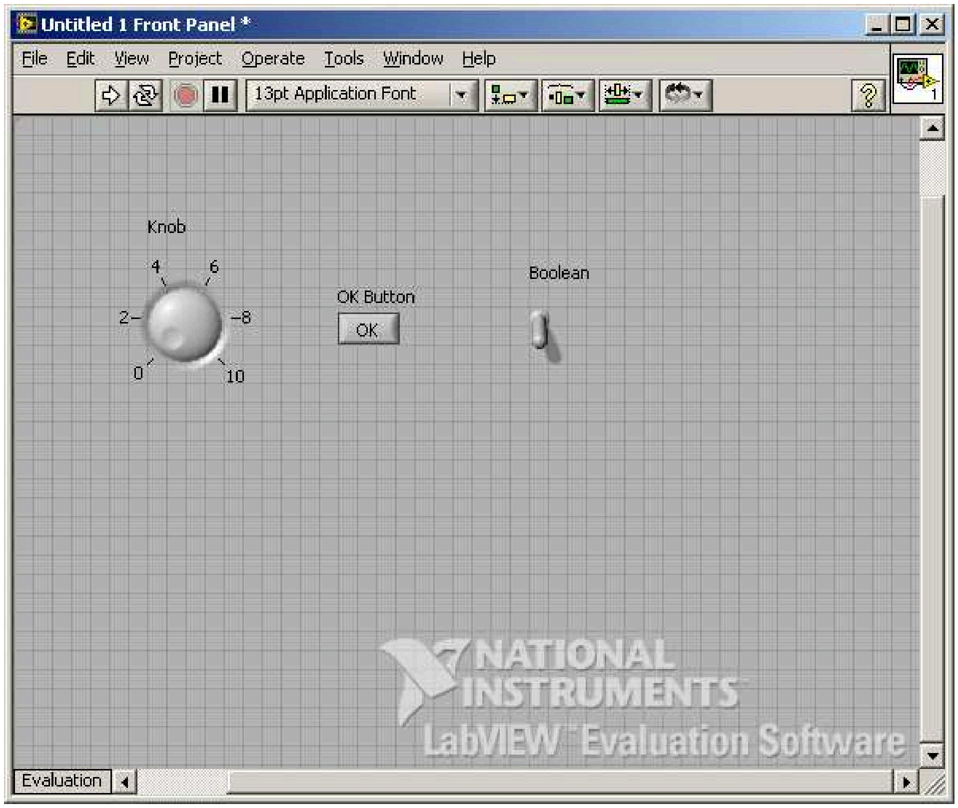


Figura 2.6.2 – Ejemplo de Controles en el Panel Frontal de LabVIEW

Los indicadores simulan instrumentos de salida y muestran los datos que el diagrama de bloques genera o adquiere. Dentro de los indicadores podemos encontrar led's, gráficos y otros tipos de display.

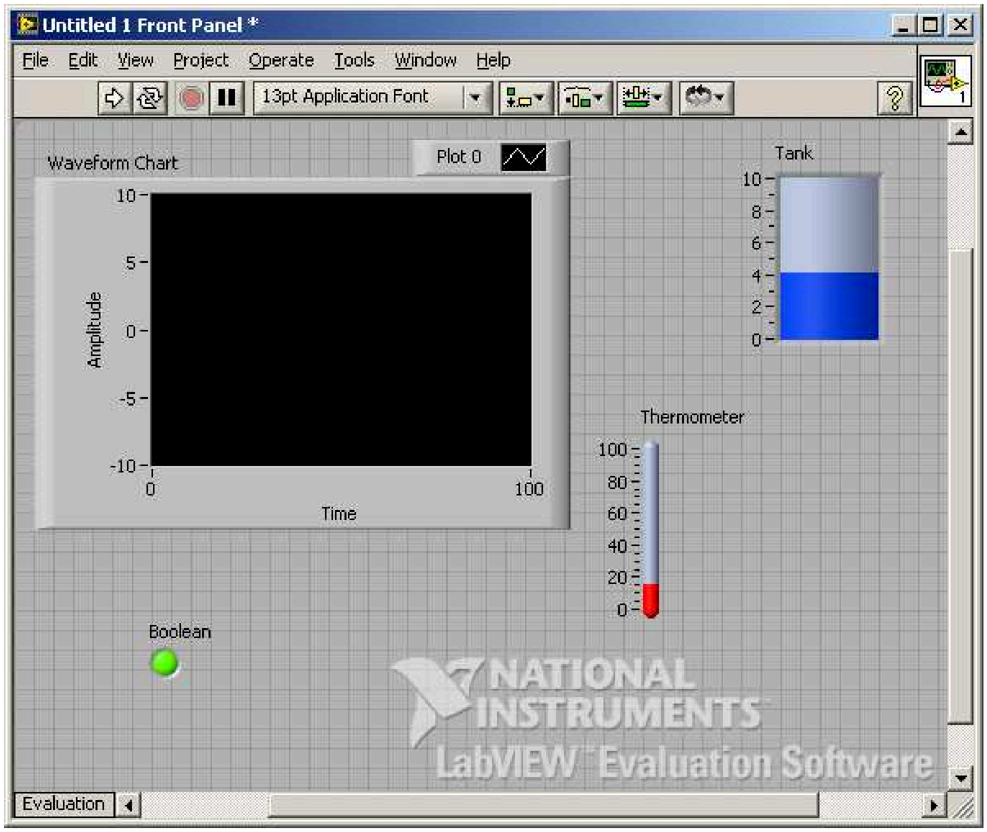


Figura 2.6.3 – Ejemplo de Indicadores en el Panel Frontal de LabVIEW

En el Diagrama de bloques están todas las conexiones de todos los controles y variables.

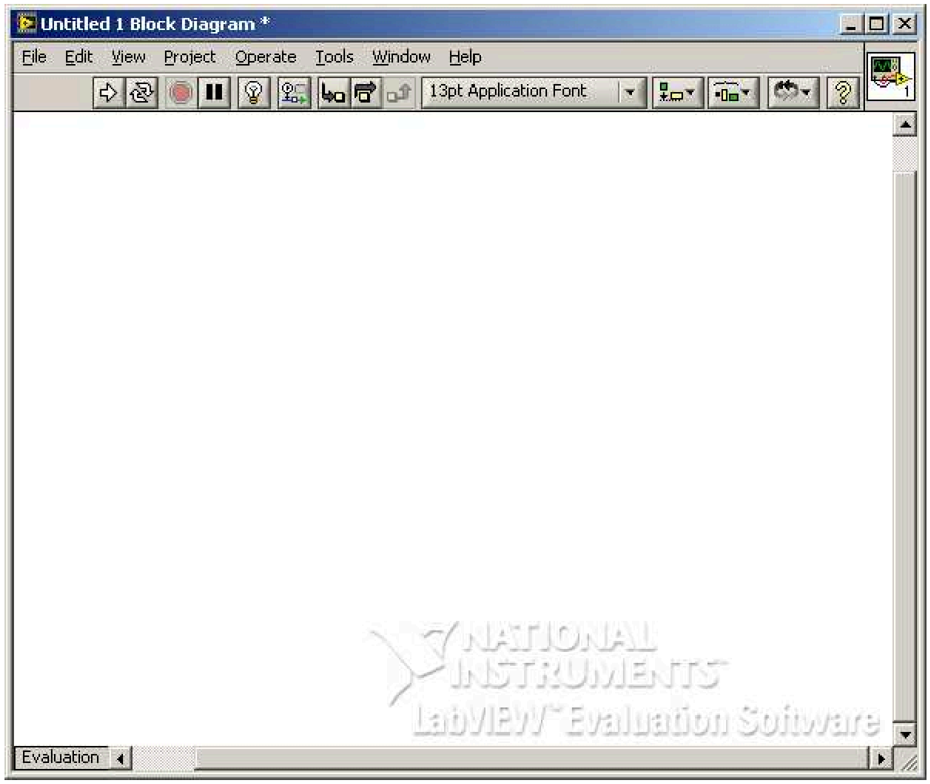


Figura 2.6.4 – Diagrama de Bloques de LabVIEW

En LabVIEW como en todo lenguaje de programación existen tipos de datos, estos tipos pueden ser agrupados en tres grandes grupos: numéricos, booleanos y alfanuméricos. De igual manera existen estructuras de control como lo son los case, for, while, etc.

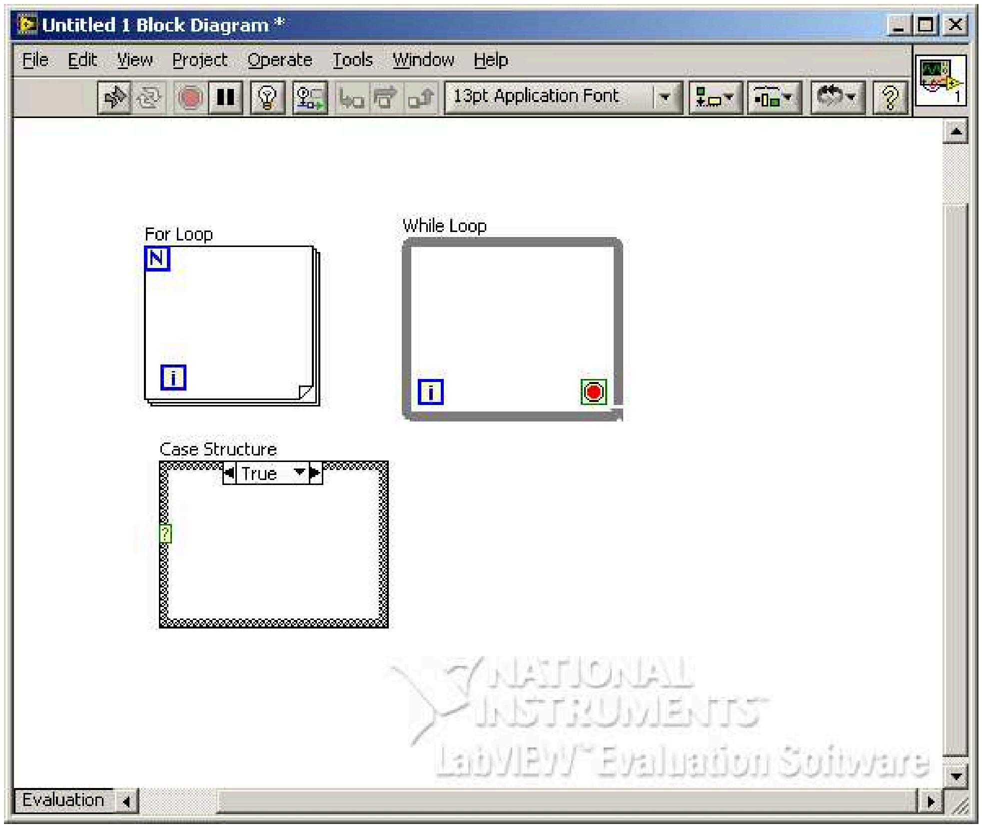


Figura 2.6.5 – Ejemplo de Estructuras de Control en LabVIEW

Ejemplo 1: Realicemos un ejemplo sencillo donde tengamos un control de perilla que al sobrepasar el valor límite de 200 encienda un led indicador.

Para el efecto en el Panel Frontal colocaremos tres controles y un indicador. **Controles:** Perilla, Control numérico y botón de Stop **Indicador:** Led

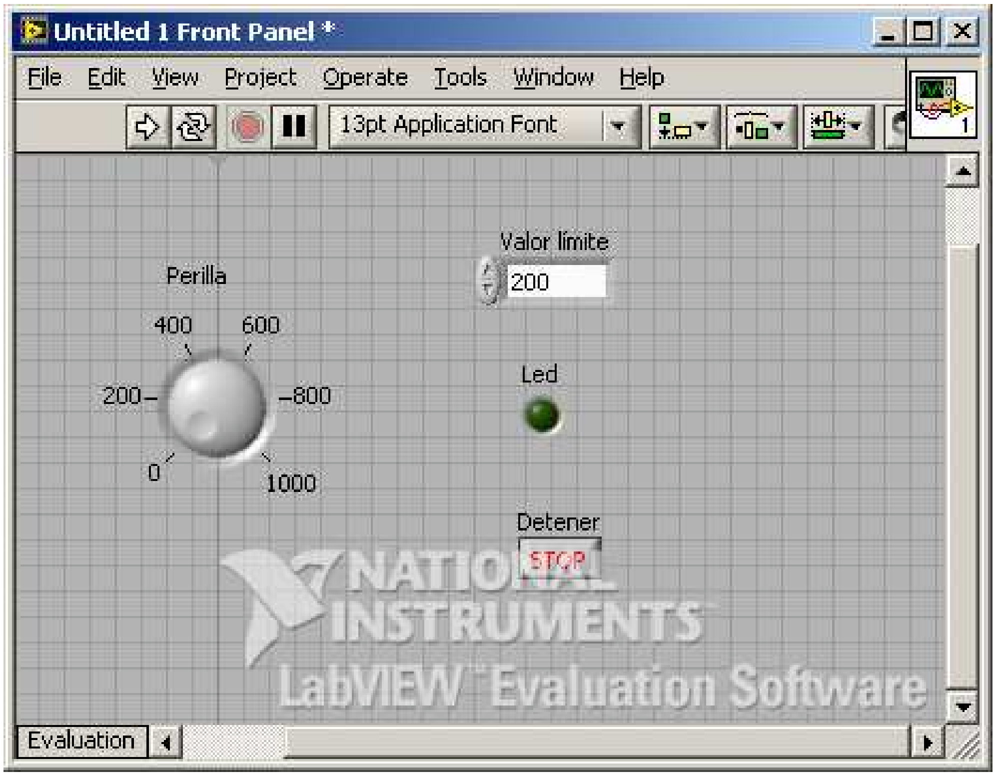


Figura 2.6.6 – Panel Frontal del Ejemplo 1

En el Diagrama de Bloques tenemos que comenzar colocando una estructura de control While Loop que mantenga nuestra aplicación ejecutándose hasta que se presione el botón Detener.

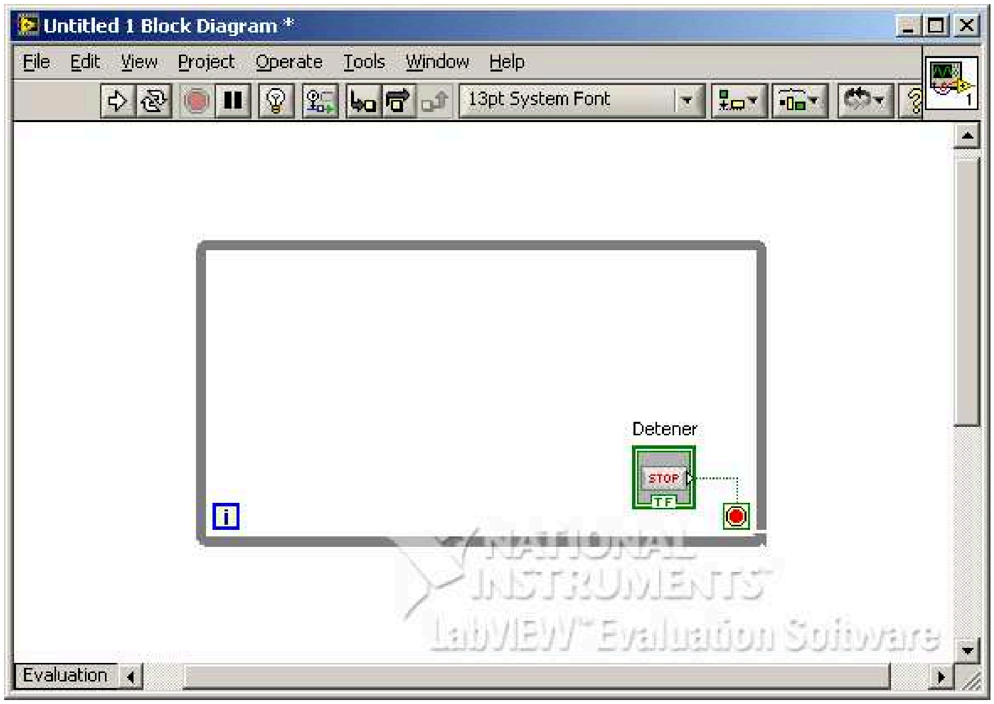


Figura 2.6.7 – Diagrama de Bloques del Ejemplo 1.

Ahora debemos colocar una estructura Case Structure para que cuando movamos la perilla y el valor de esta pase de 200 se encienda el led indicador. Para saber si el valor de la perilla ha superado el valor límite necesitamos un comparador numérico el cual deberá tener como entradas el valor de la Perilla y el valor del Control numérico, este comparador devuelve como salida un valor Booleano de True o False que será la entrada de la estructura Case Structure.

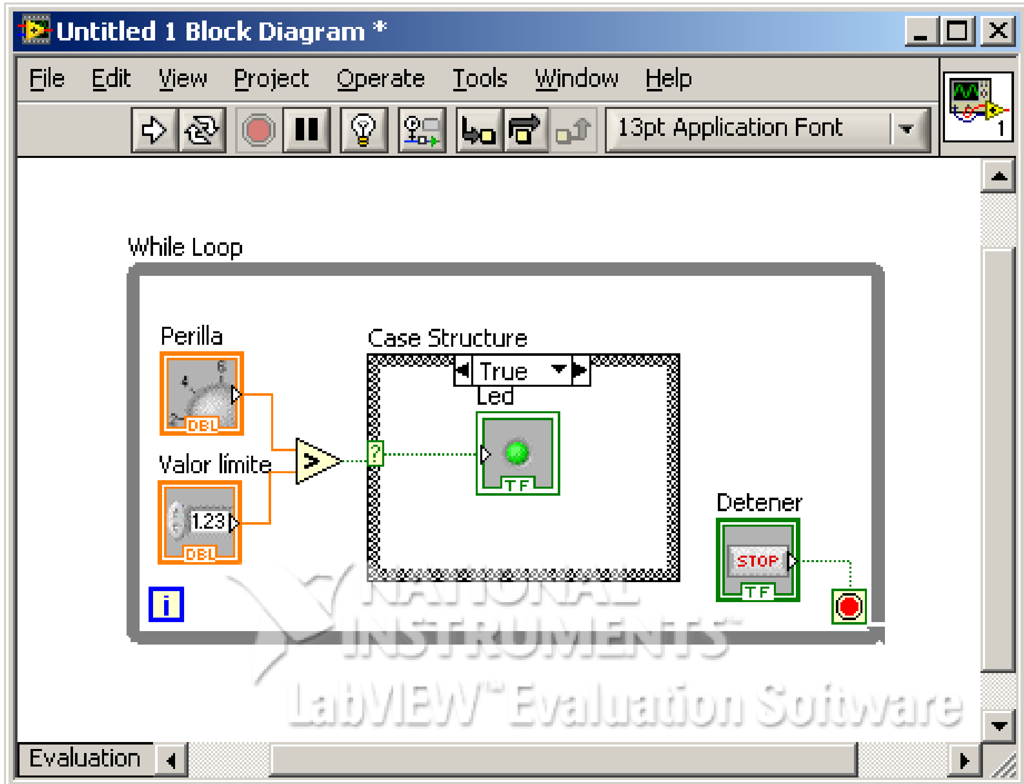


Figura 2.6.8 – Diagrama de Bloques del Ejemplo 1 (case true).

Dentro de la estructura Case Structure debemos colocar lo que queremos hacer cuando la Perilla pase el valor de 200. Esto lo conseguimos uniendo la entrada del Case Structure al led indicador cuando el Case Structure este en True.

Finalmente cambiamos el valor del Case Structure a False y colocamos lo que queremos que suceda cuando la perilla no pase el valor límite de 200. Esto lo conseguimos creando una variable local del led indicador y conectándole como entrada una constante de False.

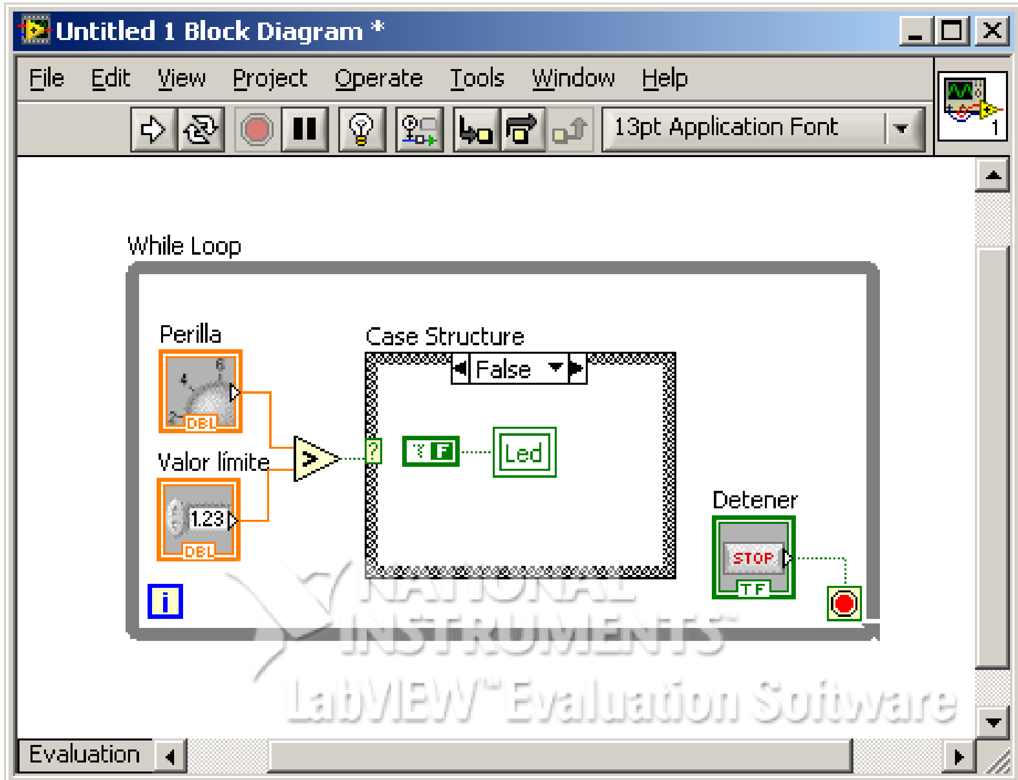


Figura 2.6.9 – Diagrama de Bloques del Ejemplo 1 (Case False)

Los programas en LabVIEW pueden ser tan sencillos como este ejemplo o tan complejos cuando existan muchos controles e indicadores en el Panel Frontal e interconexiones entre estos controles e indicadores en el Diagrama de Bloques.

2.7 Uso del Database Connectivity Toolset de LabVIEW

AL igual que como se hizo con las Base de Datos Centralizadas y luego de revisar brevemente el ambiente de trabajo en LabVIEW, nos toca revisar un conjunto de herramientas para manipular bases de datos, el

Database Connectivity Toolset, estas herramientas son muy fáciles de manejar solo hay que saber interpretar sus entrada y salidas, además de tener conocimientos básicos de Bases de Datos.

Antes de usar estas herramientas tenemos que hacer un pasos previos, como vamos a utilizar Motor de Base de Datos MySQL tenemos que crear un ODBC para conectarnos desde LabVIEW a nuestra Base de Datos. Esto es necesario ya que el objeto Open Connection del Database Connectivity (ver Figura 2.7.2.1) recibe un parámetro llamado ***connection information*** el cual justamente es el nombre que le pondremos a nuestro ODBC.

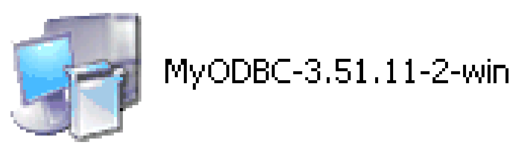


Figura 2.7.1¡Error! No hay texto con el estilo especificado en el documento. **– Icono de MyODBC Driver**

Para este proyecto se utilizo el MyODBC – 3.51.11-2-win, el cual luego de instalar tenemos que configurar de la siguiente manera.

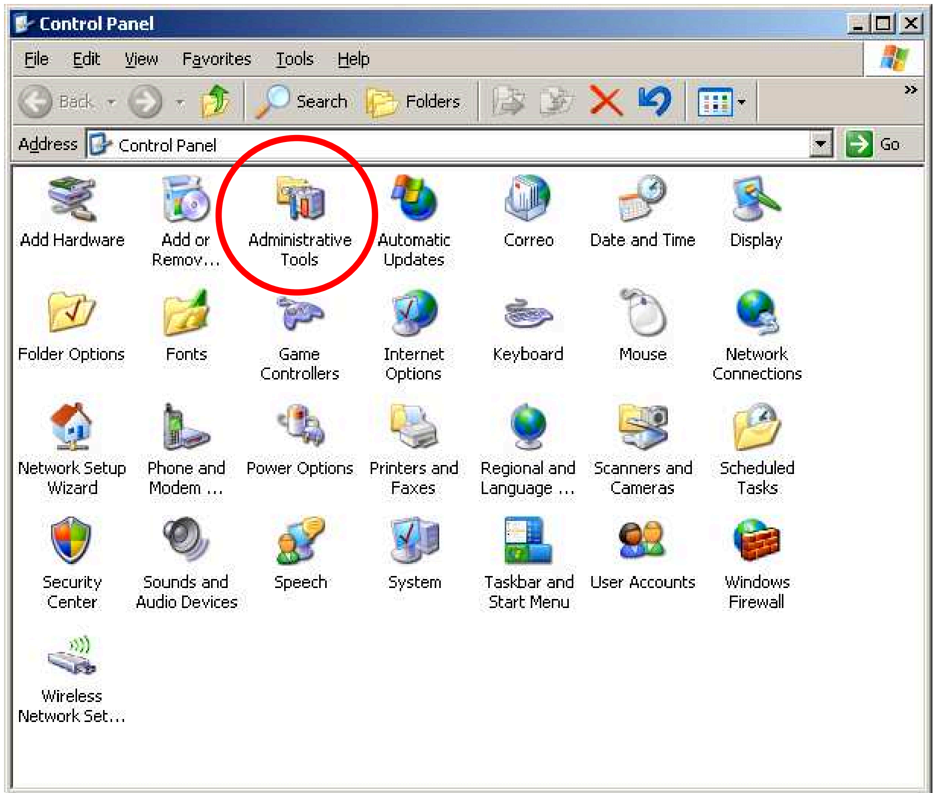


Figura 2.7.2 – Panel de Control → Administrative Tools

Nos dirigimos al Control Panel de Windows XP → Administrative Tools

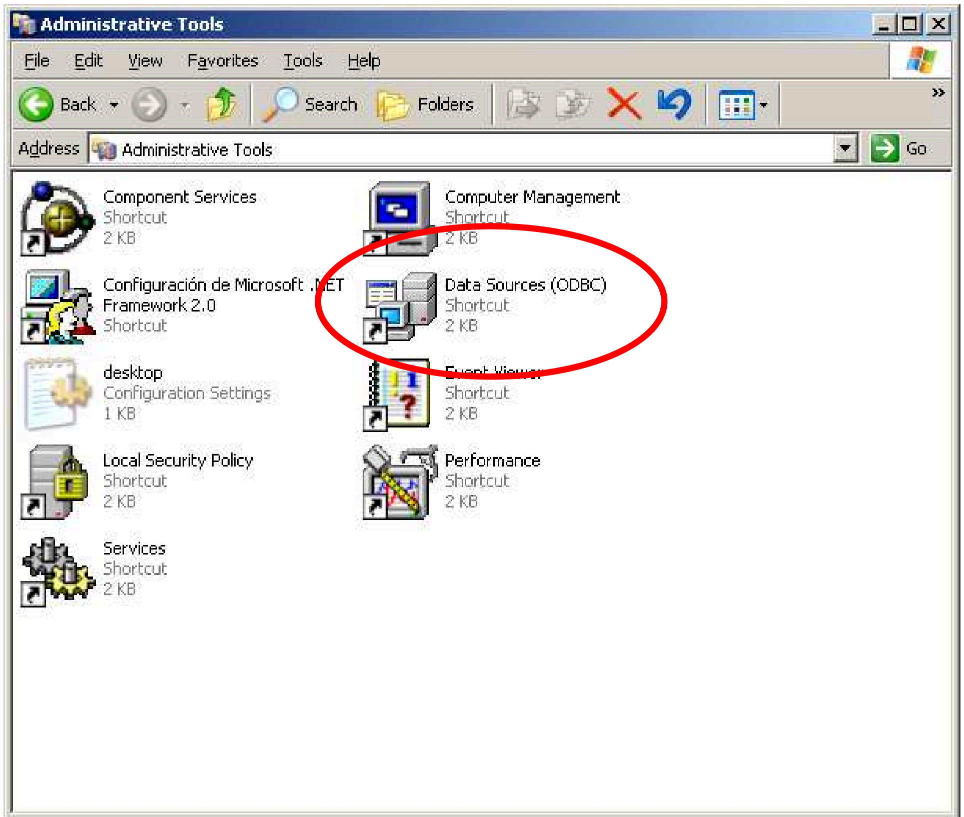


Figura 2.7.3 – Administrative Tools → Data Sorce (ODBC)

Luego seleccionamos Data Souces (ODBC)

2.7.1 Pasos para configurar un ODBC para conectarse con MySQL

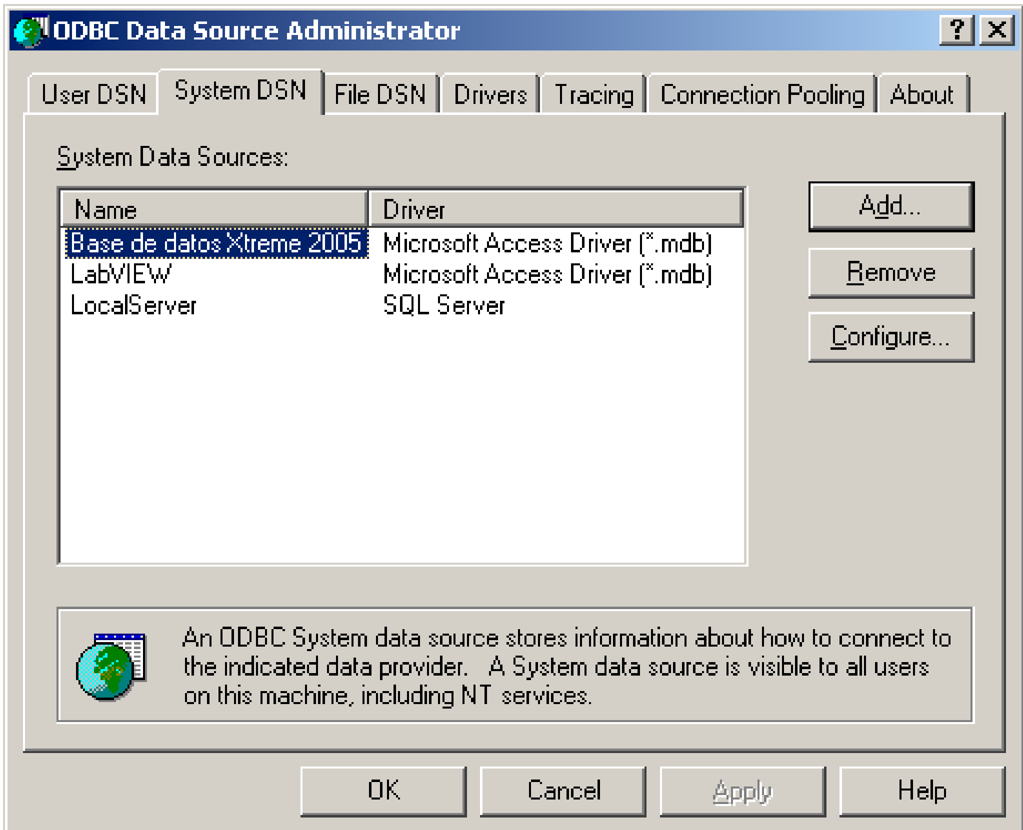


Figura 2.7.1.1 ODBC Data Source Administrator → Click en "Add"

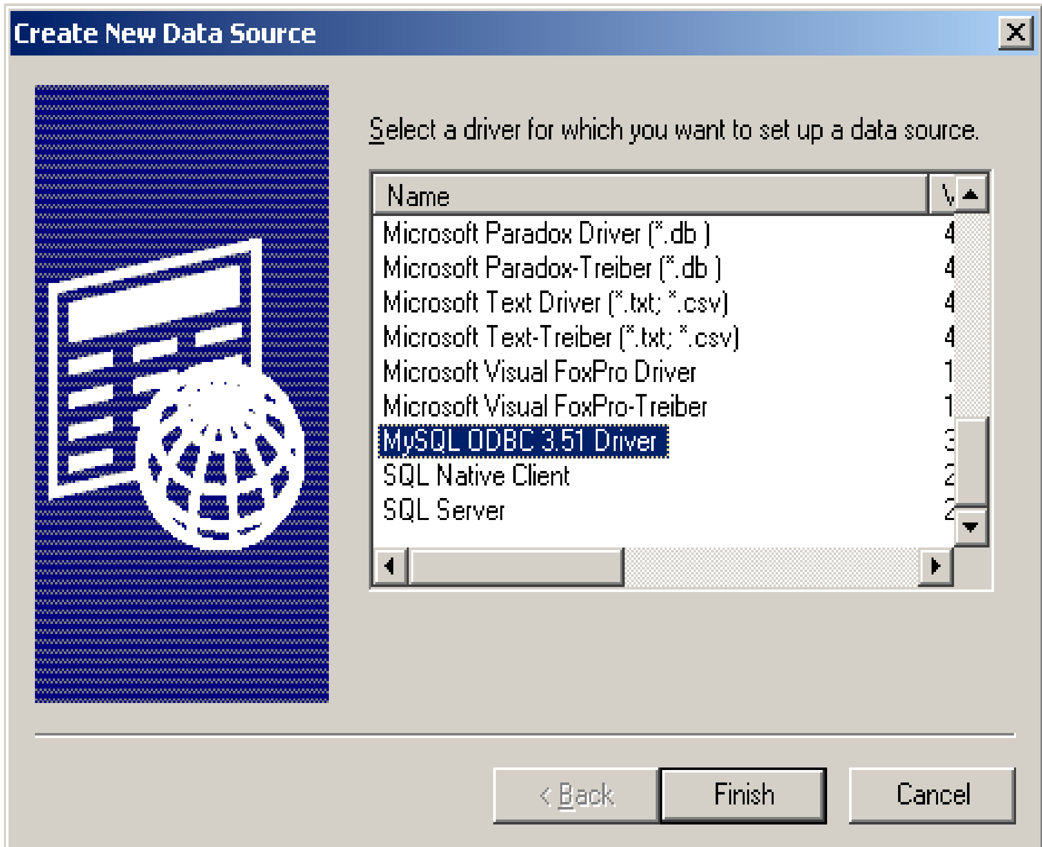


Figura 2.7.1.2 Create New Data Source → MySQL ODBC 3.51 → Click en "Finish"

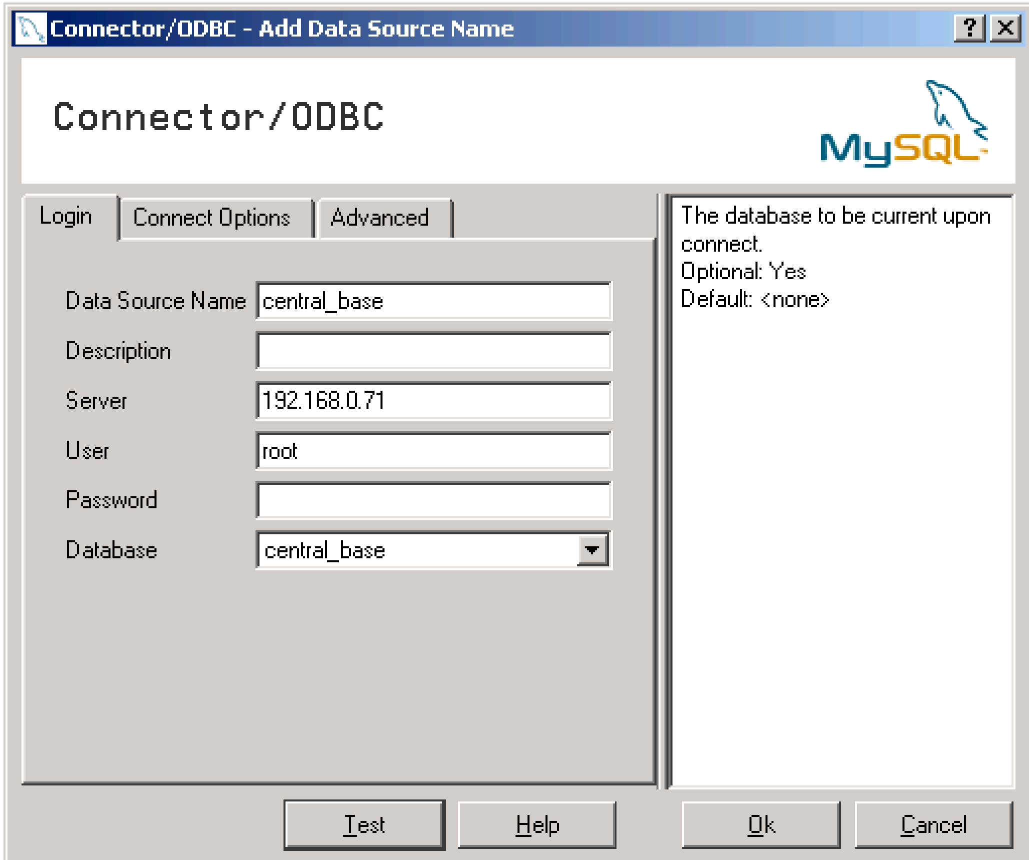


Figura 2.7.1.3 Connector / ODBC - Add Data Source Name → Click en "Test".

Figura 2.7.1.4 Connection / ODBC → Test Connection → Success → Click en "Ok".

Luego de hacer estos pasos tenemos que nuestro ODBC se llamara tal como el Data Source Name que colocamos, para nuestro caso este se llama “*central_base*”, cabe recalcar que los datos que llenamos en la Figura 2.7.1.3 A son importantes ya que se especifica la dirección IP de donde está la Base de Datos Central a la cual nos queremos conectar, así como el username y el password.

Para abrir y cerrar una conexión a una base de datos necesitamos usar dos iconos del Database Connectivity Toolset los cuales son DB Tools Open Connection y DB Tools Close Connection.

2.7.2 DB Tools Open Connection

Abre una conexión de base de datos utilizando la información de conexión, que por lo general es el Data Source Name.

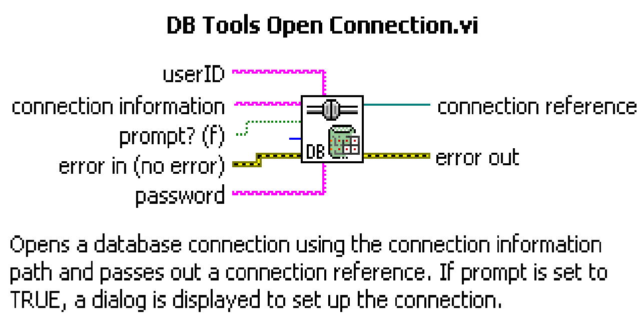


Figura 2.7.2.1 – Icono DB Tools Open Connection

2.7.3 DB Tools Close Connection

Cierra una conexión de base de datos mediante la destrucción de sus referencias de conexión asociadas.

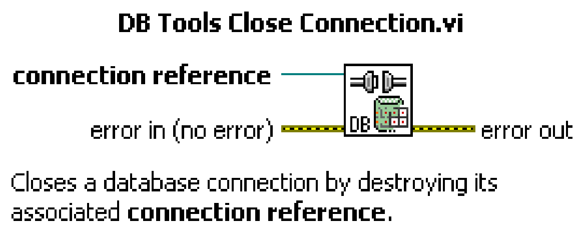


Figura 2.7.3.1 – Icono DB Tools Close Connection

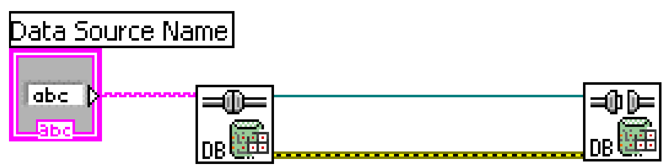


Figura 2.7.3.2 – Diagrama de Bloques de como abrir y cerrar una conexión a una Base de Datos

Para hacer una consulta a la base de datos se utilizan otros iconos del Database Connectivity Toolset, estos son: DB Tools Execute Query, DB Tools Fetch Recordset Data y DB Tools Free Object.

2.7.4 DB Tools Execute Query

Ejecuta una consulta SQL y devuelve conjunto de registros de referencia que luego deben ser liberados.

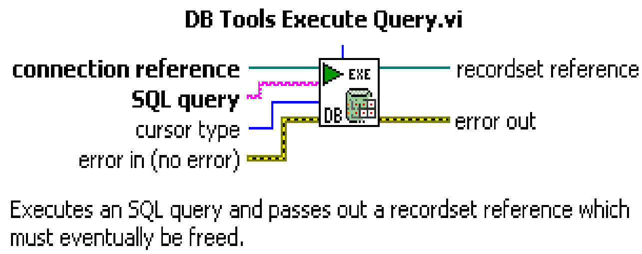


Figura 2.7.4.1 - Icono DB Tools Execute Query

2.7.5 DB Tools Fetch Recordset Data

Exporta los resultados obtenidos del ejecutar una consulta SQL a un objeto array 2-D de variants.

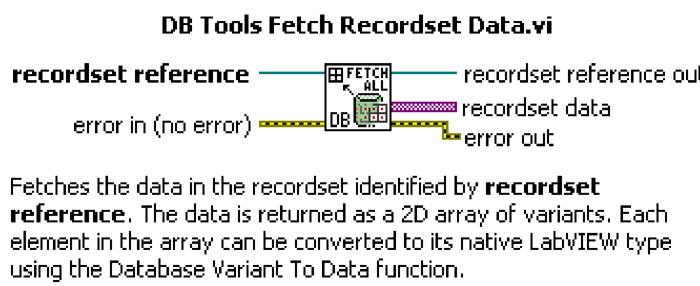


Figura 2.7.5.1 - Icono DB Tools Fetch Recordset Data

2.7.6 DB Tools Free Object

Libera un objeto mediante la destrucción de sus referencias asociadas y devuelve la referencia a la conexión de la base de datos.

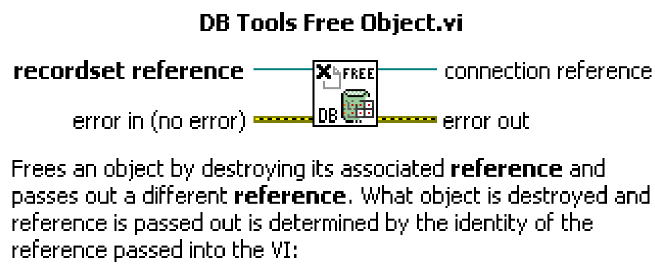


Figura 2.7.6.1 - Icono DB Tools Free Object

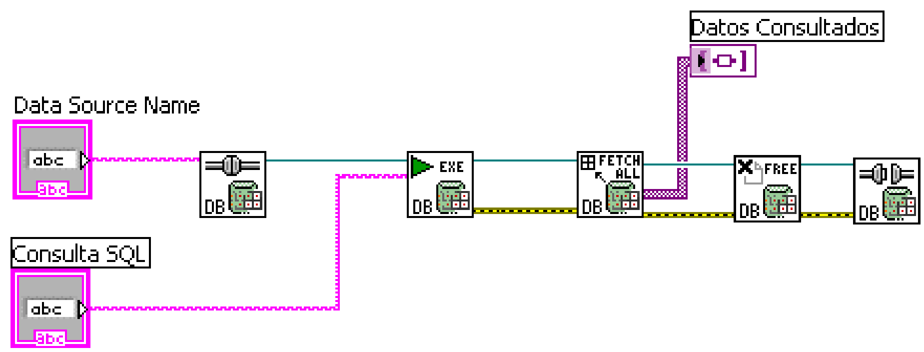


Figura 2.7.6.2 – Diagrama de Bloques → Consulta SQL a una Base de Datos

De una forma muy similar como se hizo la consulta a la base de datos podemos realizar otras operaciones como son insertar, actualizar y eliminar registros.

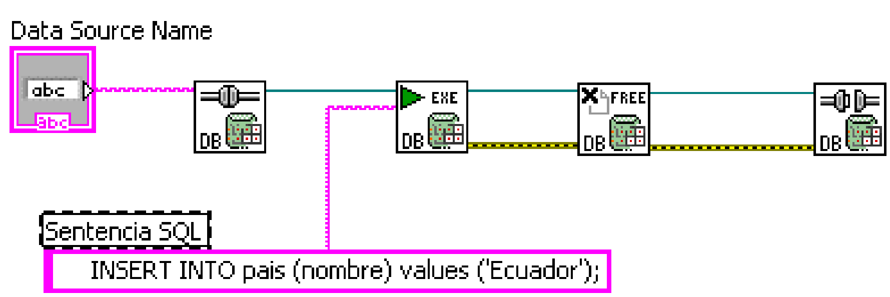


Figura 2.7.6.3 - Diagrama de Bloques → Insertar un registro en una tabla de una Base de Datos

Aquí utilizamos iconos del Database Connectivity Toolset ya conocidos, entre los cuales tenemos el DB Tools Open Connection, DB Tools Query Execute, DB Tools Free Object y el DB Tools Close Connection.

2.8 Uso del NI-IMAQ for USB Cameras

Al igual que se revisaron básicamente los iconos para el manejo de Bases de Datos ahora pasaremos a revisar brevemente el uso de los iconos para el manejo de imágenes el LabVIEW los cuales están en el paquete ***NI - IMAQ for USB Cameras***.

2.8.1 IMAQ Create

Este icono crea un espacio de memoria temporal para una imagen, se debe especificar al menos el nombre de la imagen que se quiere crear.

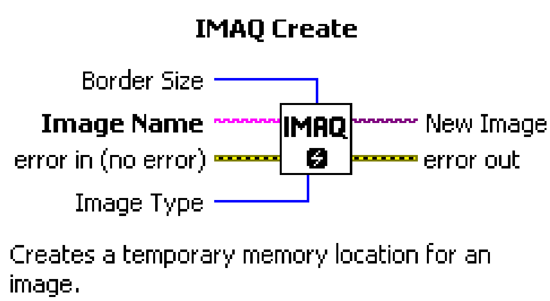


Figura 2.8.1.1 – Icono IMAQ Create

2.8.2 IMAQ Dispose

Este icono elimina una imagen y libera el espacio de memoria que esta ocupaba, como entrada recibe una imagen creada. Se puede usar un solo icono de IMAQ Dispose para todas las imágenes creadas con el icono IMAQ Create.

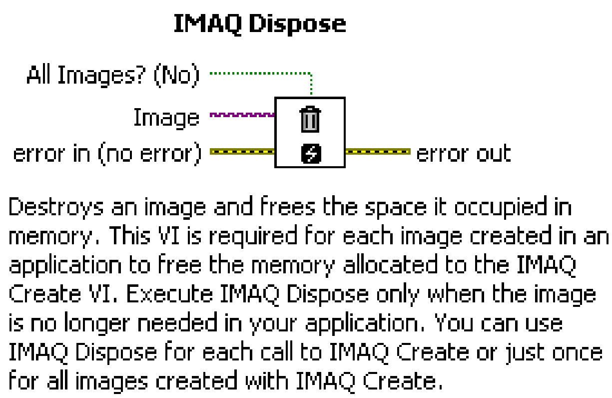


Figura 2.8.2.1 – Icono IMAQ Dispose

2.8.3 IMAQ Snap

Adquiere y devuelve una imagen que es capturada con un dispositivo de baja velocidad.

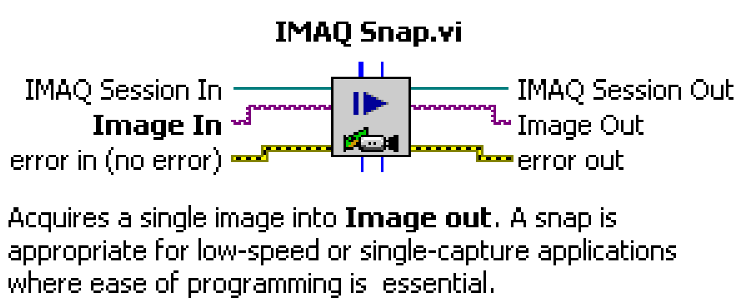


Figura 2.8.3.1 – Icono IMAQ Snap.

En la Figura 2.8.3.2 se muestra como seria adquirir una imagen con los iconos revisados hasta el momento

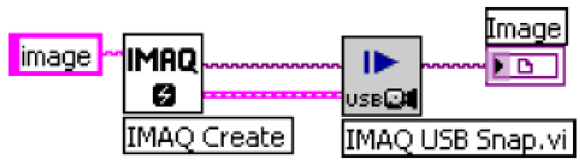


Figura 2.8.3.2 – Como crear y ver una imagen usando iconos del NI-IMAQ for USB Cameras.

2.8.4 IMAQ Grab Setup

Inicializa la sesión para comenzar a adquirir una imagen, este icono provee acceso a la imagen más reciente adquirida.

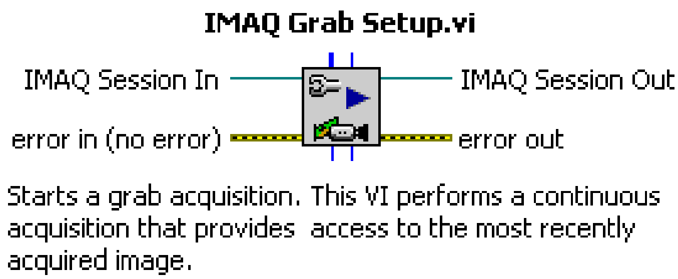


Figura 2.8.4.1 – Icono IMAQ Grab Setup.

2.8.5 IMAQ Grab Acquire

Adquiere una imagen de una sesión iniciada, este icono es usado para adquisición de imágenes a alta velocidad.

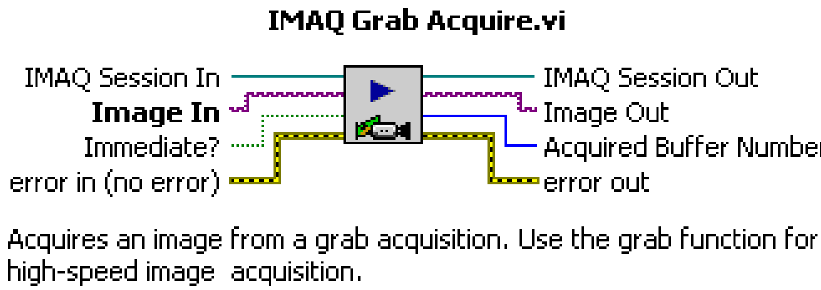


Figura 2.8.5.1 – Icono IMAQ Grab Acquire.

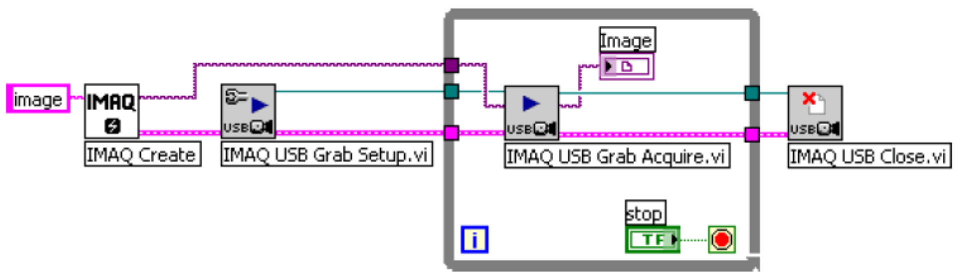


Figura 2.8.5.2 – Como obtener una imagen usando iconos de NI – IMAQ for USB Cameras

CAPÍTULO III

DISEÑO DE LA BASE DE DATOS EN MYSQL Y DE LA INTERFAZ EN LABVIEW

3.1 Diseño de la Base de Datos

Para diseñar la Base de Datos empezamos con la abstracción del mundo real, las acciones, procesos y sucesos que suceden en el entorno a abstraer.

Partimos con la premisa de que la seguridad está enfocada o establecida en una localidad. Ya sea para dar seguridad a personas, lugares, vehículos y objetos, donde los objetos los podemos clasificar en puertas, ventanas, elementos naturales, animales, plantas, etc.

Una vez establecido que se dará seguridad a una localidad, podemos definir tipos de áreas o localidades tales como: Laboratorios, Talleres, Centros Educativos, Parques, Industrias, Garita de entrada y salida de vehículos, casas, etc.

Este tipo de localidades es una forma de representar en un formato estándar lugares que pertenecen a un mismo grupo y que pueden estar en diferentes ubicaciones.

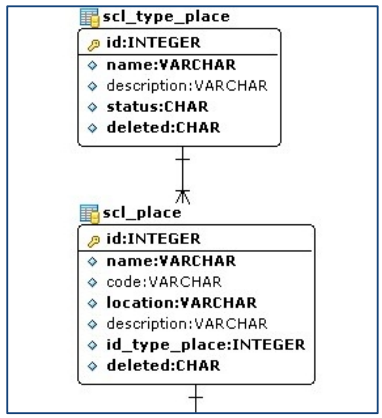


Figura 3.1.1 Tablas scl_type_place y scl_place

Hasta este punto tenemos analizado lo que es una localidad y sus tipos, como es de suponer en estas localidades existen los objetivos y a que o a quienes se les va a dar seguridad, para explicar este punto se cita los siguientes ejemplos.

Si se quiere saber quien ingreso a un departamento o un laboratorio en particular, se puede dar que exista un sensor en el laboratorio, si notamos bien este sensor esta en el laboratorio y en algún objeto específico dentro del laboratorio como una puerta, una cámara en una pared, etc. Esto es para el caso de sensores.

Si se cita un ejemplo donde hay uso de un actuador como un cerrojo electrónico para abrir una puerta que está en el laboratorio, podemos notar que el cerrojo electrónico estaría en el objeto puerta.

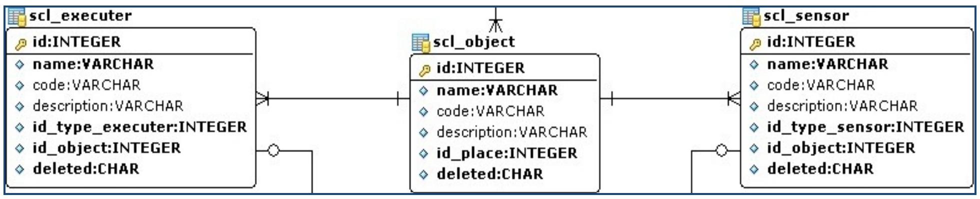


Figura 3.1.2 Tablas scl_executer, scl_object y scl_sensor

Con este par de ejemplos dados podemos sacar como una conclusión particular que los sensores y actuadores deben estar en “algo” y ese “algo” está en alguna localidad, a ese “algo” lo vamos a denominar objeto.

Bajo la deducción anterior obtuvimos que existen objetos que están dentro de una localidad y que estos objetos pueden tener un sensor y/o actuador para proveer seguridad.

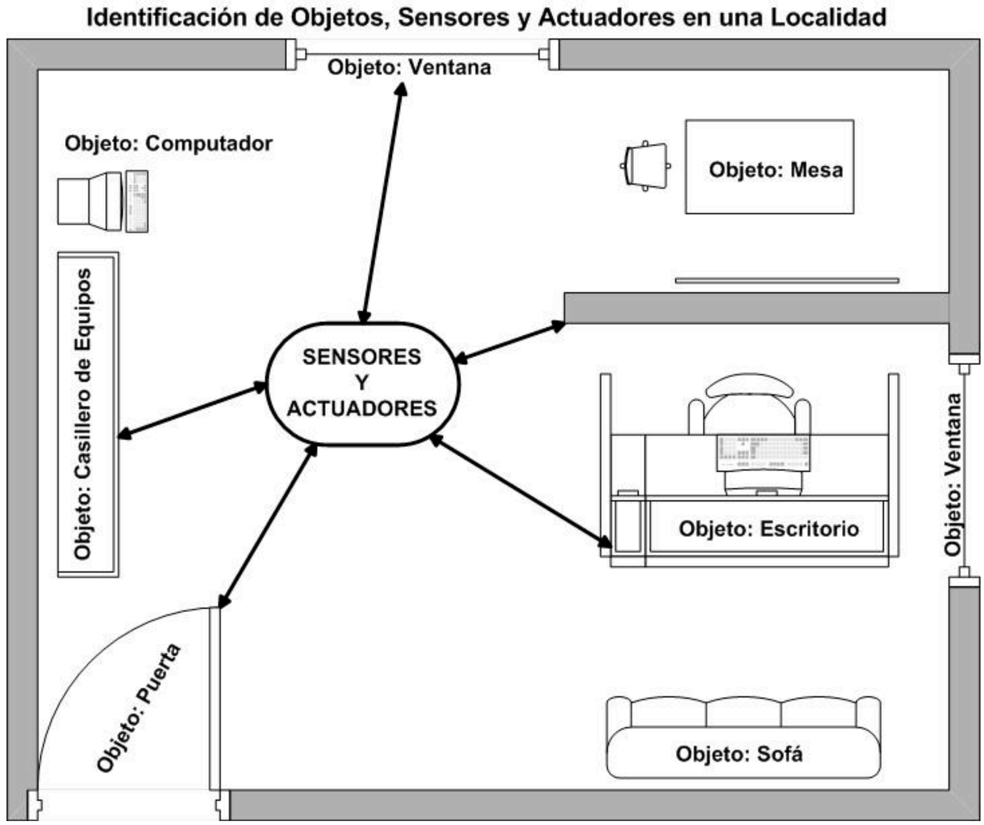


Figura 3.1.3 Objetos, sensores y actuadores de una localidad

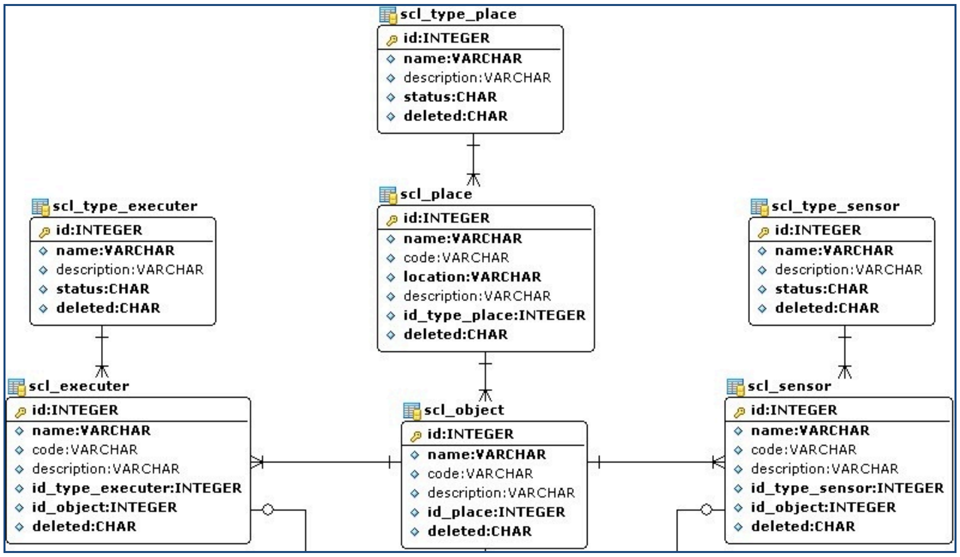


Figura 3.1.4 Tablas scl_type_place, scl_type_executer, scl_place, scl_type_sensor, scl_executer, scl_object, scl_sensor.

Un ejemplo particular, dado un objeto que posea un sensor y/o actuador, estos últimos puede sensar y/o ejecutar ya sea a un "agente" o al mismo objeto.

Cabe mencionar que el objeto no es necesario que tenga un sensor y/o actuador, puede darse el caso de que el objeto sea el "agente".

Otro de los análisis de abstracción para este proyecto y como notamos en el párrafo anterior son los "agentes", aquellos que promueven que un sensor y/o actuador opere o entre en estado activado y/o ejecutado respectivamente.

Los agentes son aquellos que estimulan que un sensor se active y/o un actuador entre en marcha. Entre los posibles agentes obtenidos bajo el análisis de los proyectos soportados en la materia de Graduación tenemos:

Persona.- Todo ser humano; niño, adolecente, joven, señor, tercera edad.

Vehículo.- Representa a cualquier medio de transporte, ya sea un automóvil, bicicleta, moto, camioneta, etc.

Objeto.- Representa a todo lo que no está vivo, como puertas, ventanas, sillas, paredes, mesas, calderos, motores, etc. Con la excepción de que vamos a considerar como objetos a las plantas y animales según amerite el caso y a los elementos naturales como aire, calor, energía, etc.

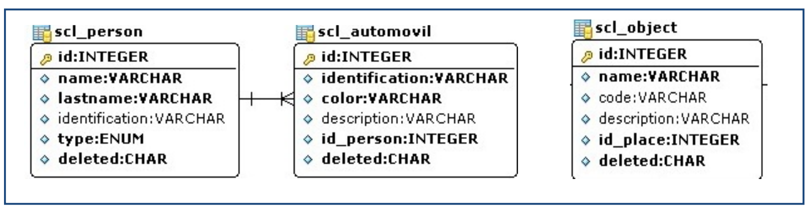


Figura 3.1.5 Tablas scl_type_person, scl_automovil, scl_object.

Podemos quizás notar que hay una redundancia del agente objeto, que es el caso de vehículo, pero necesitamos particularizar los vehículos ya que su participación como agente son importantes en comparación con otros objetos y se necesita obtener más datos de su comportamiento.

En resumen general de la primera parte de la abstracción. Tenemos que los objetos pueden o no pueden poseer sensores y/o actuadores, que estos a su vez están en alguna localidad y estas localidades pueden ser clasificadas por tipos de localidades. Que los agentes que activan y ejecutan los sensores y/o actuadores pueden ser personas, vehículos y objetos.

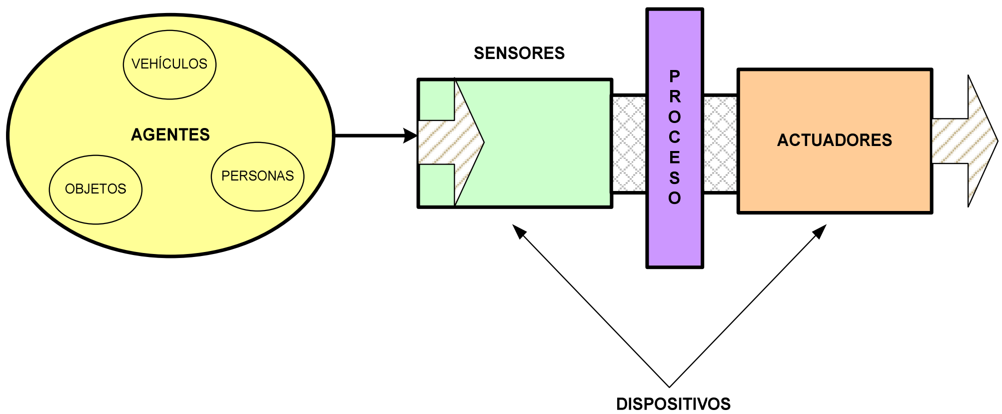


Figura 3.1.6 Diagrama de comportamiento del sistema.

Para mejorar el modelo y poder tener una clasificación ordenada de los sensores y actuadores hemos definido tipos respectivamente.

Tipo de sensores.- Agrupa de forma estándar a los sensores por sus propiedades o características, sin notar sus fabricantes, ni modelos particulares.

Ejemplo:

- Sensor de Temperatura
- Sensor de Presencia

- Sensor de Proximidad
- Sensor de Tope
- Sensor de Luminosidad
- Sensor de Falla Eléctrica
- Sensor de Fuego
- Sensor Lector RFID
- Sensor IR

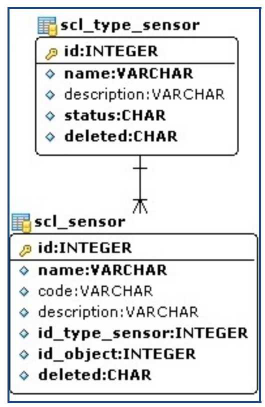


Figura 3.1.7 Tablas scl_type_sensor y scl_sensor.

Tipo de actuadores.- Agrupa de forma estándar a los actuadores por sus propiedades o características, sin notar sus fabricantes, ni modelos particulares.

Ejemplo:

- Chapa Electromecánica
- Motor Eléctrico
- Sirena o alarma
- Encendido de luz

- Apagado de luz
- Encendido general de algún equipo
- Apagado general de algún equipo
- Brazo mecánico
- Rotación de Cámara
- Electroimán

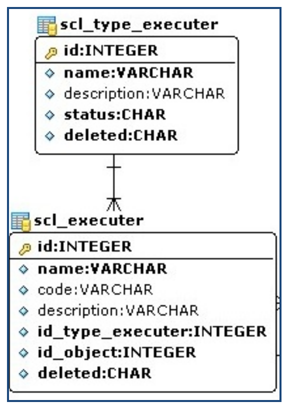


Figura 3.1.8 Tabla scl_type_executer y scl_executer.

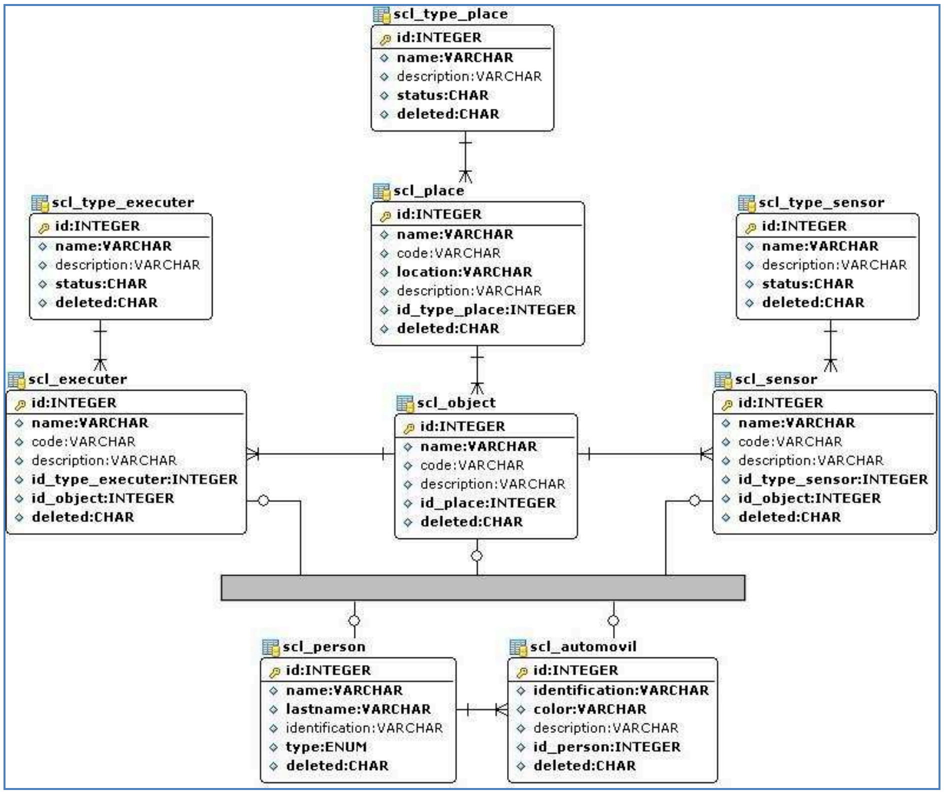


Figura 3.1.9 Parte del Diagrama Entidad Relación de la Base de Datos.

Una vez que hemos podido representar de forma abstracta los agentes, actuadores y sensores, necesitamos representar y guardar de forma homogénea o estándar los datos que pueden proveer este sistema. Hay que notar que la naturaleza de los dispositivos (sensores y actuadores) no tienen propiedades en común entre ellos, por ello es necesario establecer un estándar para vincular agentes, dispositivos e información.

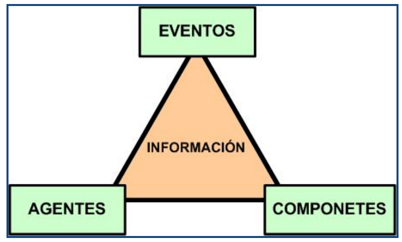


Figura 3.1.10 Relación ternaria de la información en el sistema.

Gracias a la abstracción inicial tenemos que para cada sensor y/o actuador posee un identificador único para la Base de Datos, esto nos ayuda a que la primera información que se debe considerar sea este identificador único llamado “id”. Como hemos de notar hay dos tipo de dispositivos por ende esta es otra información para considerar de la cual será SENSOR y ACTUADOR.

Con lo anterior expuesto tenemos descrito la información para identificar al tipo del dispositivo (ya sea SENSOR o ACTUADOR) con su respectivo “id” que es único para el sistema de la Base de Datos.

Falta estandarizar la información que está dada por los agentes, como hemos analizado anteriormente, los agentes son personas, vehículo y objetos, esto representa la primera información importante para los agentes. Llamándolo tipo de agente, la cual para cada agente ya sea persona, vehículo y objeto poseen un “id” único para representarlo dentro de la Base de Datos siendo este id otra información importante para describir al agente.

Con lo anterior expuesto tenemos descrito la información para identificar a el tipo de agente (ya sea PERSONA, AUTOMOVIL y OBJETO) con su respectivo “id” que es único para el sistema de Base de Datos.

Abstracción de Evento

Previo a lo analizado sobre como describir a los dispositivos y agentes, nos damos cuenta que aquello que el agente ejerce sobre un sensor y/o actuador lo vamos a llamar “evento”.

El evento es el que contiene la información conjunta de la descripción de lo que ha pasado en algún sistema de seguridad de forma particular.

Para esa descripción conjunta tenemos:

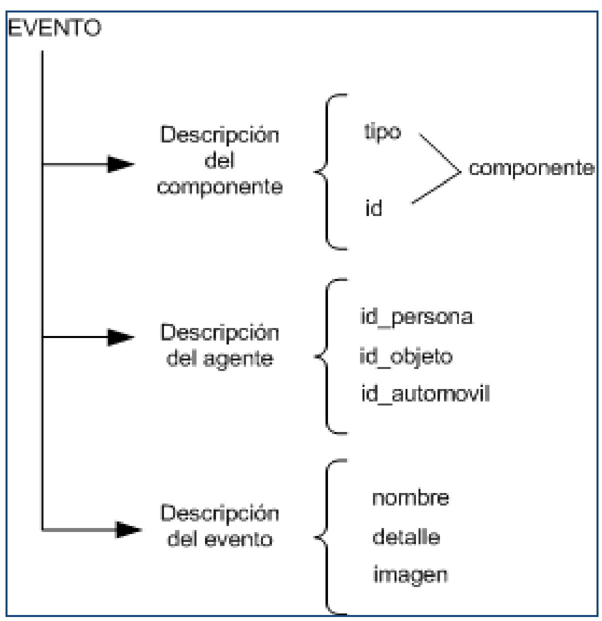


Figura 3.1.11 Descripción general de un Evento.

Descripción del Dispositivo

- Tipo del dispositivo
    - Sensor
    - Actuador
- Id único del dispositivo

Descripción del Agente

- Tipo de agente
    - Persona
    - Vehículo
    - Objeto
- Id único del agente

Descripción del Evento

- Nombre del evento
- Detalle o descripción
- Foto

Como hemos de notar se puede obtener una descripción en si del evento en la cual se obtiene:

Nombre del evento.- Es el nombre que describe al evento en sí. Ejemplo:

- Cambio de Temperatura
- Puerta se Abrió
- Puerta se Cerró

- Cambio de intensidad de luz
- Luz se prendió
- Luz se apagó
- Lectura en lector RFID
- Vehículo ha ingresado
- Vehículo ha salido

Detalle o descripción.- Se expresa de forma explícita lo que el evento ha capturado. En ciertos eventos no es tan necesario definirlo.

Ejemplo 1:

Nombre de Evento: Cambio de Temperatura

Detalle: La temperatura ha cambiado de 50°C a 70°C, el valor normal debe ser 60°C.

Ejemplo 2:

Nombre de Evento: Puerta se Cerró

Detalle: Ninguno.

Foto: En todos los proyectos de seguridades expuestos en la materia de graduación se ha definido el uso de captura de imágenes por medio de cámaras, para ciertos eventos se incluye la imagen asociada.

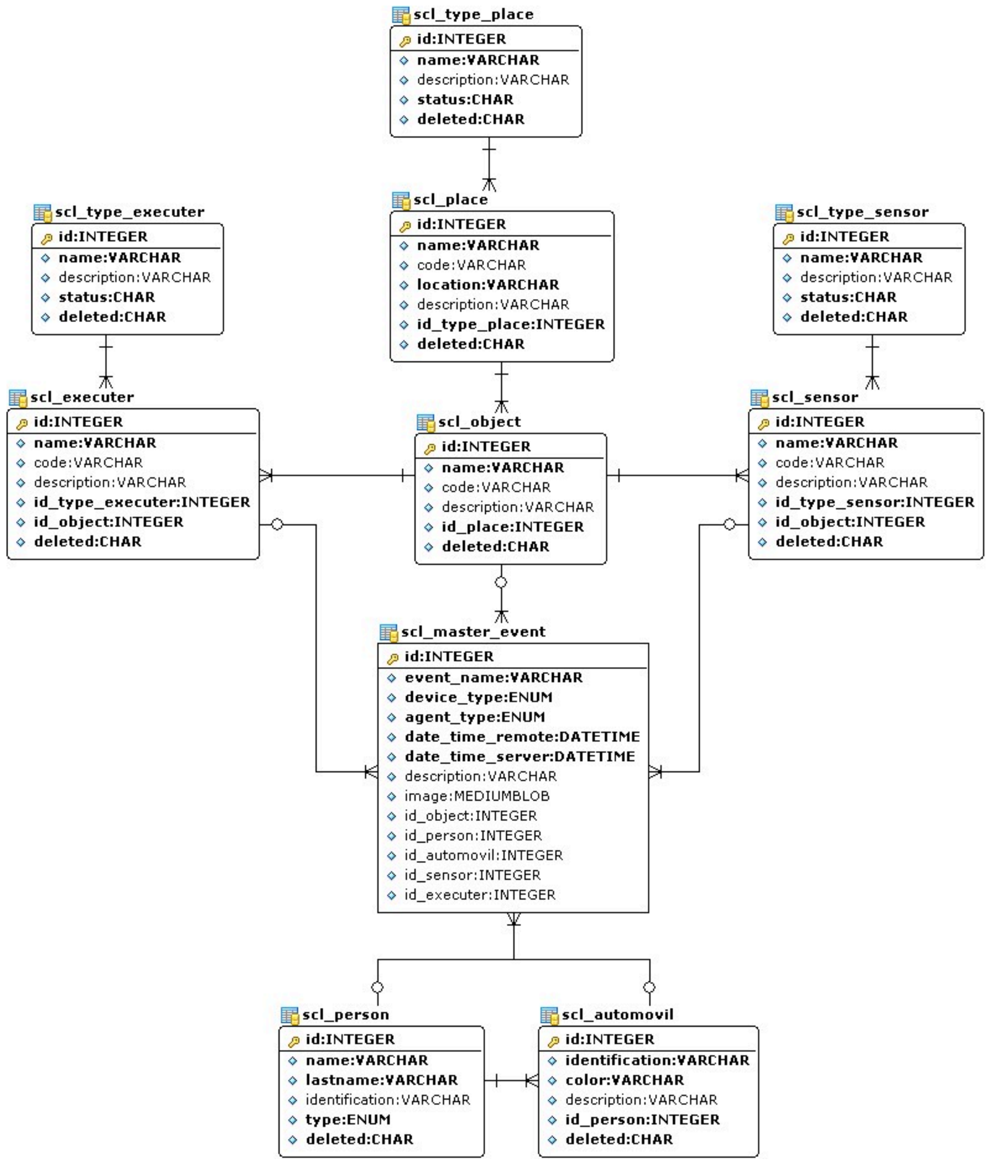


Figura 3.1.12 Diagrama Entidad Relación de la Base de Datos Centralizada.

CAPÍTULO IV

IMPLEMENTACION Y PRUEBAS DEL PROYECTO

4.1 Implementación

Procedemos a implementar el SQL para la creación de la Base de Datos, tablas y funciones en MySQL.

4.1.1 Creación de la Base de Datos:

```
DROP DATABASE IF EXISTS `central_base`;
CREATE DATABASE `central_base`
   CHARACTER SET 'utf8'
   COLLATE 'utf8_general_ci';
USE `central_base`;
```

4.1.2 Creación de la tabla scl_type_place:

```
DROP TABLE IF EXISTS `scl_type_place`;
CREATE TABLE `scl_type_place` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `description` varchar(150) default NULL,
  `status` char(1) NOT NULL default 'A',
```

```
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.3 Creación de la tabla scl_place:

```
DROP TABLE IF EXISTS `scl_place`;
CREATE TABLE `scl_place` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `code` varchar(50) default NULL,
  `location` varchar(50) NOT NULL,
  `description` varchar(150) default NULL,
  `id_type_place` int(11) NOT NULL,
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`),
  KEY `id_type_place` (`id_type_place`),
  CONSTRAINT  `scl_place_ibfk_1`  FOREIGN  KEY  (`id_type_place`)
REFERENCES `scl_type_place` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.4 Creación de la tabla scl_object:

```
DROP TABLE IF EXISTS `scl_object`;
```

```
CREATE TABLE `scl_object` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `code` varchar(50) default NULL,
  `description` varchar(150) default NULL,
  `id_place` int(11) NOT NULL,
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`),
  KEY `id_place` (`id_place`),
  CONSTRAINT `scl_object_ibfk_1` FOREIGN KEY (`id_place`)
REFERENCES `scl_place` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.5 Creación de la tabla scl_type_executer:

```
DROP TABLE IF EXISTS `scl_type_executer`;
CREATE TABLE `scl_type_executer` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `description` varchar(150) default NULL,
  `status` char(1) NOT NULL default 'A',
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.6 Creación de la tabla scl_type_sensor:

```
DROP TABLE IF EXISTS `scl_type_sensor`;
CREATE TABLE `scl_type_sensor` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `description` varchar(150) default NULL,
  `status` char(1) NOT NULL default 'A',
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.7 Creación de la tabla scl_executer:

```
DROP TABLE IF EXISTS `scl_executer`;
CREATE TABLE `scl_executer` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `code` varchar(50) default NULL,
  `description` varchar(150) default NULL,
  `id_type_executer` int(11) NOT NULL,
```

```
  `id_object` int(11) NOT NULL,
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`),
  KEY `id_type_executer` (`id_type_executer`),
  KEY `id_thing` (`id_object`),
  CONSTRAINT `scl_executer_ibfk_1` FOREIGN KEY (`id_type_executer`) REFERENCES `scl_type_executer` (`id`),
  CONSTRAINT `scl_executer_ibfk_4` FOREIGN KEY (`id_object`) REFERENCES `scl_object` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.8 Creación de la tabla scl_sensor:

```
DROP TABLE IF EXISTS `scl_sensor`;
CREATE TABLE `scl_sensor` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `code` varchar(50) default NULL,
  `description` varchar(150) default NULL,
  `id_type_sensor` int(11) NOT NULL,
  `id_object` int(11) NOT NULL,
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`),
```

```
  KEY `id_type_sensor` (`id_type_sensor`),
  KEY `id_thing` (`id_object`),
  CONSTRAINT `scl_sensor_ibfk_1` FOREIGN KEY
(`id_type_sensor`) REFERENCES `scl_type_sensor` (`id`),
  CONSTRAINT `scl_sensor_ibfk_4` FOREIGN KEY (`id_object`)
REFERENCES `scl_object` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.9 Creación de la tabla scl_person:

```
DROP TABLE IF EXISTS `scl_person`;
CREATE TABLE `scl_person` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `lastname` varchar(50) NOT NULL,
  `identification` varchar(50) default NULL,
  `type`
enum('ESTUDIANTE','PROFESOR','ADMINISTRATIVA','RESIDENTE'
,'VISITANTE') NOT NULL,
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`),
  KEY `id_type_person` (`type`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.10 Creación de la tabla scl_automovil:

```
DROP TABLE IF EXISTS `scl_automovil`;
CREATE TABLE `scl_automovil` (
  `id` int(11) NOT NULL auto_increment,
  `identification` varchar(7) NOT NULL,
  `color` varchar(50) NOT NULL,
  `description` varchar(150) default NULL,
  `id_person` int(11) NOT NULL,
  `deleted` char(1) NOT NULL default '0',
  PRIMARY KEY  (`id`),
  KEY `id_person` (`id_person`),
  CONSTRAINT  `scl_automovil_ibfk_1`  FOREIGN  KEY  (`id_person`)
REFERENCES `scl_person` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.11 Creación de la tabla scl_master_event:

```
DROP TABLE IF EXISTS `scl_master_event`;
CREATE TABLE `scl_master_event` (
  `id` int(11) NOT NULL auto_increment,
  `event_name` varchar(50) NOT NULL,
  `device_type` enum('SENSOR','EXECUTER') NOT NULL,
  `agent_type` enum('PERSON','OBJECT','AUTOMOVIL') NOT NULL,
```

```
  `date_time_remote` datetime NOT NULL,
  `date_time_server` datetime NOT NULL,
  `description` varchar(150) default NULL,
  `image` mediumblob,
  `id_object` int(11) default NULL,
  `id_person` int(11) default NULL,
  `id_automovil` int(11) default NULL,
  `id_sensor` int(11) default NULL,
  `id_executer` int(11) default NULL,
  PRIMARY KEY  (`id`),
  KEY `id_thing` (`id_object`),
  KEY `id_sensor_thing` (`id_sensor`),
  KEY `id_executer_thing` (`id_executer`),
  KEY `id_person` (`id_person`),
  KEY `id_automovil` (`id_automovil`),
  CONSTRAINT `scl_master_event_ibfk_1` FOREIGN KEY
(`id_object`) REFERENCES `scl_object` (`id`),
  CONSTRAINT `scl_master_event_ibfk_2` FOREIGN KEY
(`id_person`) REFERENCES `scl_person` (`id`),
  CONSTRAINT `scl_master_event_ibfk_3` FOREIGN KEY
(`id_automovil`) REFERENCES `scl_automovil` (`id`),
```

```
 CONSTRAINT `scl_master_event_ibfk_4` FOREIGN KEY
(`id_sensor`) REFERENCES `scl_sensor` (`id`),
 CONSTRAINT `scl_master_event_ibfk_5` FOREIGN KEY
(`id_executer`) REFERENCES `scl_executer` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.1.12 Creación de la función getDataAgent:

```
DROP FUNCTION IF EXISTS `getDataAgent`;
CREATE FUNCTION `getDataAgent`(type_agent VARCHAR(10),
id_person INTEGER(11), id_object INTEGER(11), id_automovil
INTEGER(11), type_device VARCHAR(8))
   RETURNS varchar(400)
   NOT DETERMINISTIC
   SQL SECURITY DEFINER
   COMMENT ''
BEGIN
   declare answer varchar(400);
   declare device varchar(30);

   if (type_device = 'SENSOR') then
     select ' \nha activado el sensor.' into device;
   elseif (type_device = 'EXECUTER') then
```

```
      select ' \nha ejecutado el actuador.' into device;
    end if;

    if (type_agent = 'PERSON') then
      select
          concat(sp.type,', ',sp.lastname,' ',sp.name,' \ncon identificación
(',sp.identification,')',device)  into answer
      from
          scl_person sp
      where
          sp.id=id_person;
    elseif (type_agent = 'OBJECT') then
      select
          concat('OBJETO, ',so.name,' (cod. ',so.code,') \nubicado en
',sp.name,' (cod. ',sp.code,'), \n',sp.location,device)  into answer
      from
          scl_object so
      inner join
          scl_place sp on so.id_place = sp.id
      where
          so.id=id_object;
    elseif (type_agent = 'AUTOMOVIL') then
```

```
select
    concat('AUTOMOVIL, con placa ',sa.identification,' de color ',sa.color,' \npropiedad de ',sp.lastname,' ',sp.name,device) into answer
from
    scl_automovil sa
inner join
    scl_person sp on sa.id_person=sp.id
where
    sa.id=id_automovil;
end if;
return answer;
END;
```

4.1.13 Creación de la función getDataDevice:

```
DROP FUNCTION IF EXISTS `getDataDevice`;
CREATE FUNCTION `getDataDevice`(type_device VARCHAR(8), id_sensor INTEGER(11), id_executer INTEGER(11))
  RETURNS varchar(400)
  NOT DETERMINISTIC
  SQL SECURITY DEFINER
  COMMENT ''
```

```
BEGIN
   declare respuesta varchar(400);
   if (type_device = 'SENSOR') then
     select
         concat('El sensor ',ss.name,' (cod. ',ss.code,') \nse ha activado, este sensor está en el objeto \n',
              so.name,' (cod. ',so.code,') \nubicado en ',sp.name,' (cod. ',sp.code,'), \n',sp.location)  into respuesta
     from
         scl_sensor ss
     inner join
         scl_object so on ss.id_object=so.id
     inner join
         scl_place sp on so.id_place = sp.id
     where
         ss.id=id_sensor;
   elseif (type_device = 'EXECUTER') then
     select
         concat('El actuador ',se.name,' (cod. ',se.code,') \nse ha ejecutado, este actuador está en el objeto \n',
              so.name,' (cod. ',so.code,') \nubicado en ',sp.name,' (cod. ',sp.code,'), \n',sp.location)  into respuesta
```

```
    from
        scl_executer se
    inner join
        scl_object so on se.id_object=so.id
    inner join
        scl_place sp on so.id_place = sp.id
    where
        se.id=id_executer;
  end if;
  return respuesta;
END;
```

4.1.2 Programación en LabVIEW

En LabVIEW se ha desarrollado, algunas interfaces para el mantenimiento de las entidades en relación a la Base de Datos. Entre estas entidades tenemos:

- Entidad type_place
- Entidad place
- Entidad object
- Entidad type_sensor
- Entidad sensor
- Entidad type_executer

- Entidad executer
- Entidad person
- Entidad automóvil
- Entidad master_event

4.1.2.1 Entidad type_place

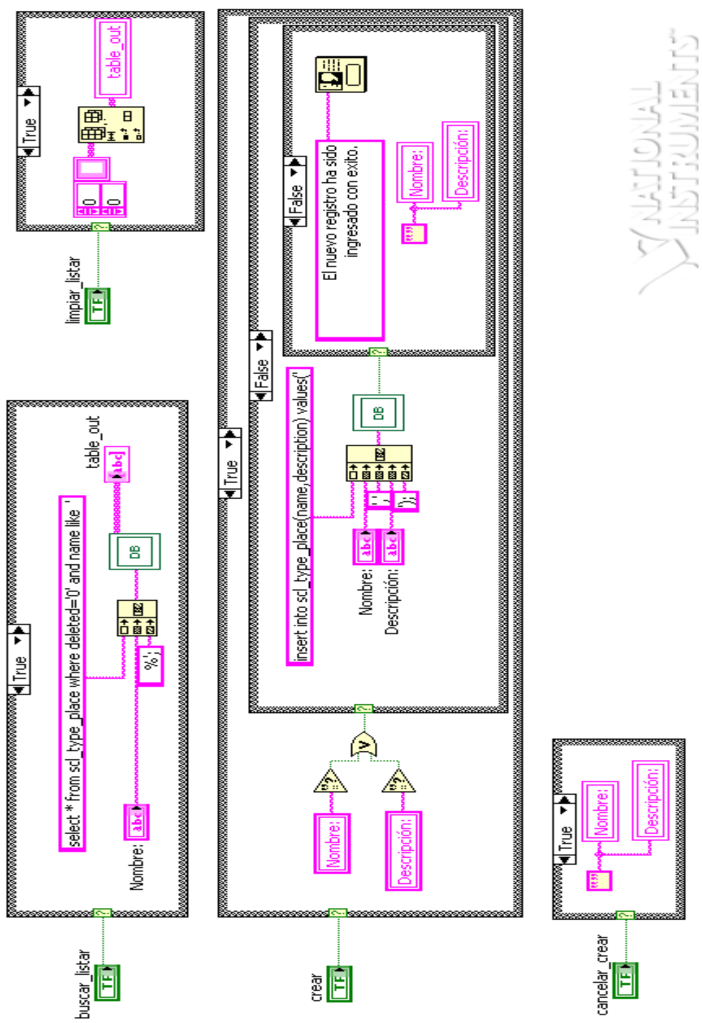


Figura 4.1.2.1.1 – Diagrama de Bloques de la Entidad type_place (parte 1)

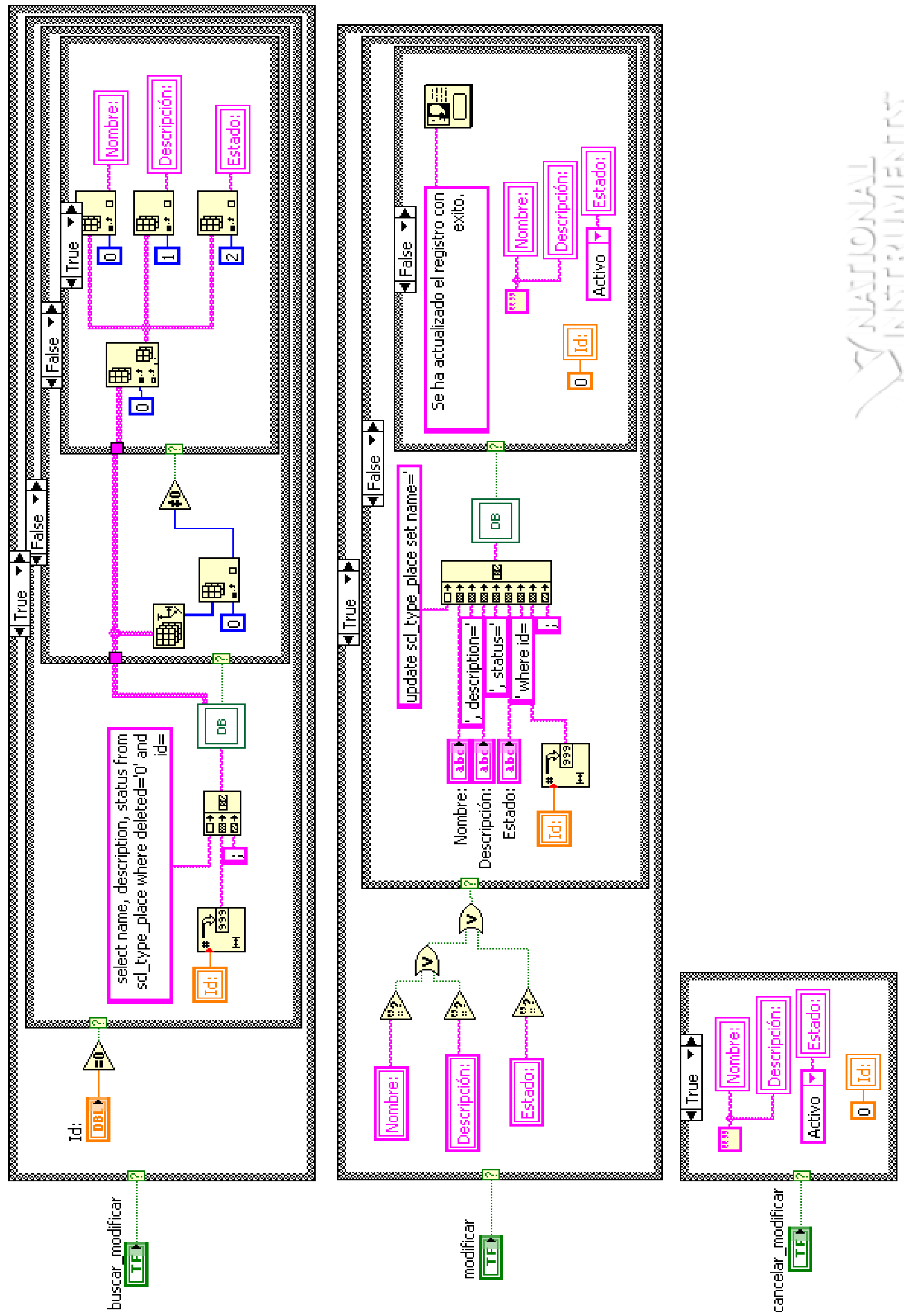


Figura 4.1.2.1.2 – Diagrama de Bloques de la Entidad type_place (parte 2)

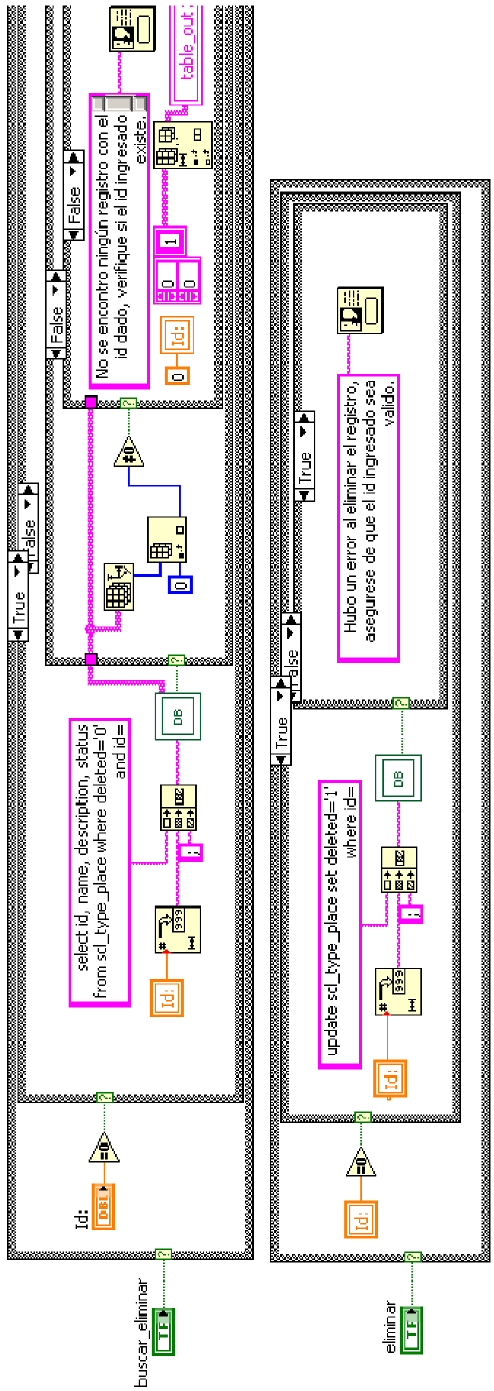


Figura 4.1.2.1.3 – Diagrama de Bloques de la Entidad type_place (parte 3)

4.1.2.2 Entidad place

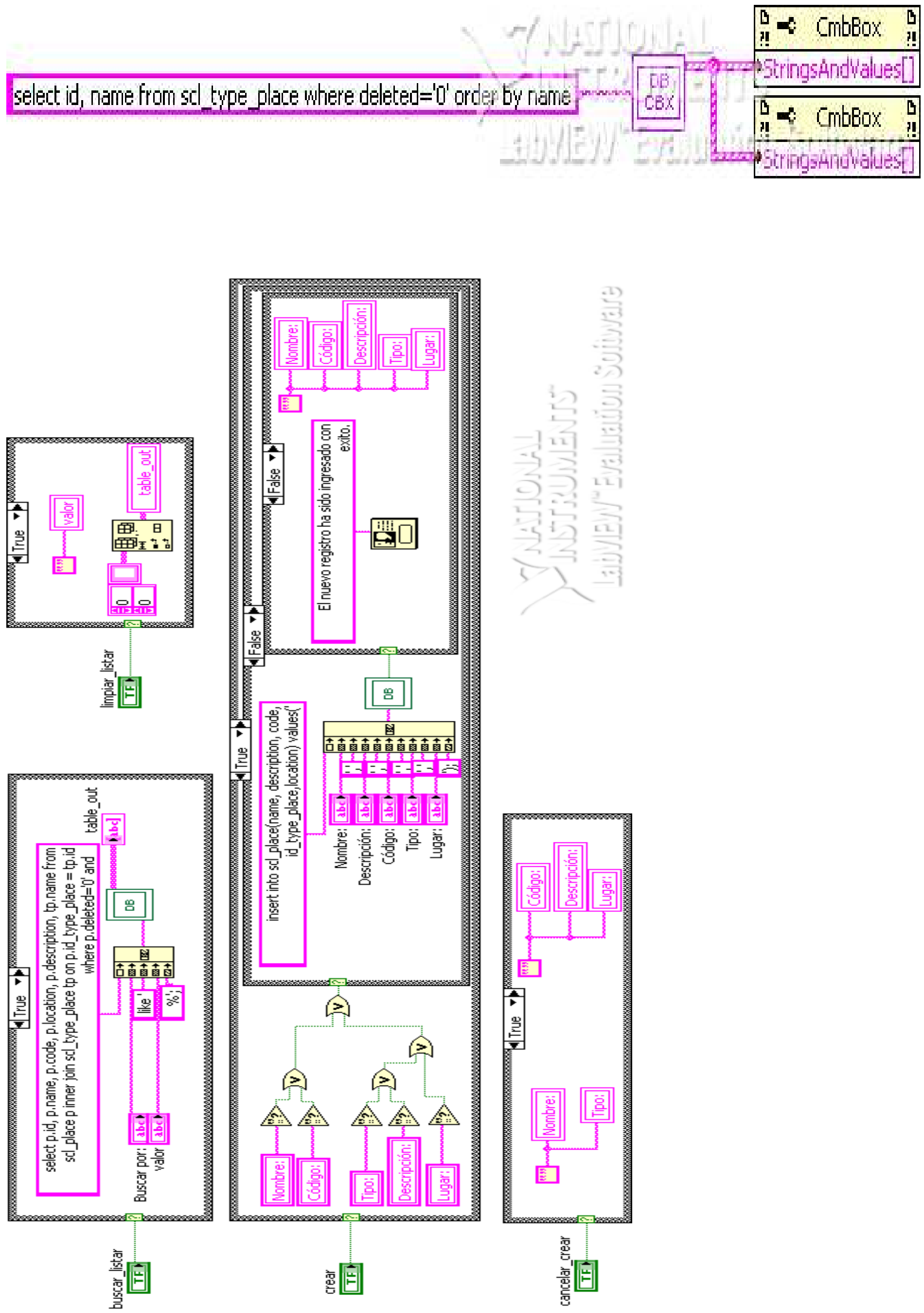


Figura 4.1.2.2.1 – Diagrama de Bloques de la Entidad place (parte 1)

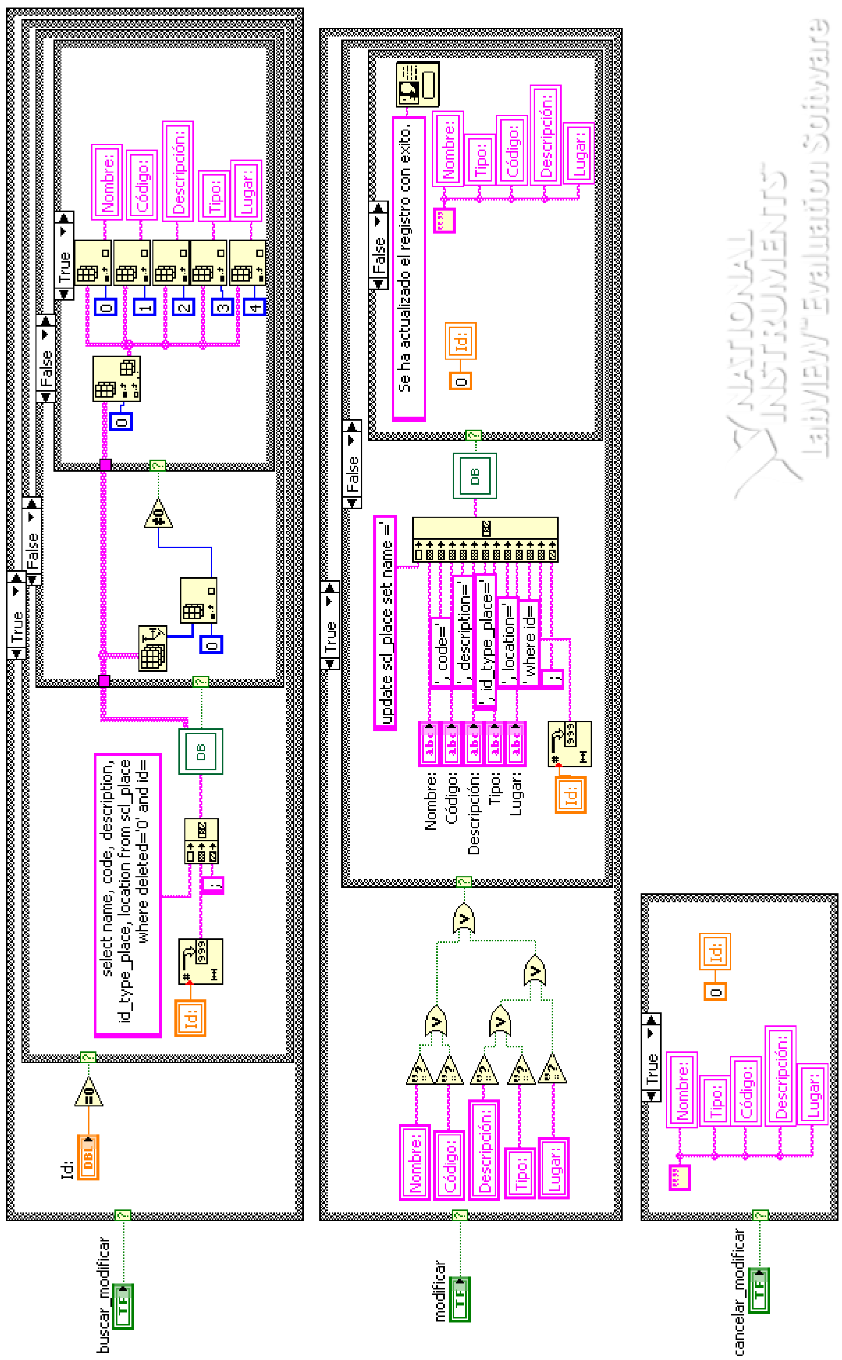


Figura 4.1.2.2.2 – Diagrama de Bloques de la Entidad place (parte 2)

4.1.2.3 Entidad object

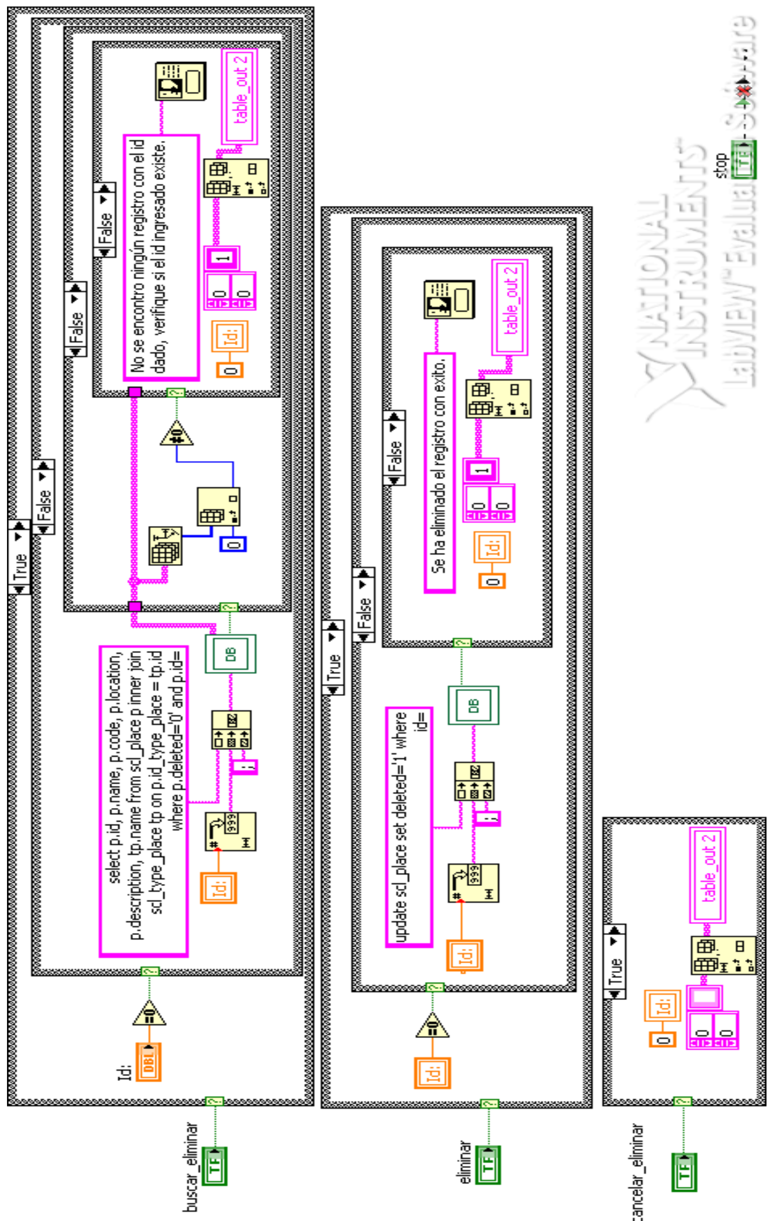


Figura 4.1.2.3.1 – Diagrama de Bloques de la Entidad object (parte 1)

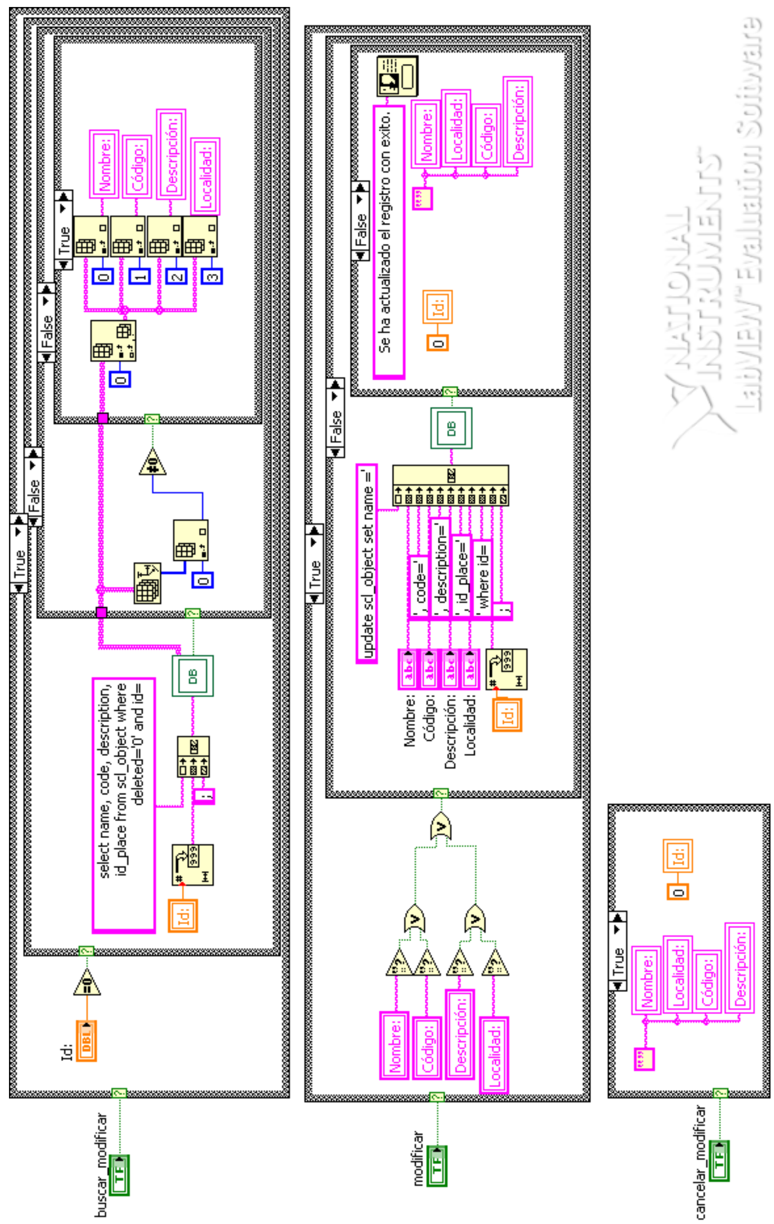


Figura 4.1.2.3.2 – Diagrama de Bloques de la Entidad object (parte 2)

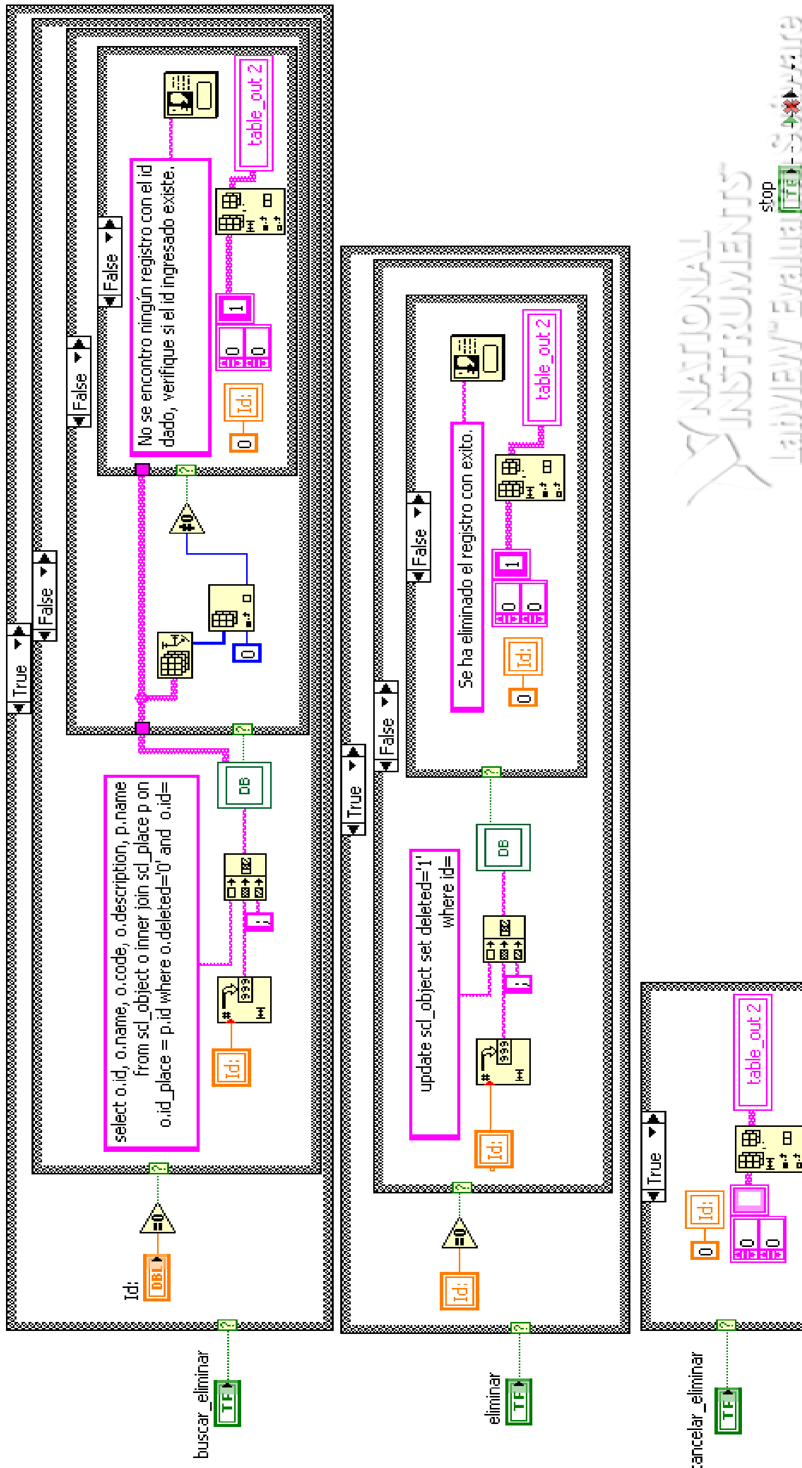


Figura 4.1.2.3.3 – Diagrama de Bloques de la Entidad object (parte 3)

4.1.2.4 Entidad type_sensor

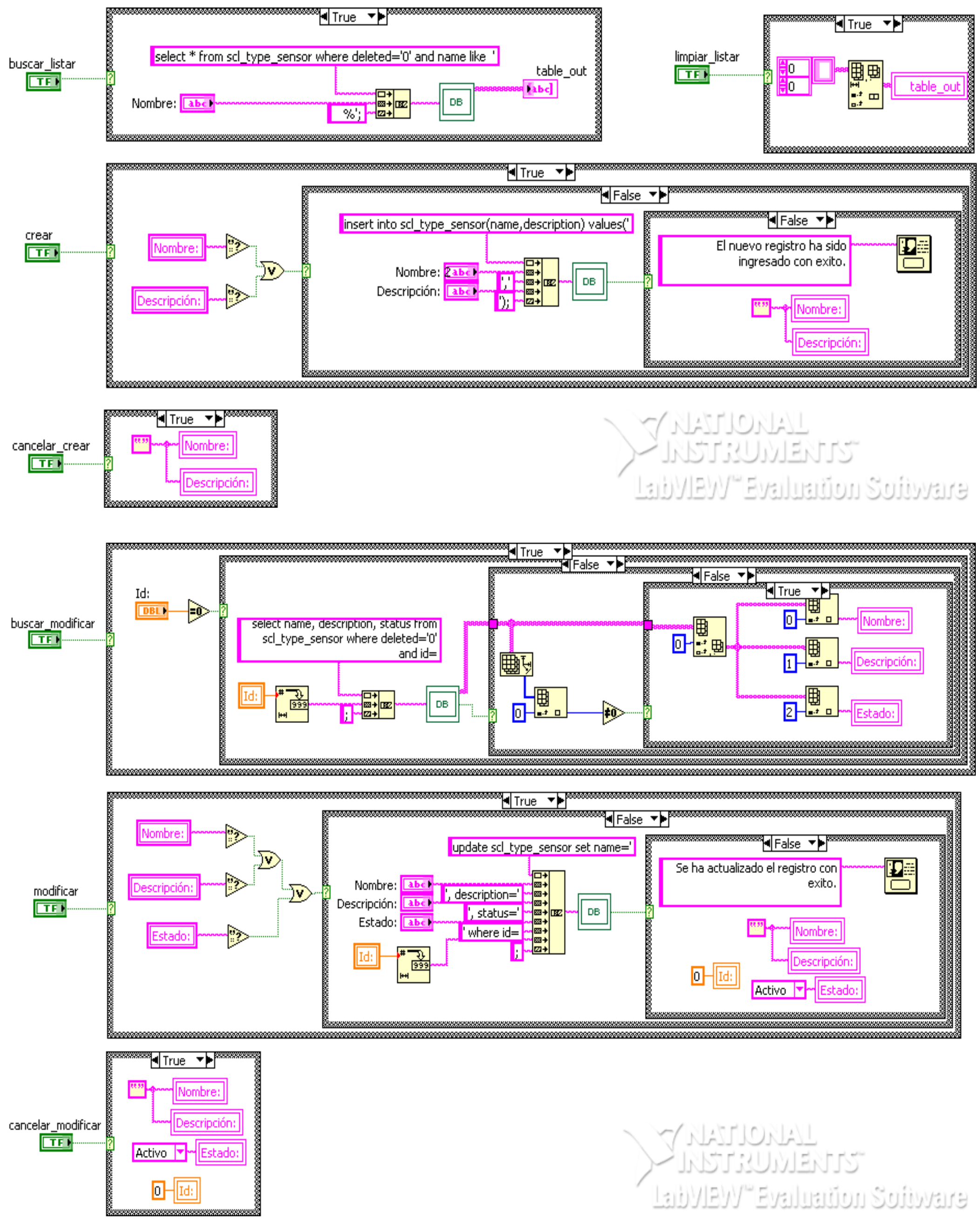


Figura 4.1.2.4.1 – Diagrama de Bloques de la Entidad type_sensor (parte 1)

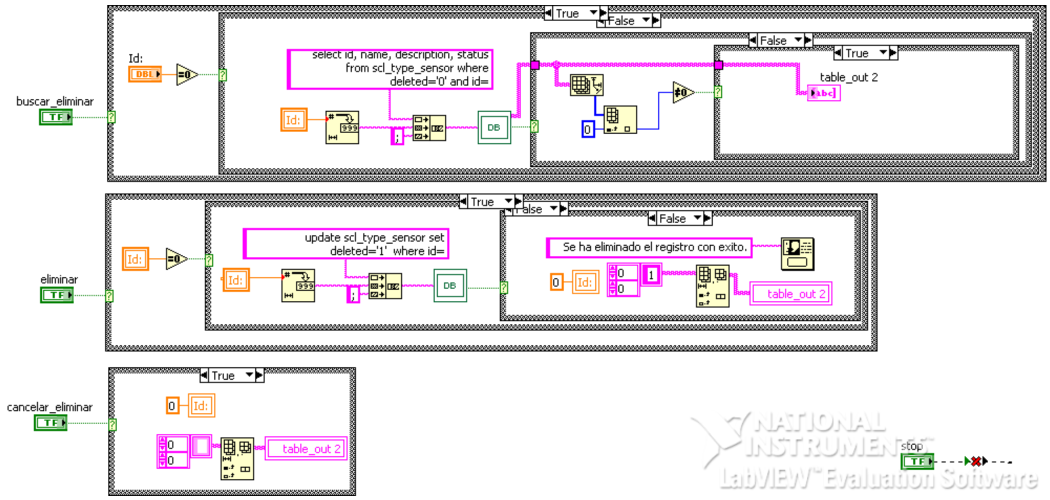


Figura 4.1.2.4.2 – Diagrama de Bloques de la Entidad type_sensor (parte 2)

4.1.2.5 Entidad sensor

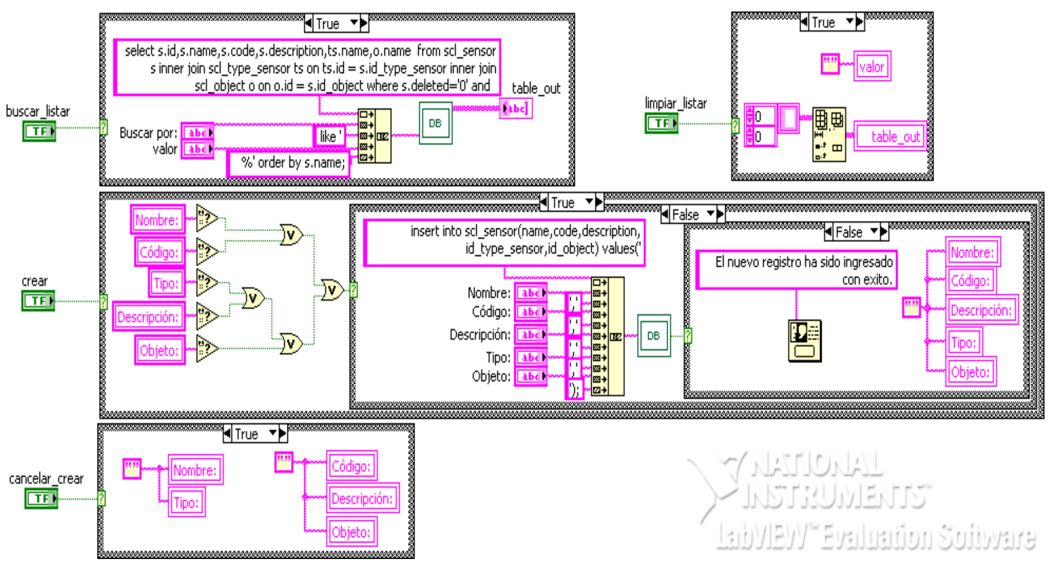


Figura 4.1.2.5.1 – Diagrama de Bloques de la Entidad sensor (parte 1)

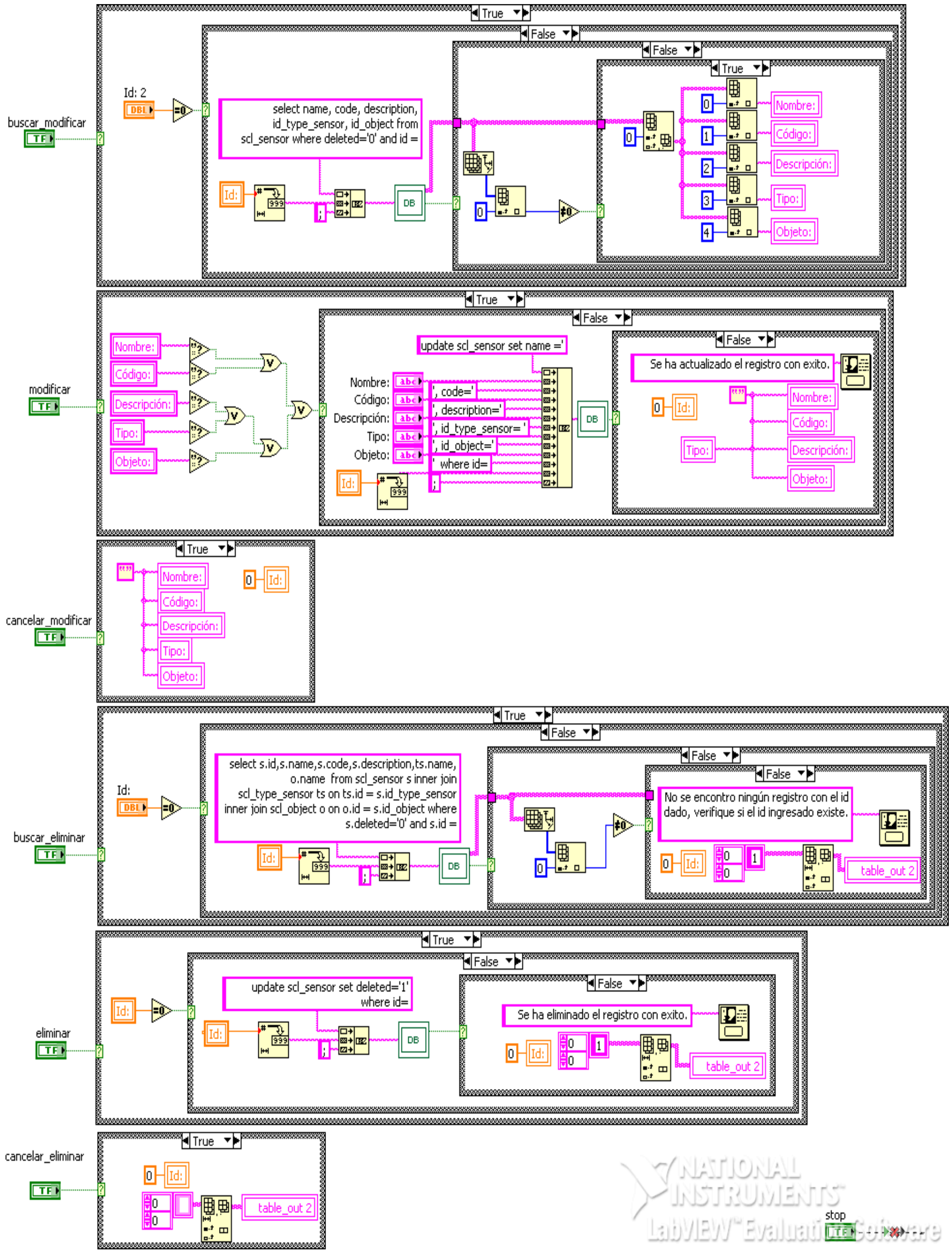


Figura 4.1.2.5.2 – Diagrama de Bloques de la Entidad sensor (parte 2)

4.1.2.6 Entidad type_executer

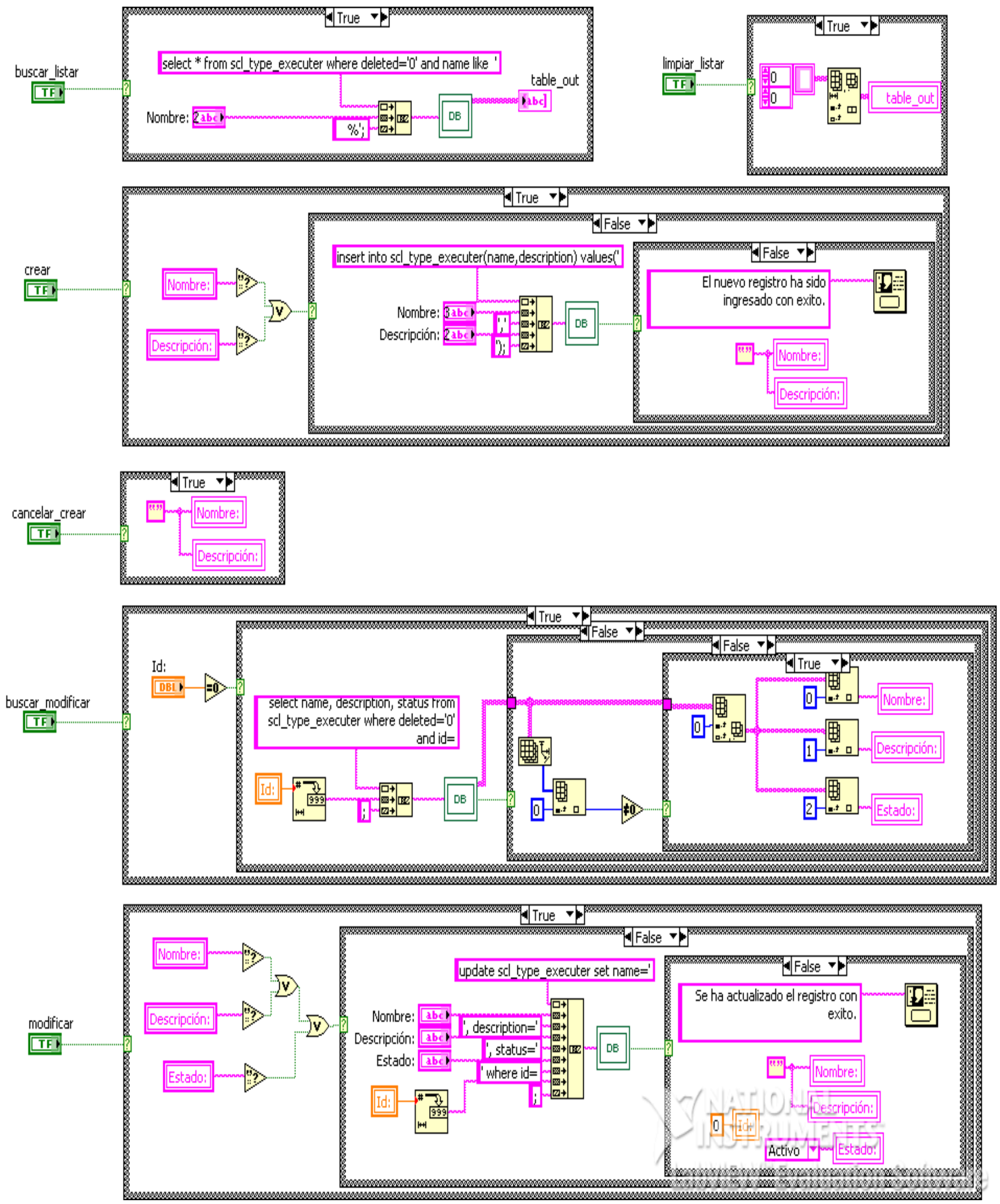


Figura 4.1.2.6.1 – Diagrama de Bloques de la Entidad type_executer (parte 1)

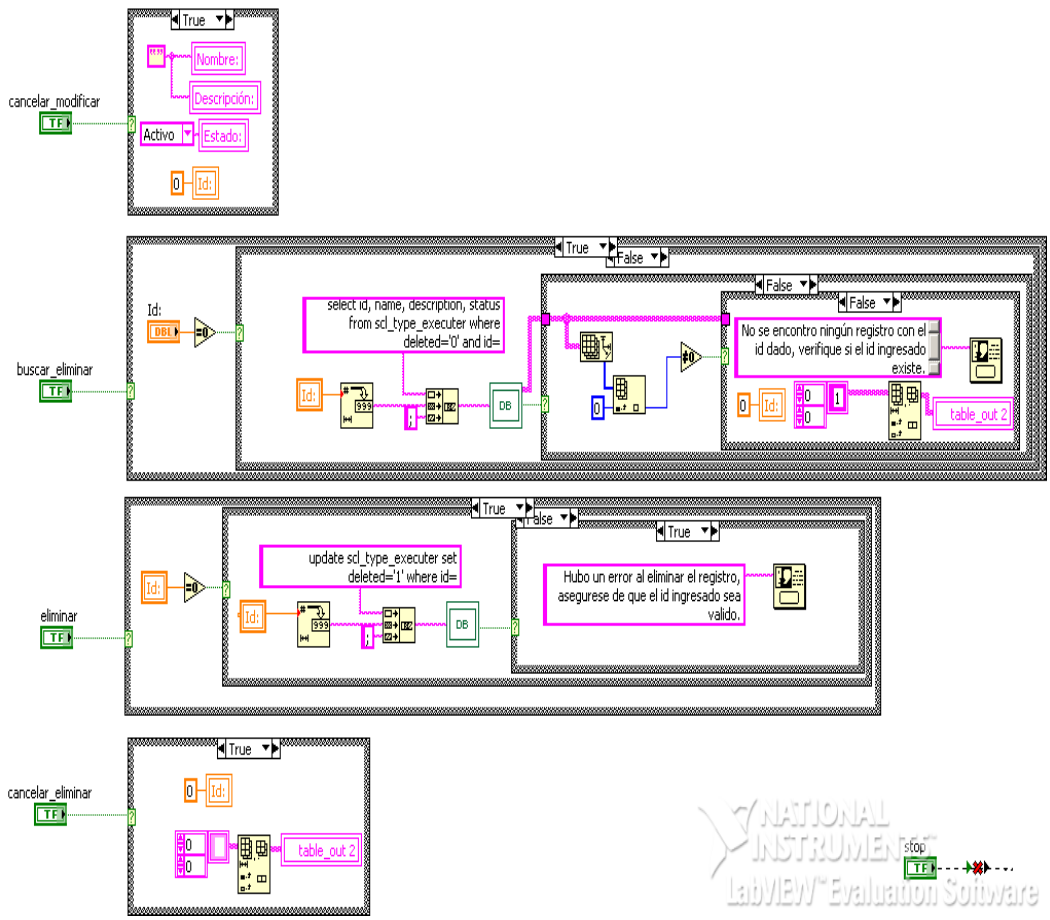


Figura 4.1.2.6.2 – Diagrama de Bloques de la Entidad type_executer (parte 2)

4.1.2.7 Entidad executer

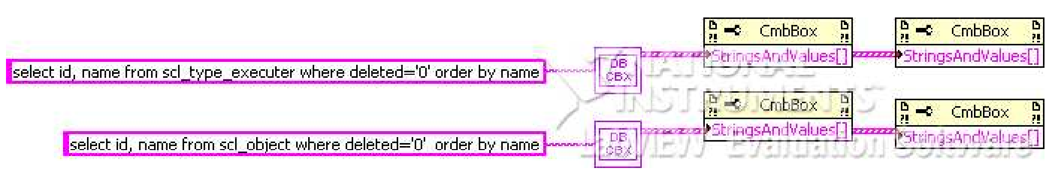


Figura 4.1.2.7.1 – Diagrama de Bloques de la Entidad executer (parte 1)

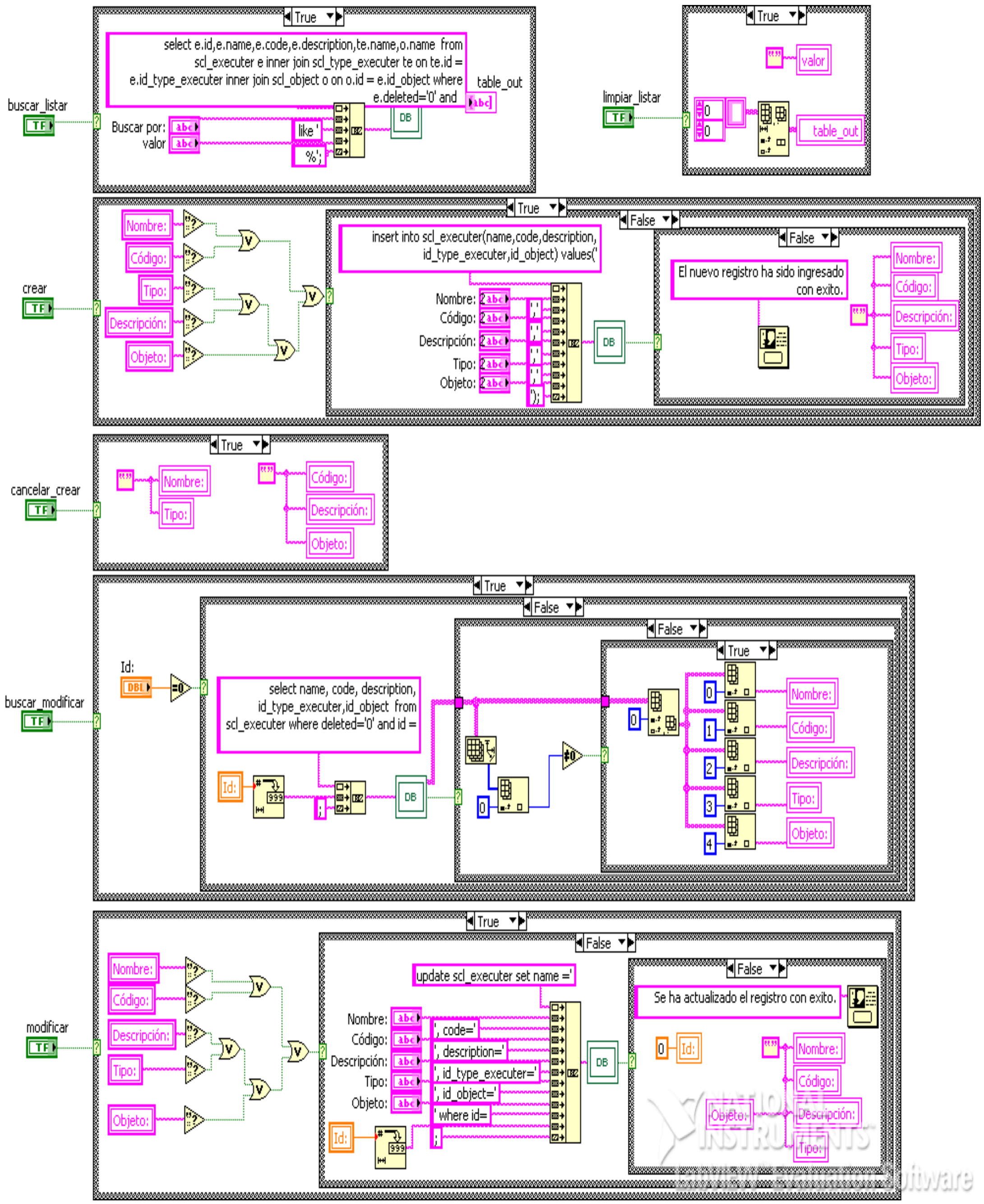


Figura 4.1.2.7.2 – Diagrama de Bloques de la Entidad executer (parte 2)

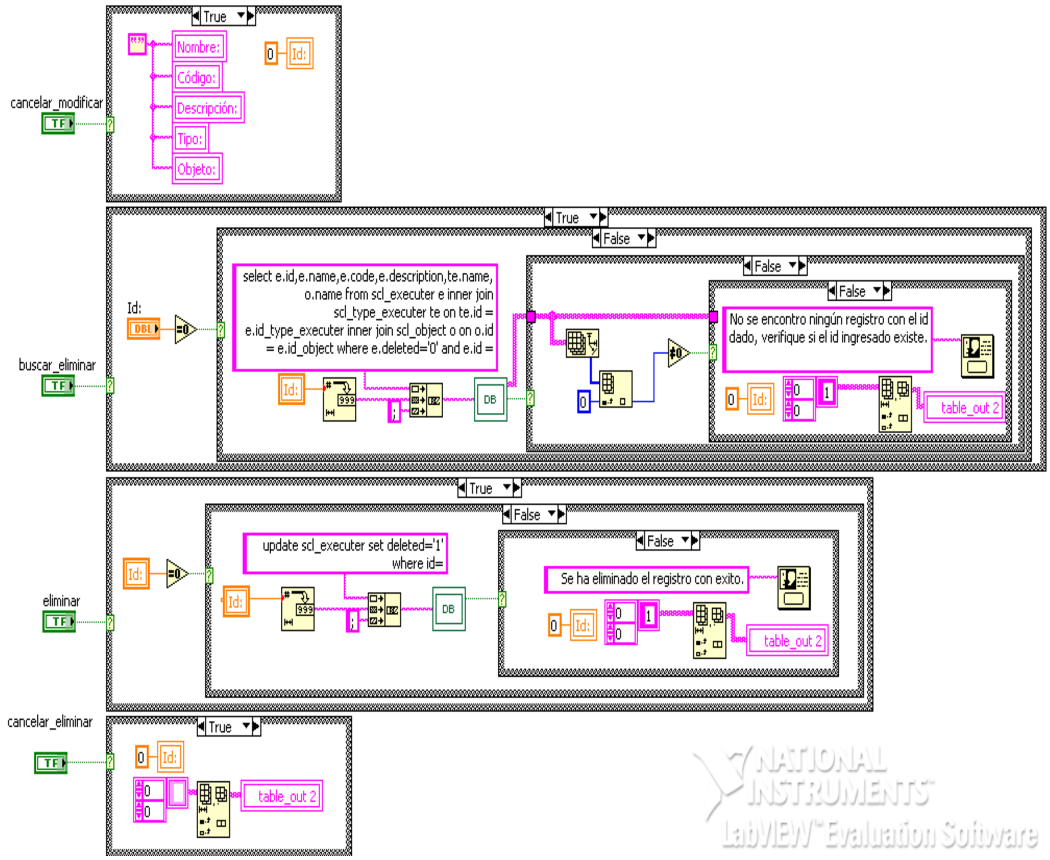


Figura 4.1.2.7.3 – Diagrama de Bloques de la Entidad executer (parte 3)

4.1.2.8 Entidad person

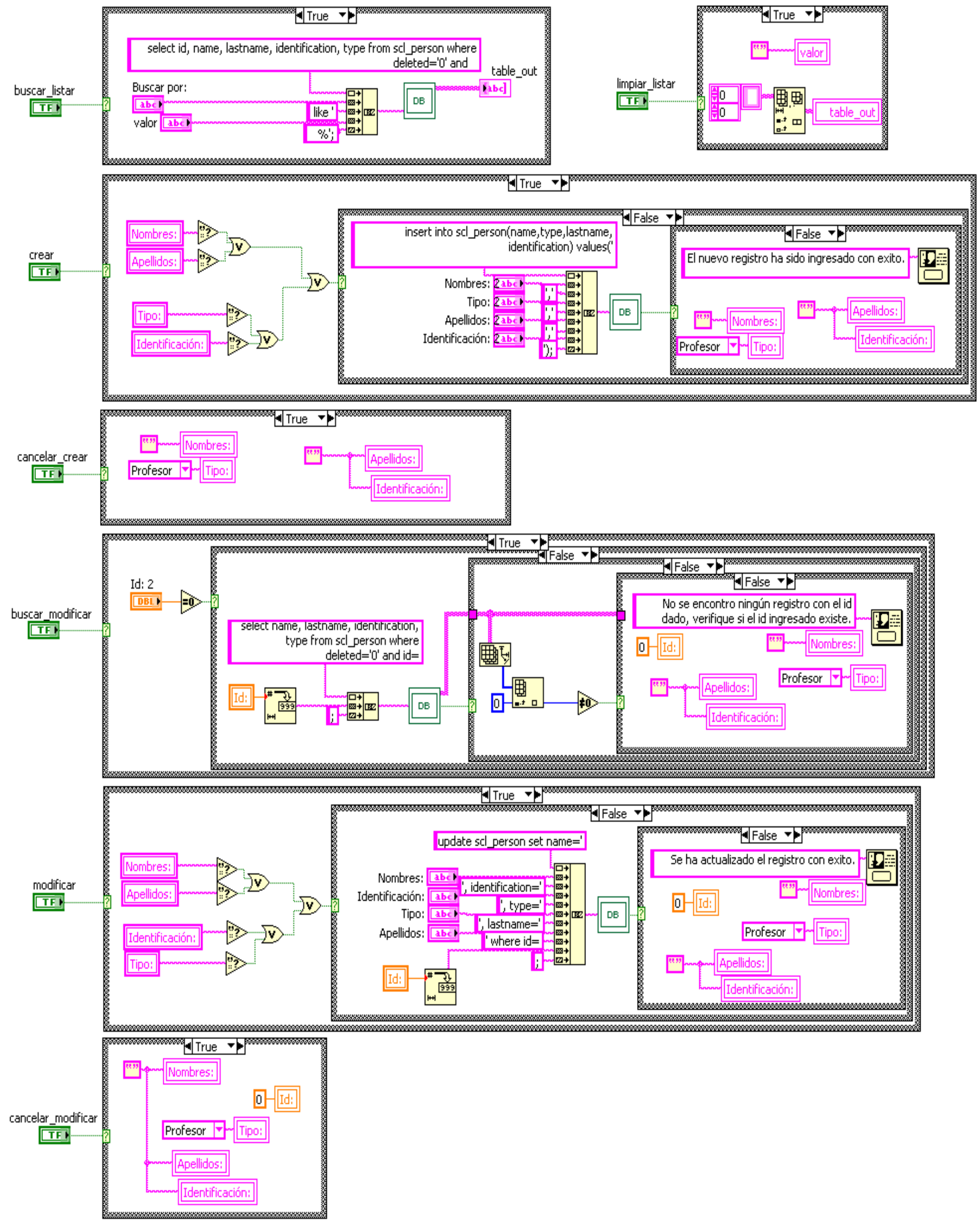


Figura 4.1.2.8.1 – Diagrama de Bloques de la Entidad person (parte 1)

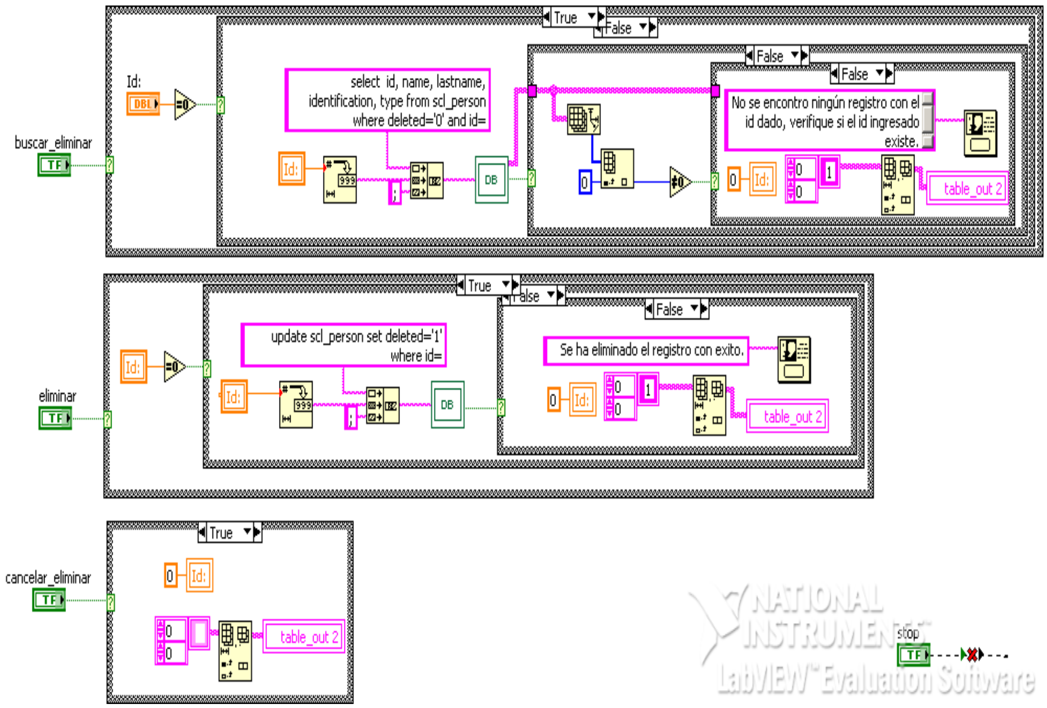


Figura 4.1.2.8.2 – Diagrama de Bloques de la Entidad person (parte 2)

4.1.2.9 Entidad automovil

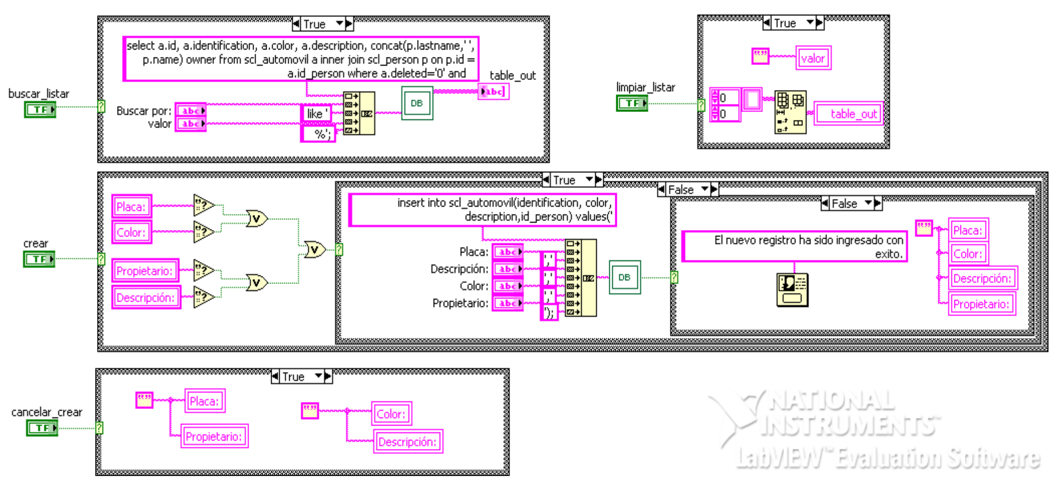


Figura 4.1.2.9.1 – Diagrama de Bloques de la Entidad automovil (parte 1)

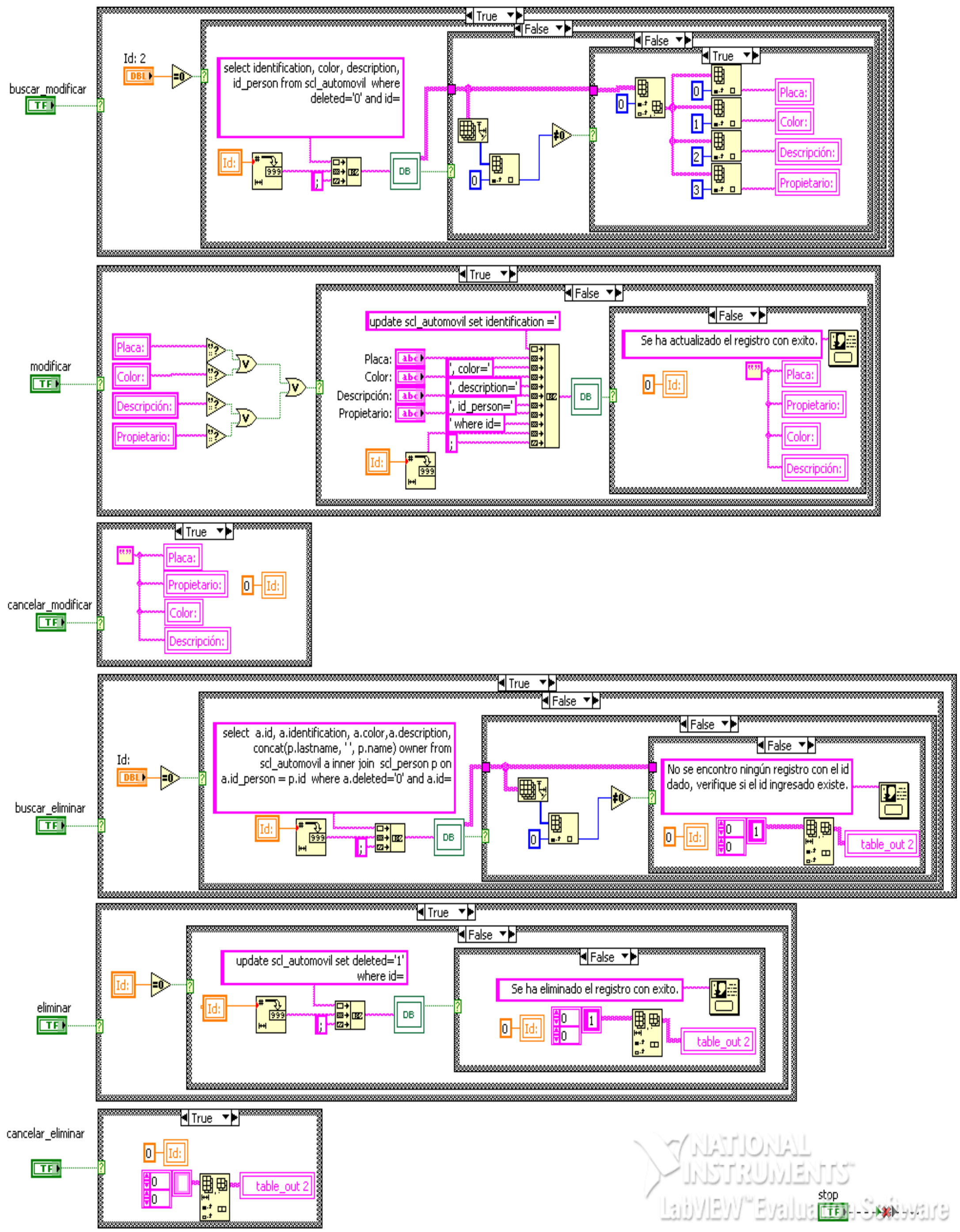


Figura 4.1.2.9.2 – Diagrama de Bloques de la Entidad automovil (parte 2)

4.1.2.10 Entidad master_event

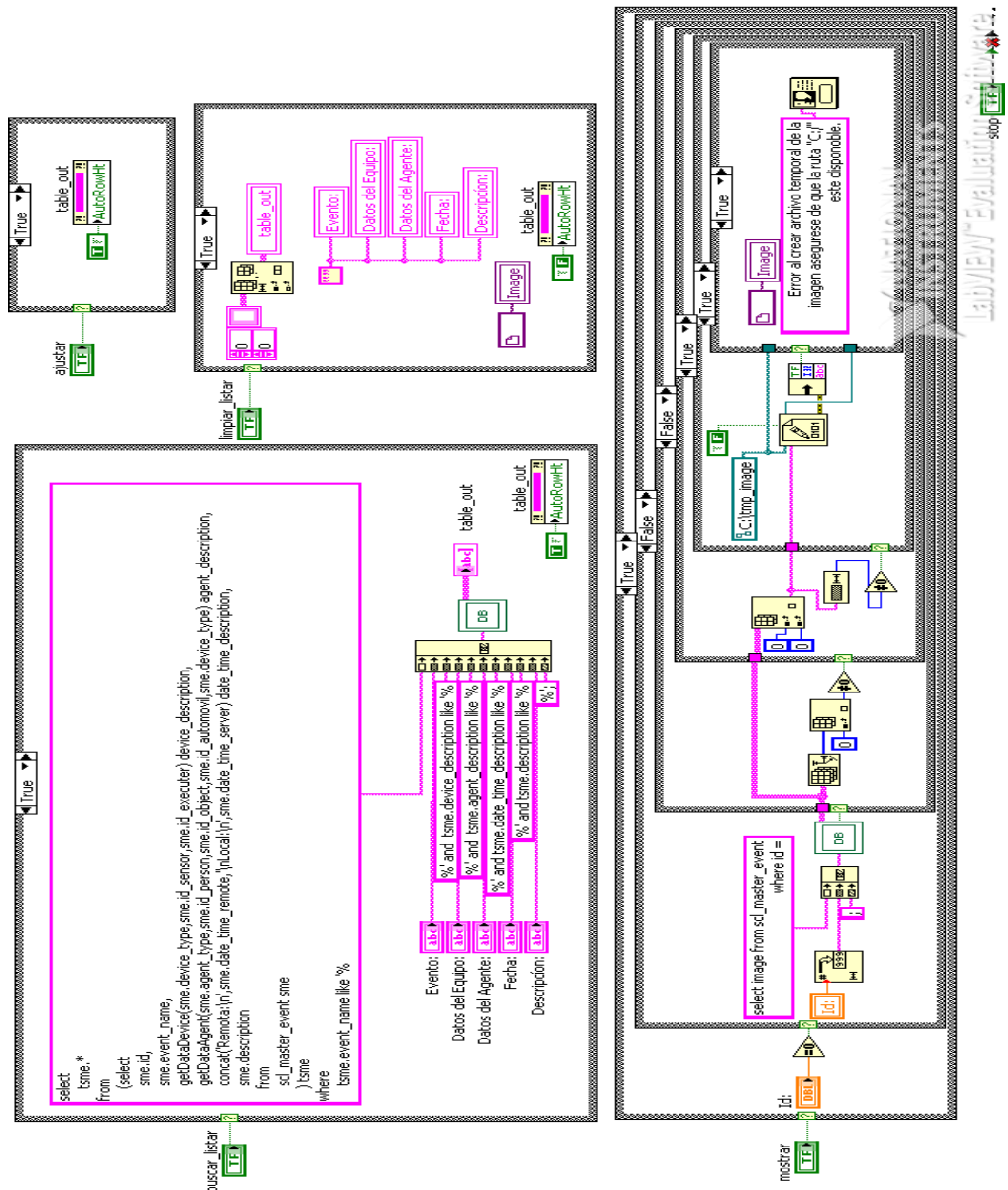


Figura 4.1.2.10.1 – Diagrama de Bloques de la Entidad master_event

4.1.3 Sent Event

Este modulo es usado para tener un estándar, forma correcta y transparencia para que los sistemas de seguridades soportados enviarán sus eventos.

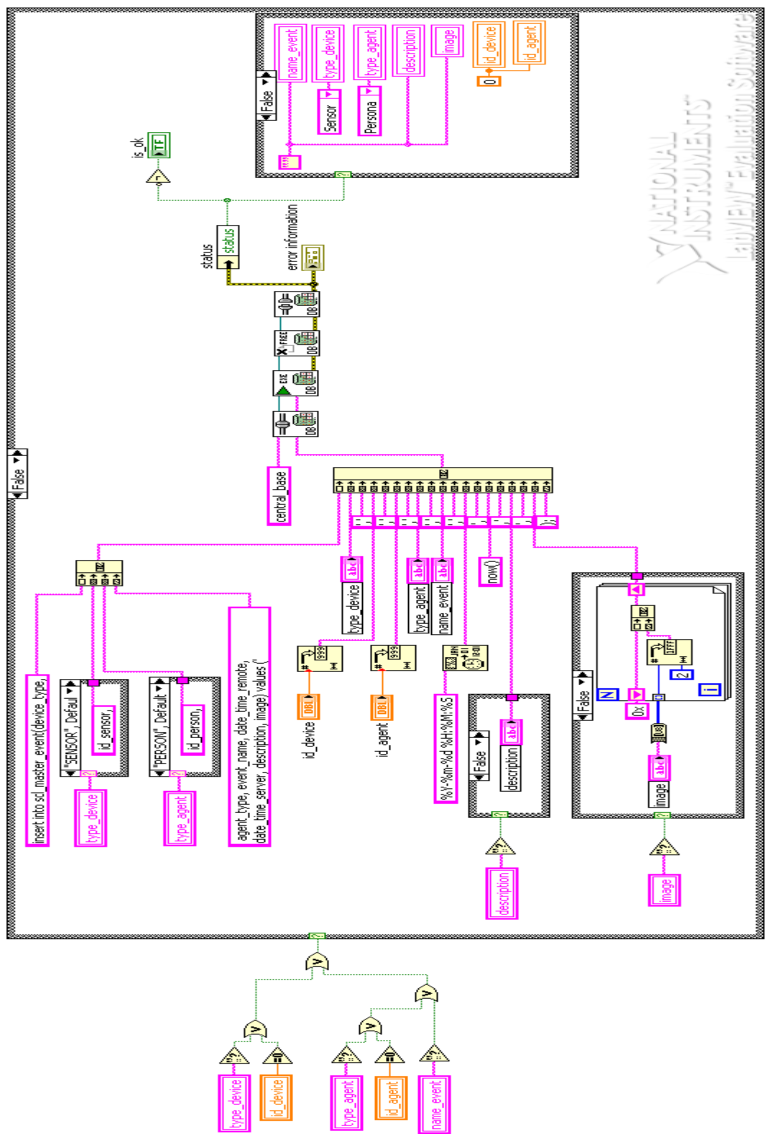


Figura 4.1.3.1 – Diagrama de Bloques de la Modulo send_event

4.2 Pruebas

Ejemplo de cómo conectar los pines del modulo send_event, en este ejemplo se está enviando un evento que ha ocurrido en la puerta de alguna localidad y el agente fue un vehículo. Se envía también una foto o imagen que el evento capturó.

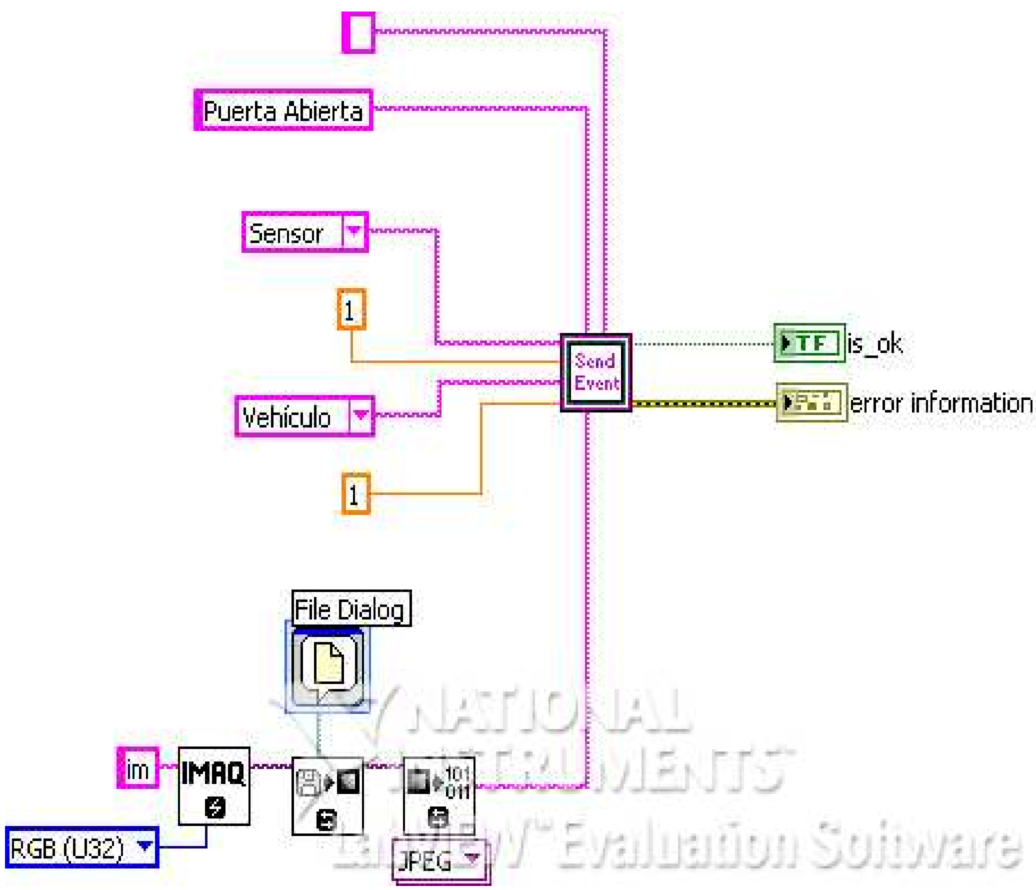


Figura 4.2.1 – Ejemplo de uso de Modulo send_event (parte 1)

Ejemplo de envio de un evento, pero se esta usando la librería de IMAQ de LabWIEW donde la imagen se la obtiene de una cámara.

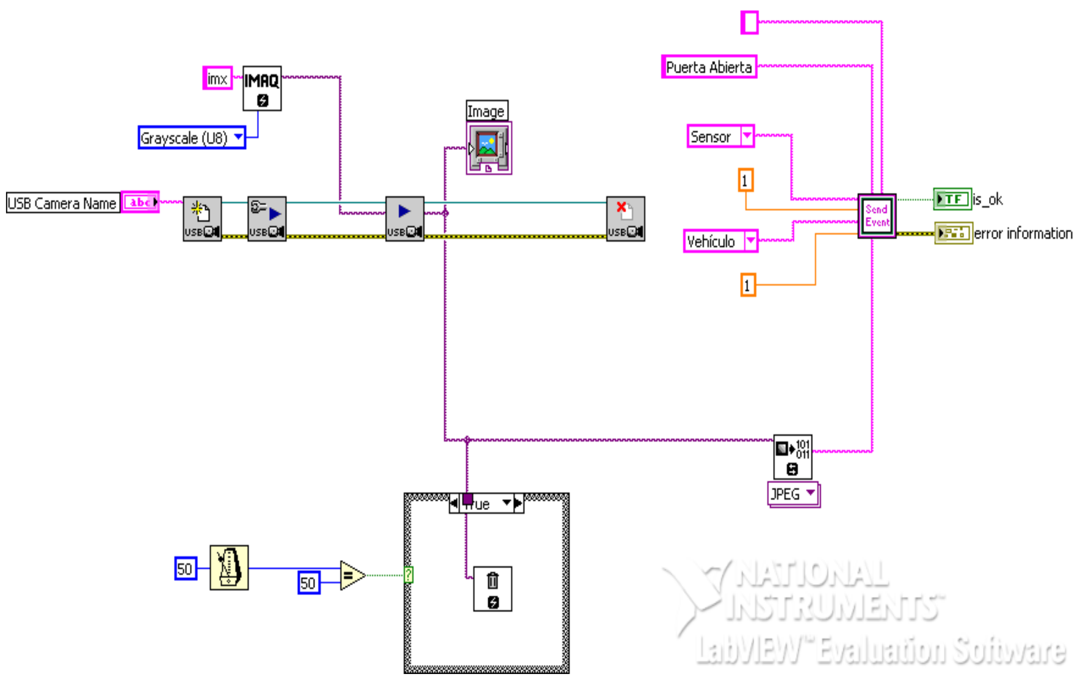


Figura 4.2.2 – Ejemplo de uso de Modulo send_event (parte 2)

CONCLUSIONES

1. Podemos concluir que estableciendo el alcance (número de proyectos soportados) y elaborando una abstracción de la lógica de los sistemas de seguridades, hemos logrado obtener un diseño genérico de base de datos, capaz de soportar diferentes sistemas de seguridades.

2. Algo de notar es que el diseño obtenido se basa en los equipos o terminales (sensores y actuadores) haciendo esto posible la adaptación de dicho diseño a un sistema de seguridad no soportado, es decir el diseño está abierto para otros sistemas de seguridades similares.

3. Podemos asegurar que este proyecto ha cumplido con su objetivo principal el cual era de Diseñar una Base de Datos Centralizada para Administrar los datos adquiridos por los diferentes Sistemas de Seguridad, de esta materia de graduación, mediante una interfaz gráfica que desarrollamos en LabVIEW, esto se logro debido a su modularización lo cual disminuirá considerablemente las actualizaciones y mantenimientos futuros, su gran escalabilidad, fácil entendimiento y manipulación.

4. Mediante el uso de la Arquitectura MCV2, arquitectura de desarrollo aplicado para sitios web, pudimos llevar al lenguage G de LabView a que cumpla este patron de arquitectura, haciendo posible la escalabilidad, expansión y adaptación de futuros desarrollos para mejoras del sistema.

RECOMENDACIONES

1. Como una de las recomendaciones, para aquellos Sistemas De Seguridad que se integren con el Sistema descrito, se debe monitorear en tiempos regulares que la conexión de red este disponible, si llegase a fallar la conexión de red, es recomendable que estos sistemas implementen un mecanismo de reenvió de los Eventos que no se pudieron enviar en el tiempo que estuvo caído el enlace de red para poder hacer este mecanismo de reenvió, estos sistemas deben poseer una Base de Datos Local como soporte de Seguridad.

2. También es recomendable que cuando esto pase estos Sistemas de Seguridad sean capaces de avisar por algún medio de comunicación que el enlace esta caído y así se le de la asistencia Inmediata.

3. Es aconsejable que dado el crecimiento de la demanda de dispositivos móviles con grandes capacidades y facilidades de navegación en Internet y aprovechando que la interfaz de nuestro Proyecto puede ser accesada vía web desde cualquier equipo que se conecte al Internet se desarrolle la versión para dispositivos móviles para Supervisión de Sistemas de Seguridad.

BIBLIOGRAFÍA

- “BASE DE DATOS”, Wikipedia, disponible en:

  http://es.wikipedia.org/wiki/base_de_datos

- “MYSQL”, Wikipedia, disponible en:

  http://es.wikipedia.org/wiki/mysql

- “LABVIEW”, Wikipedia, disponible en:

  http://es.wikipedia.org/wiki/labview

- "LABVIEW DATABASE CONNECTIVITY TOOLKIT", disponible en:

  http://www.ni.com/pdf/labview/us/database_connectivity_toolkit.pdf

- “DESARROLLO TECNOLÓGICO”, disponible en:

  http://www.utmetropolitana.edu.mx/webroot/site/mi/ci/revistacontacto/ci7web.pdf

- “CENTRALIZACIÓN DEL SISTEMA DE INFORMACIÓN DE ATENCIÓN PRIMARIA: HISTORIA CLÍNICA ELECTRÓNICA INTEGRADA”, disponible en:

  http://www.csi.map.es/csi/tecnimap/tecnimap_2007/presentadas/tco-154-2007dp/comunicacion2_tco-154-2007dp.pdf

- “GFI EVENTS MANAGER”, disponible en:

  http://www.gfi.com/eventsmanager

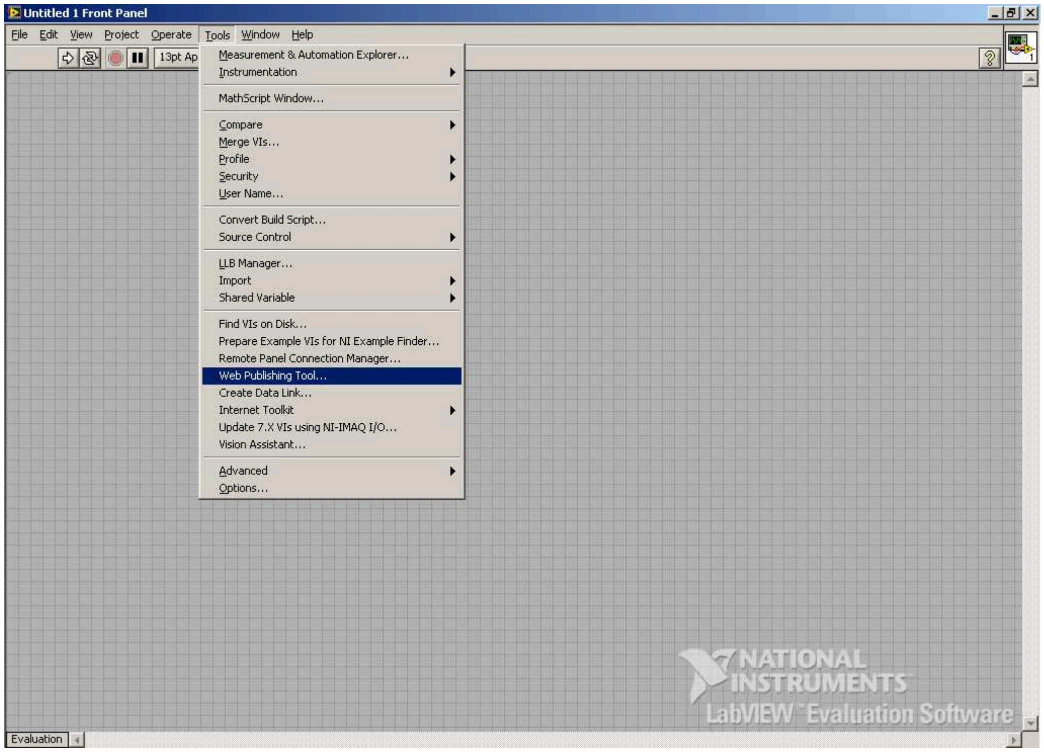
ANEXOS

MANUAL DE USUARIO

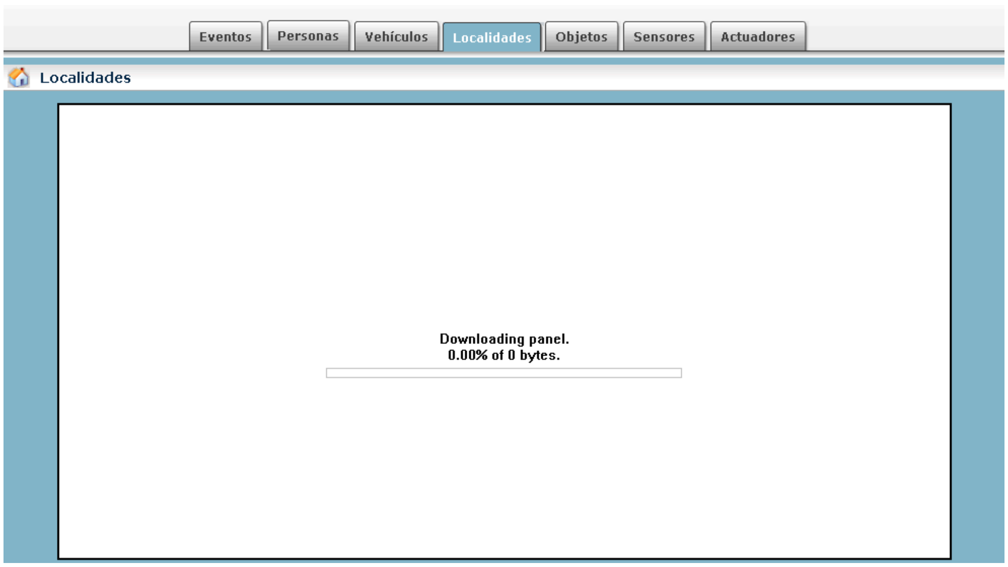
Este manual está estructurado secuencialmente siguiendo la lógica de la interfaz en LabVIEW para Administrar una Base de Datos Centralizada para Sistema de Seguridades.

El modo de administrar el aplicativo comienza por definir los tipos de localidades y sus **localidades**. Luego de esto pasamos a definir los **objetos** que pertenecen a una localidad en particular. Al hacer esto podemos asignar, si fuera necesario, **sensores** y/o **actuadores** a cada objeto. Por otra parte, se deben definir los agentes que conforman los automóviles, personas y objetos.

El aplicativo que maneja la interfaz gráfica que administra la Base de Datos Centralizada para Sistema de Seguridades ha sido diseñado utilizando la opción Web Publishing Tool del menú Tools de LabVIEW , el cual nos permite mostrar y usar en un navegador web el Front Panel de cualquier VI creado en LabVIEW.



Como se van a manejar muchas ventanas de mantenimientos y consultas, estas se agruparon en una sola página web llamada ***index.html*** la cual posee un menú principal de navegación. Para que el index.html pueda funcionar correctamente se requiere que todos los VI que son usados estén abiertos, para esto se creo un archivo llamado ***start.bat*** que mediante su ejecución abre los VI requeridos por el index.html, cada opción del menú principal nos lleva a un VI de consulta y mantenimiento de un objeto en el modelo lógico de la Base de Datos Centralizada, excepto el menú Eventos que nos lleva al VI más importante de consultas generales de los eventos y acciones de los ejecutores de los diferentes Sistemas de Seguridades.

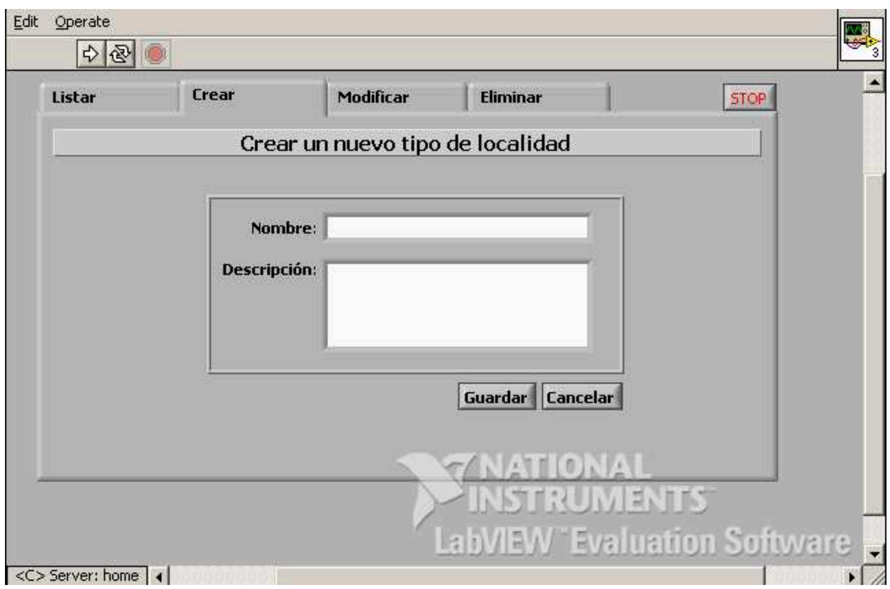

Eventos
Personas
Vehículos
Localidades
Objetos
Sensores
Actuadores
Localidades
Downloading panel.
0.00% of 0 bytes.

1.- Menú Localidades

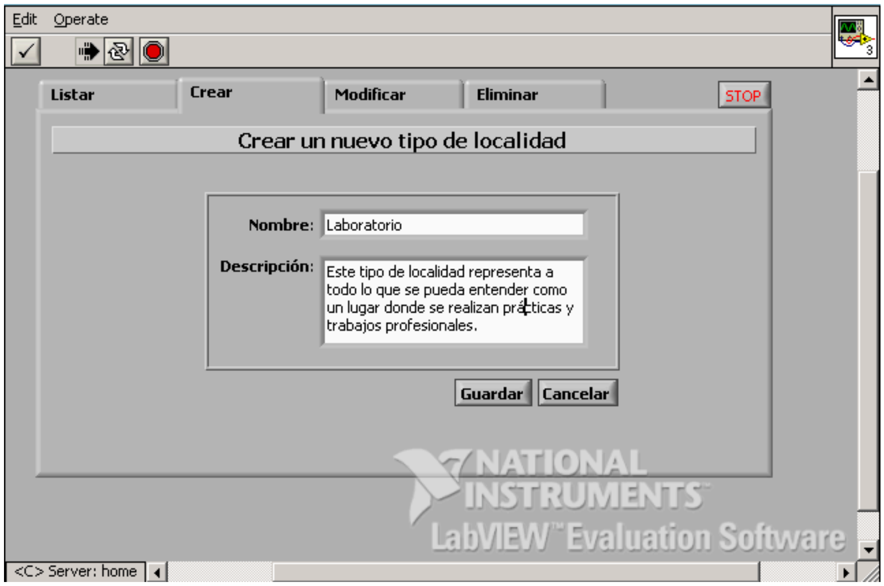
1.1.-Tipo de localidades

1.1.1.- Crear Tipos de Localidades

La secuencia de las operaciones empieza por crear un Tipo de localidad lo cual se hace dirigiéndonos al menú Localidades opción Crear un nuevo tipo de localidad de nuestra página principal. Un tipo de localidad engloba de manera general un conjunto de localidades.



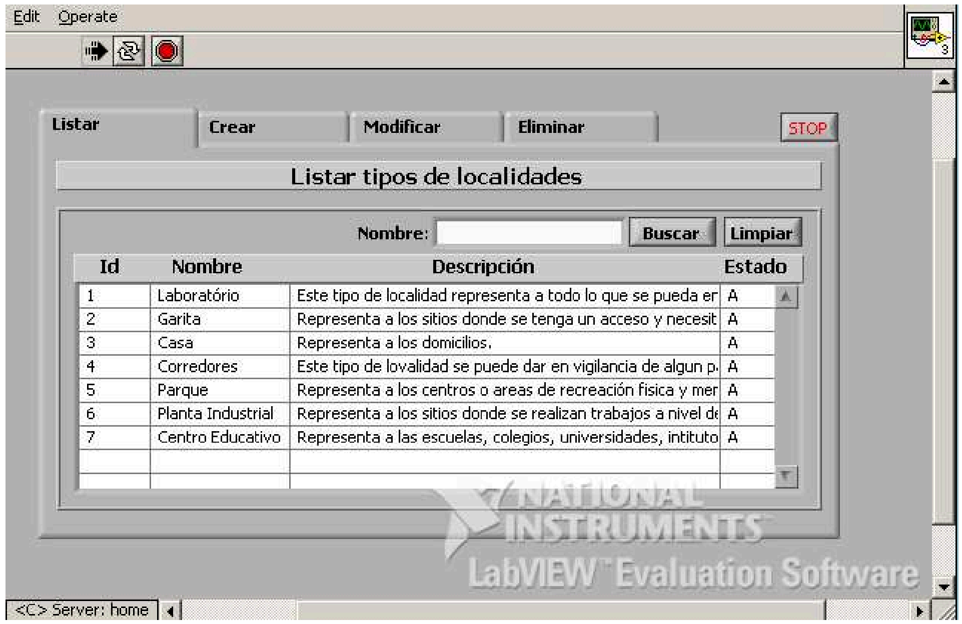
Aquí basta con definir un nombre y una breve descripción del tipo de localidad que deseamos crear, por ejemplo, el nombre de un tipo de localidad puede ser "*Laboratorio*" y su descripción puede ser: "*Este tipo de localidad representa a todo lo que se pueda entender como un lugar donde se realizan prácticas y trabajos profesionales*".



Presionamos el botón Guardar y un mensaje de confirmación nos debe indicar que los datos se guardaron exitosamente. Cabe indicar que al crear una nueva localidad, ésta se guarde con un estado de activo, lo que permite que luego pueda ser consultada, modificada o eliminada usando las opciones que se explican a continuación.

1.1.2.- Listar Tipos de Localidades

Si queremos listar los tipos de localidades que hemos creado debemos irnos a la pestaña de "Listar" la cual se encuentra al lado izquierdo de la pestaña Crear que utilizamos en el ejemplo anterior.

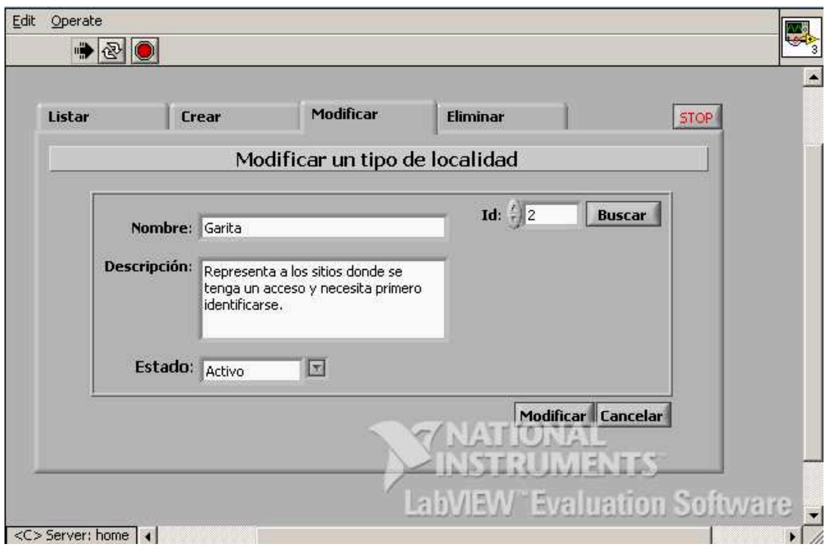


En esta opción podemos filtrar los resultados por el nombre de la localidad, si no colocamos ningún valor en el campo de nombre se listarán todas las localidades que estén almacenadas en la Base de Datos Centralizada mostrando su Id, Nombre, Descripción y Estado. Si queremos listar una localidad específica, damos clic en el botón “Limpiar”, escribimos el nombre de la localidad en el campo nombre y damos clic en el botón “Buscar”

1.1.3.- Modificar Tipos de Localidades

Para modificar un tipo de localidad se debe dar clic en la pestaña “Modificar” luego se debe saber el identificador del tipo de localidad que se desea modificar y lo escribimos en el campo Id.

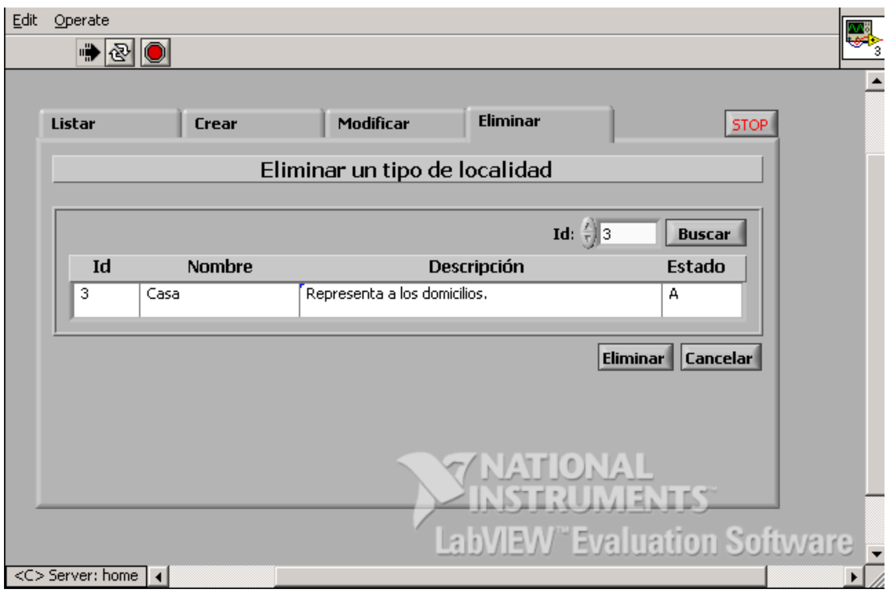
Luego presionamos el botón “Buscar” y automáticamente se cargarán los datos de ese tipo de localidad. Luego de hacer los cambios necesarios, como cambiarle el Nombre, la Descripción y/o el Estado de la Localidad de Activo a Inactivo, presionamos el botón Modificar y si no existe ningún error nos debe salir un mensaje de confirmación indicando que la acción se realizó con éxito. Si no deseamos modificar damos clic en el botón “Cancelar”



1.1.4.- Eliminar Tipos de Localidades

Finalmente supongamos que vamos a eliminar un tipo de localidad, damos clic en la pestaña “Eliminar”, y al igual que para modificar, debemos previamente conocer su identificador, luego presionamos el botón “Buscar” para cargar sus datos y al presionar el botón “Eliminar” se procederá a borrar ese tipo de localidad de la base de datos y nos debe salir un mensaje de confirmación. Si no deseamos eliminar damos clic en el botón “Cancelar”

Cabe recalcar que una vez eliminado el tipo de localidad, las localidades que pertenezcan al tipo eliminado mantendrán ese tipo, pero no se podrán crear localidades con el tipo eliminado.

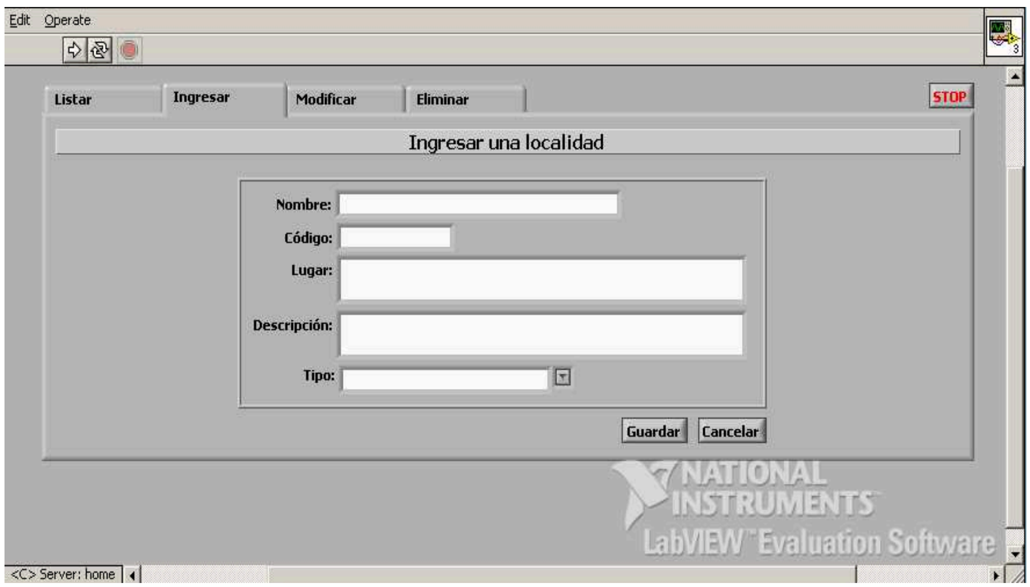


1.2.- Localidades

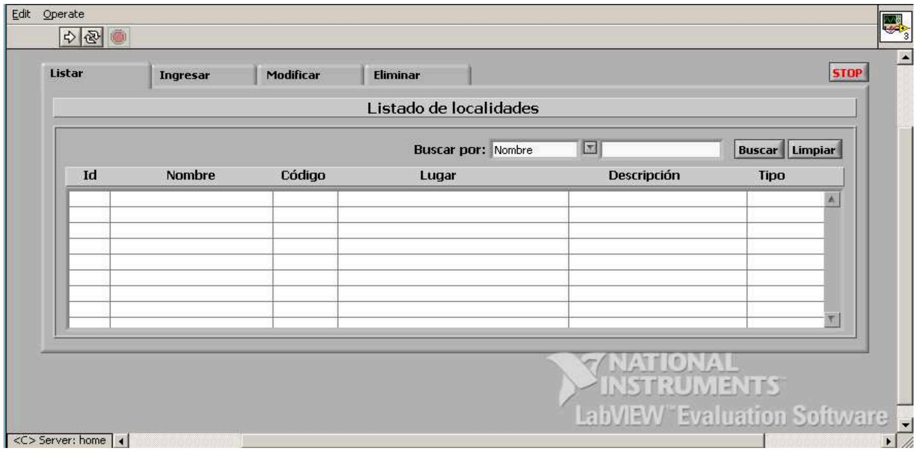
Ahora que ya sabemos cómo crear un Tipo de localidad, el siguiente paso es crear una localidad, para esto en el mismo menú “Localidades” debemos dirigirnos a la pestaña “Ingresar una localidad” donde debemos digitar los datos de la nueva Localidad a crear.

1.2.1. Ingresar una Localidad

Se da clic en la pestaña “Ingresar” luego, se define el nombre de la localidad, el código, el lugar, la descripción y el tipo de localidad previamente creado, luego se presiona el botón “Guardar” y un mensaje de confirmación nos debe indicar que los datos se guardaron exitosamente. Internamente el sistema asigna un identificador para cada localidad. Si no deseamos ingresar una localidad damos clic en el botón “Cancelar”.

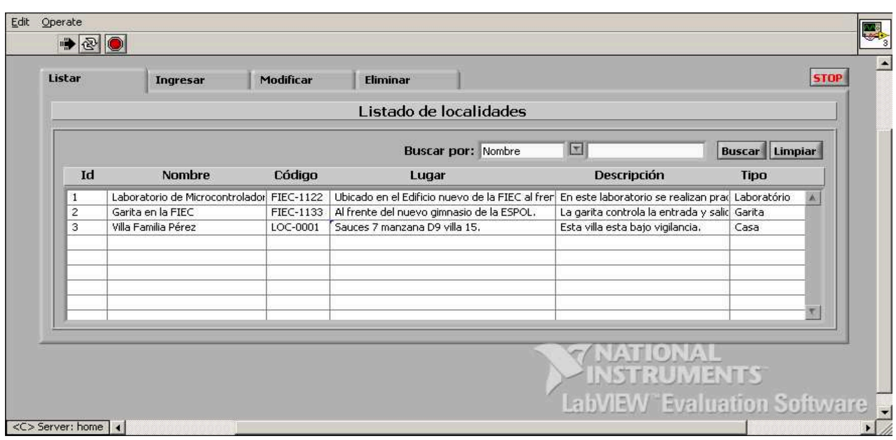


1.2.2. Listar localidades



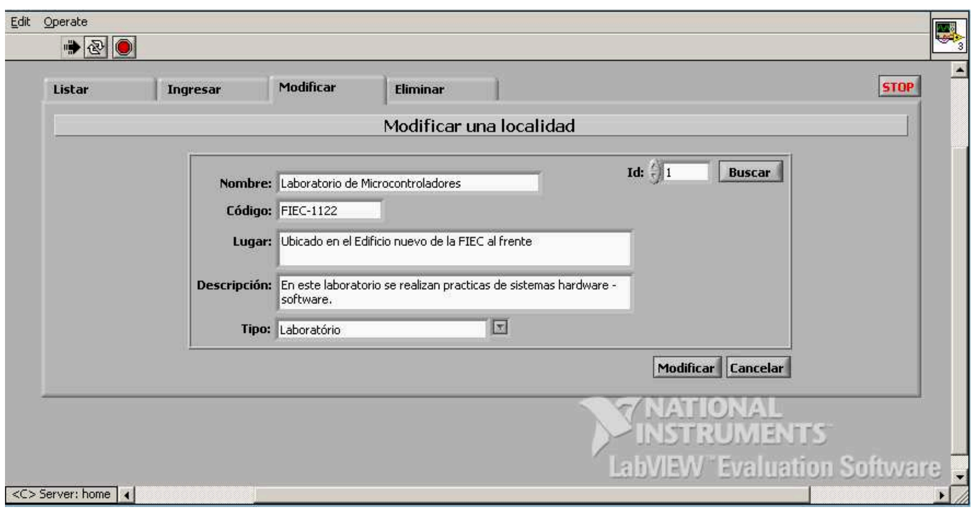
Podemos ver las localidades creadas seleccionando la pestaña "Listar" damos clic en el botón "Buscar" y aparecen todas las localidades creadas con su id, nombre, código, lugar, descripción y el tipo de localidad al que pertenece.

Se puede buscar de acuerdo al nombre, código o tipo, para esto damos clic en el botón "Limpiar", luego escogemos de acuerdo a qué deseamos filtrar y escribimos el filtro.



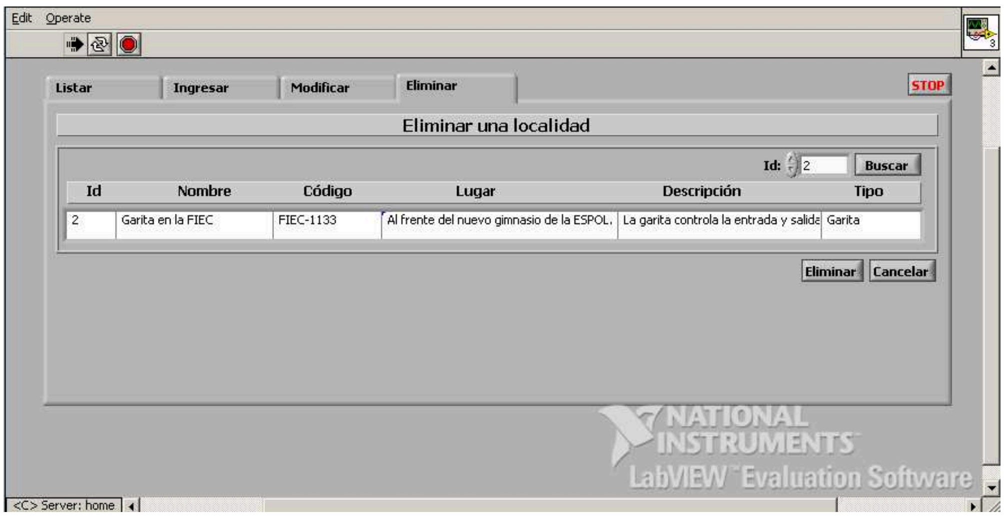
1.2.3. Modificar una Localidad

Una vez que hemos creado una localidad, podemos modificar algún campo de ésta, para esto damos clic en la pestaña “Modificar”, escribimos el identificador de la localidad, damos clic en el botón “Buscar” y automáticamente se cargan los datos de esa localidad. Podemos modificar el nombre, código, lugar, descripción y tipo, una vez modificada damos clic en el botón “Modificar” y aparecerá un mensaje de confirmación de modificación exitosa. Si no deseamos modificar la localidad damos clic en el bot{on “Cancelar”



1.2.4. Eliminar una Localidad

Podemos eliminar una localidad, dando clic en la pestaña “Eliminar” luego escribimos el identificador de la localidad que deseamos eliminar y luego le damos clic en el botón “Buscar”, automáticamente aparecen los datos correspondientes a la localidad y damos clic en el botón “Eliminar”. Se eliminará esa localidad y aparecerá un mensaje de confirmación exitosa de eliminación. Si no deseamos eliminar damos clic en el botón “Cancelar”

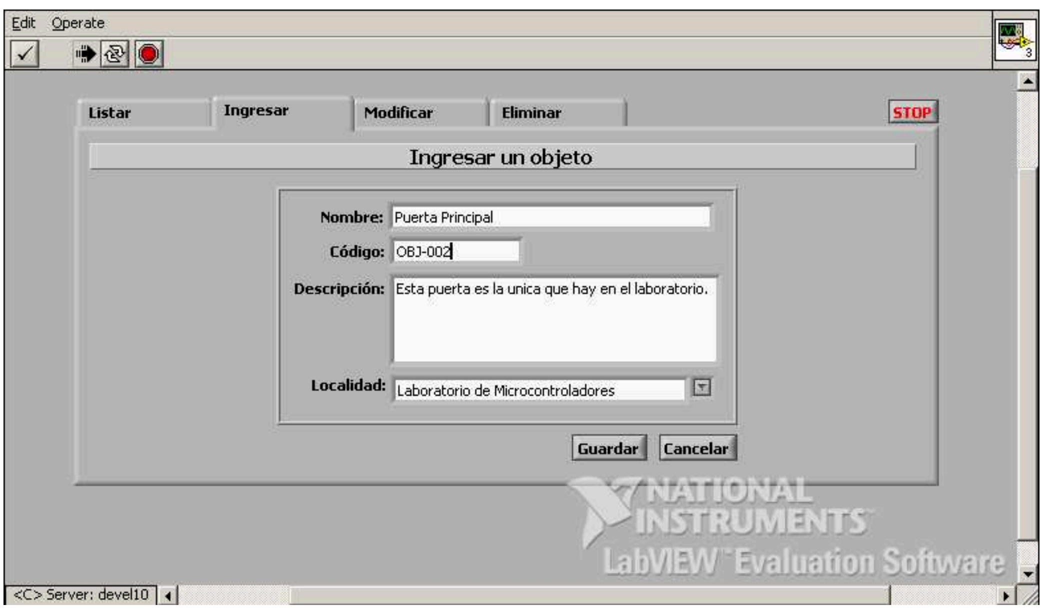


2.- Menú Objetos

Una vez que hemos definido la localidad, debemos crear objetos, los cuales pertenecerán a una localidad.

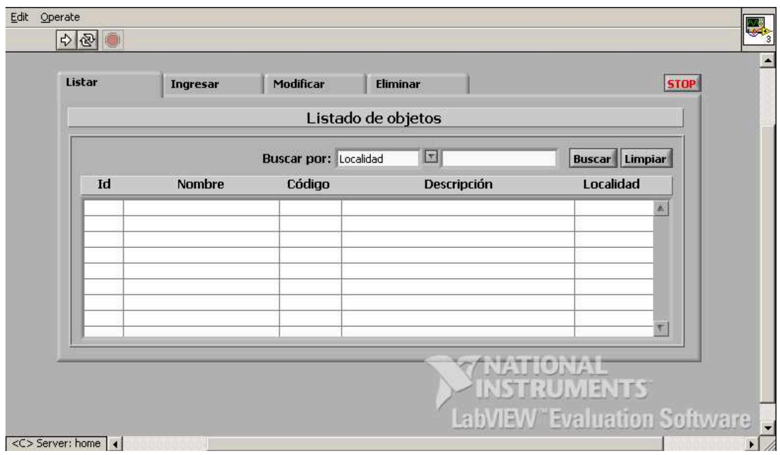
2.1. Ingresar un Objeto

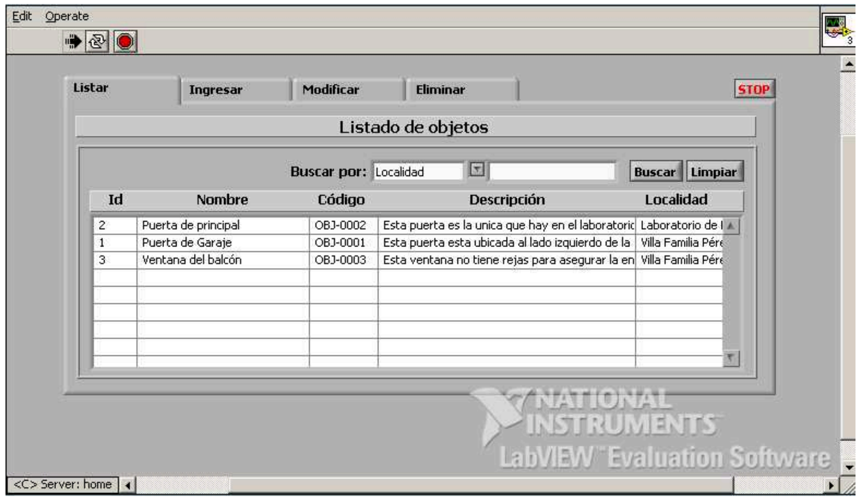
Para crear un objeto, se da clic en la pestaña “Ingresar”, se escribe el nombre del objeto, el código, la descripción y se escoge la localidad a la que se desea que pertenezca. Se da clic en el botón “Guardar” y se presentará un mensaje indicando que la creación del objeto fue exitosa. Internamente el sistema asignará un código para identificar al objeto.



2.2. Listar un objeto

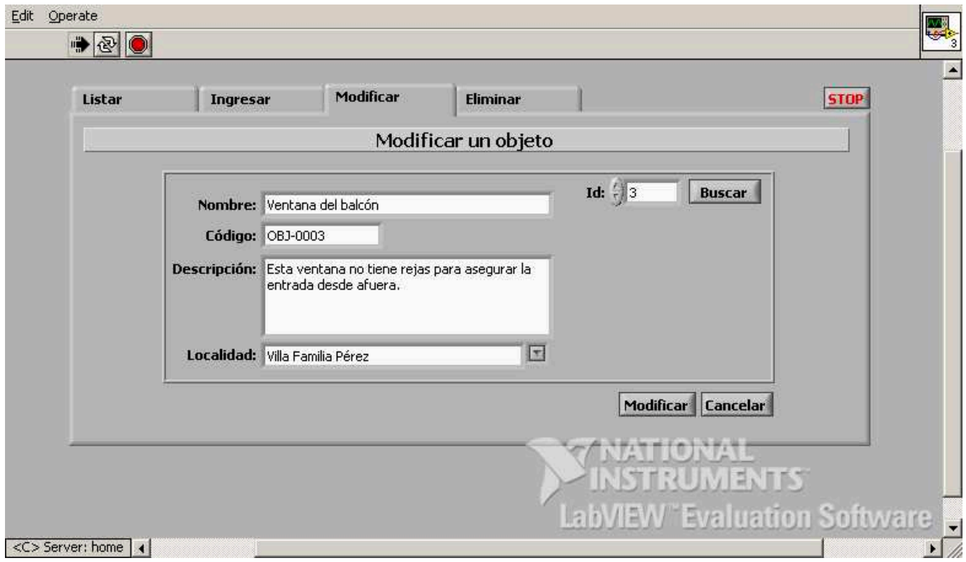
Para ver si se crearon los objetos, damos clic en la pestaña “Listar”, podemos buscar por nombre, código o localidad y damos clic en el botón “Buscar” y aparecerán los objetos que coincidan con el filtro seleccionado,, si no escogemos ningún filtro se mostrarán todos los objetos creados. Se visualiza el id, nombre, código, descripción y localidad del objeto





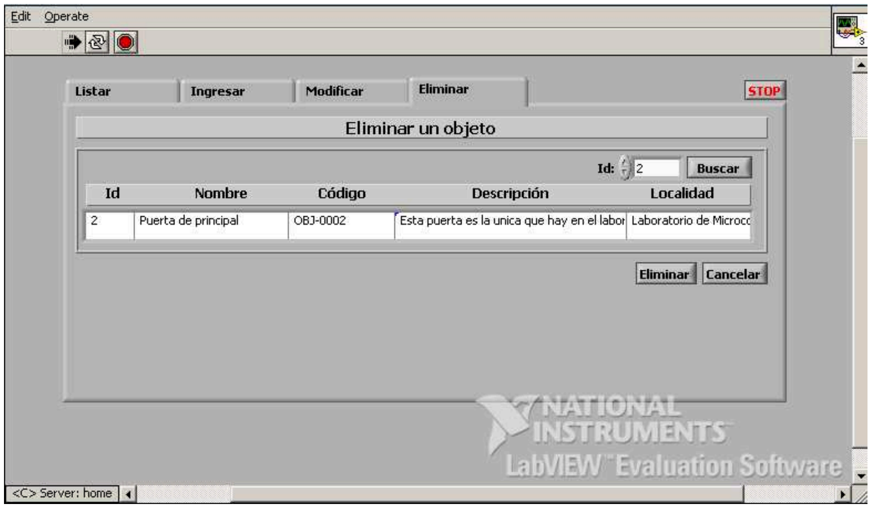
2.3. Modificar un Objeto

Una vez, que hemos creado un objeto, podemos modificarlo, para esto necesitamos dar clic en la pestaña “Modificar”, luego debemos escribir su identificador, damos clic en el botón “Buscar” y automáticamente se cargará el objeto correspondiente a ese identificador. Se puede modificar el nombre, el código, la descripción y/o cambiar la localidad, luego se da clic en el botón “Modificar” y aparece un mensaje de confirmación de que el objeto fue modificado exitosamente. Si no deseamos modificar el objeto damos clic en el botón “Cancelar”.



2.4. Eliminar un Objeto

Para eliminar un objeto, damos clic en la pestaña "Eliminar" escribimos su identificador, luego damos clic en el botón "Buscar" aparecerá el objeto y luego se da clic en el botón "Eliminar". Si no deseamos eliminar el objeto damos clic en el botón "Cancelar".



3. Menú Sensores

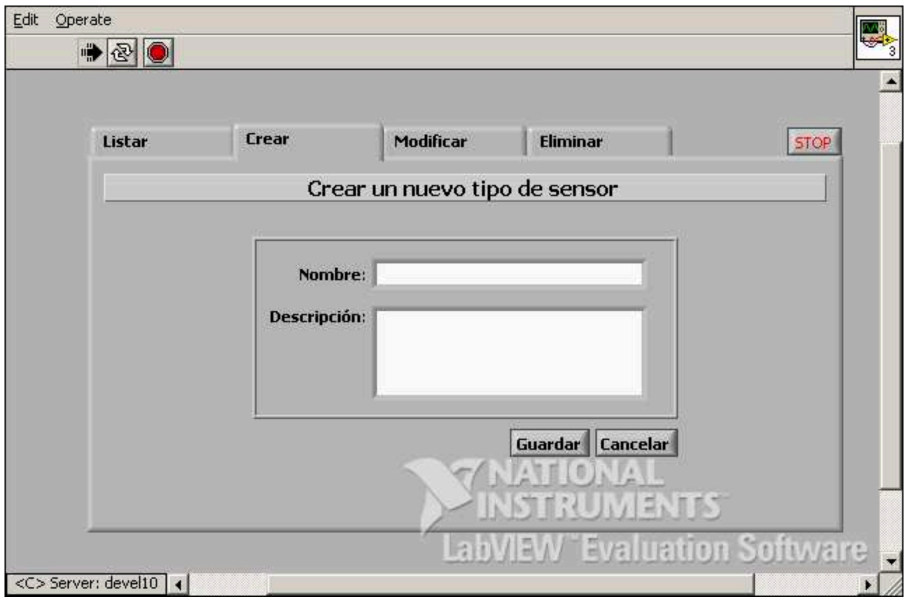
Una vez que hemos definido los objetos en sus localidades, procedemos a colocarles sus sensores, en caso de ser necesarios, para esto, primero creamos los tipos de sensores que pueden tener los objetos.

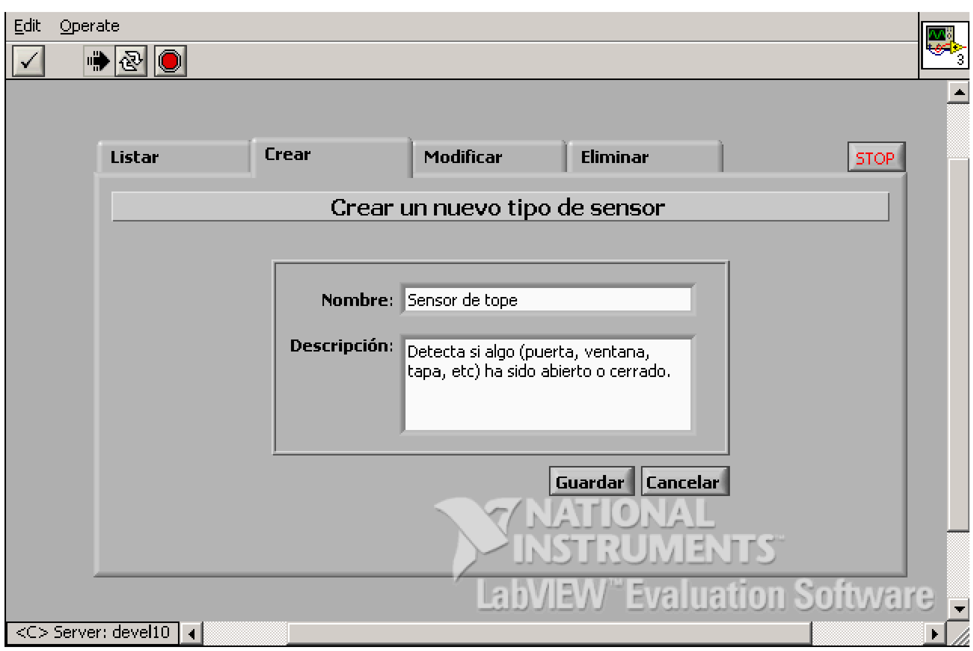
3.1. Tipos de Sensores

3.1.1. Crear un Tipo de Sensor

Dar clic al menú Sensores, desplazar hacia abajo el scrollbar vertical y escoger la pestaña “Crear” de la ventana que está en la parte inferior.

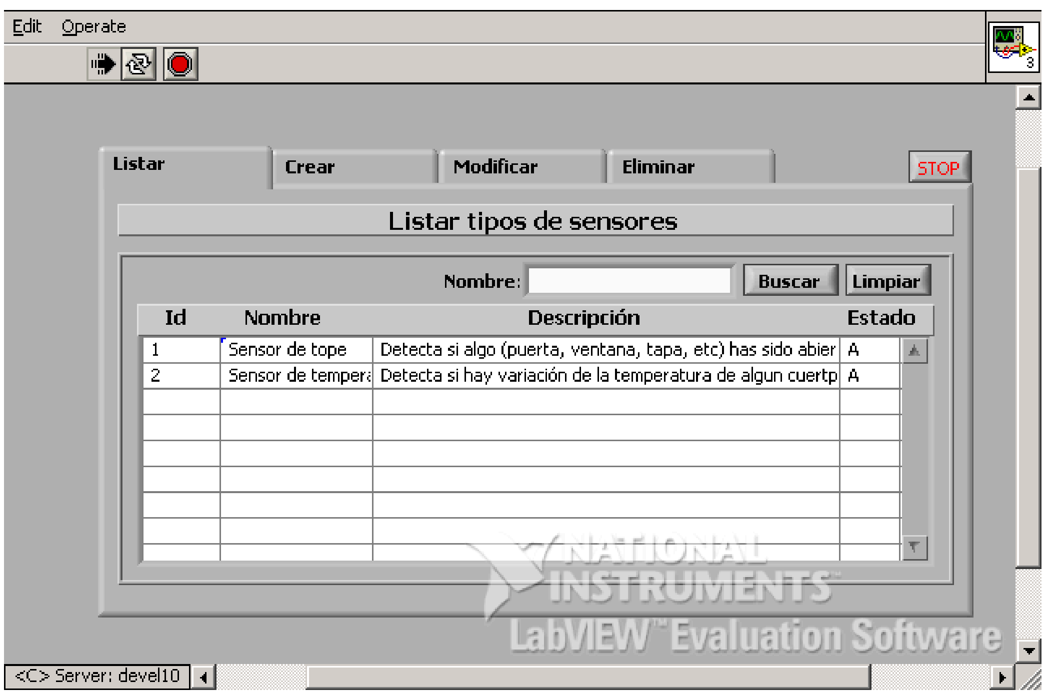
Escribir el nombre y la descripción del nuevo tipo de sensor que se desea crear, dar clic en el botón “Guardar”, aparecerá un mensaje de confirmación de creación del nuevo tipo de sensor. Internamente, cada vez que se crea un tipo de sensor, el sistema le asignará el estado de “activo”. Si no se desea crear un nuevo tipo de sensor damos clic en el botón “Cancelar”.





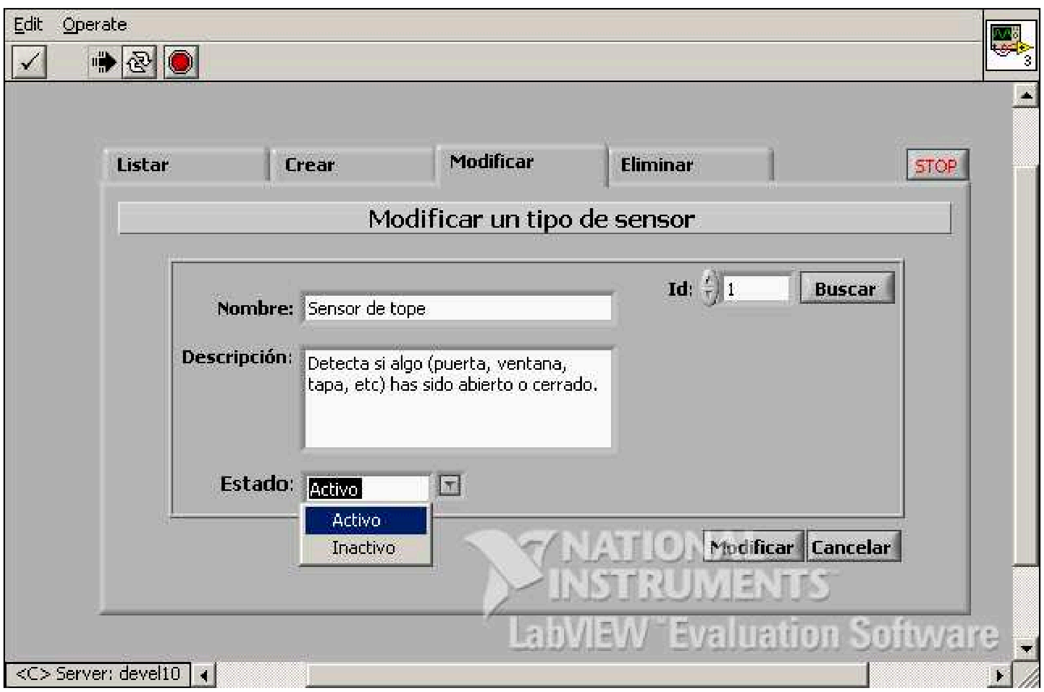
3.1.2. Listar Tipos de Sensores

Para visualizar los tipos de sensores que hemos creado, dar clic en la pestaña "Listar" y presionar el botón "Buscar" automáticamente el sistema cargará todos los tipos de sensores creados, si queremos que aparezca alguno en especial, escribir el nombre del sensor en el campo de texto "Nombre" y dar clic en el botón "Buscar". Para actualizar los datos, dar clic en el botón "Limpiar" y luego en el botón "Buscar", se borrarán los tipos de sensores y se cargarán nuevamente.



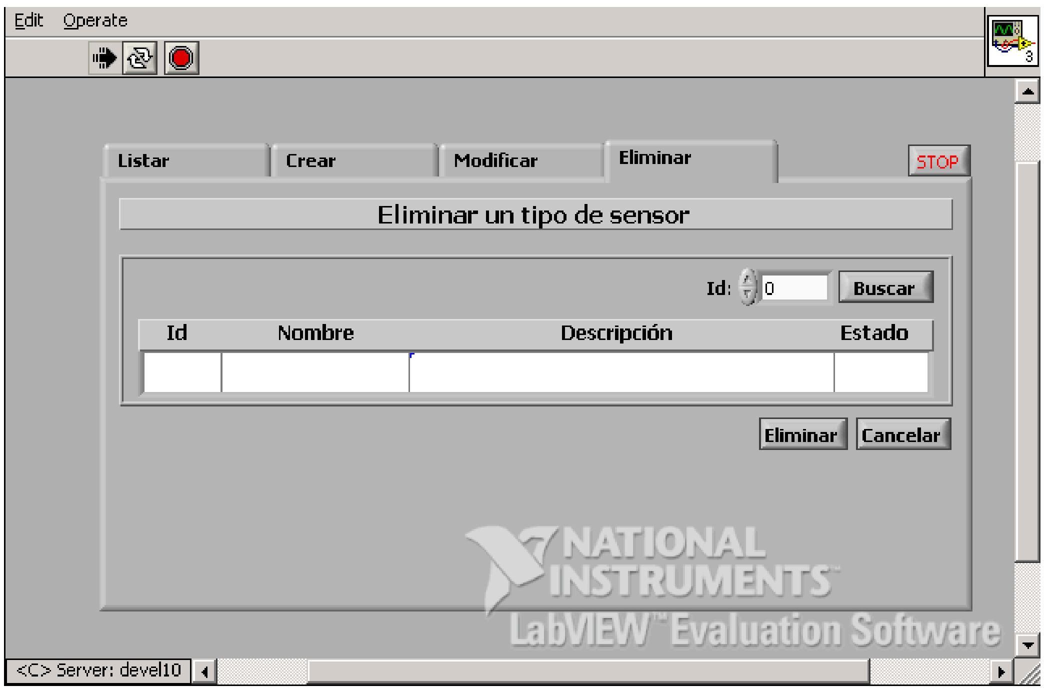
3.1.3. Modificar un Tipo de Sensor

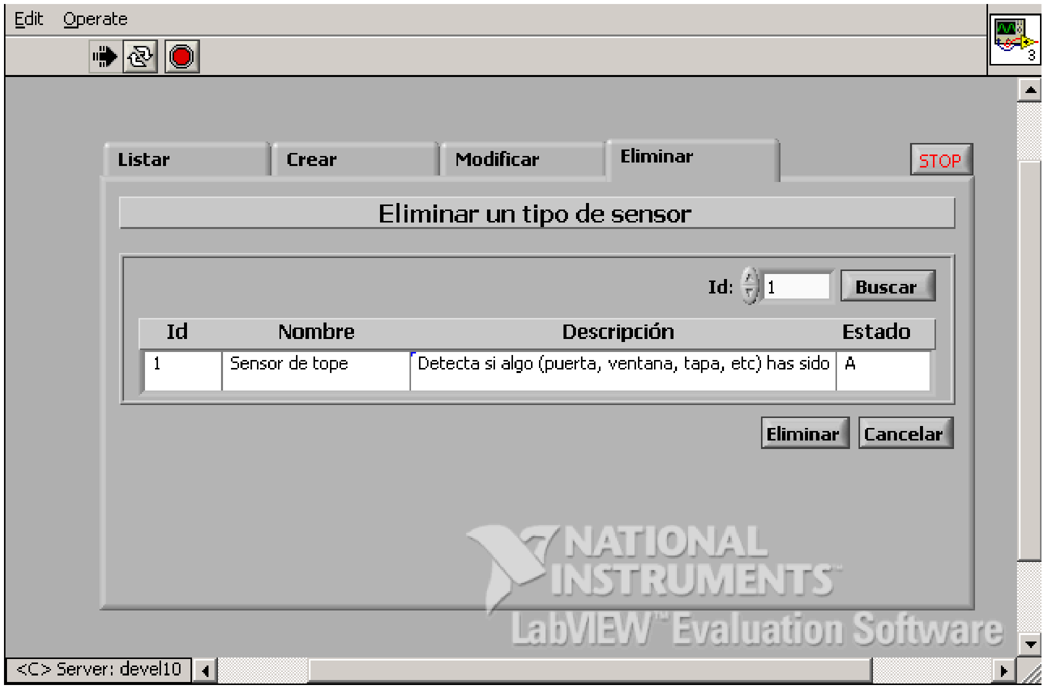
Podemos modificar los tipos de sensores, para esto damos clic en la pestaña "Modificar", escribimos el identificador (id) del sensor a modificar y damos clic en el botón "Buscar", automáticamente se cargará el nombre, la descripción y el estado del tipo del sensor, podemos cambiar cualquiera de estos atributos, luego damos clic en el botón "Modificar" y nos aparecerá un mensaje de confirmación de modificación exitosa del tipo de sensor. Si no deseamos modificar damos clic en el botón "Cancelar"



3.1.4. Eliminar un Tipo de Sensor

Podemos eliminar un tipo de sensor, escogiendo la pestaña “Eliminar” y luego escribiendo el identificador (id) del tipo de sensor y presionando el botón “Buscar”, aparecerá los campos correspondientes al tipo de sensor que se desea eliminar y presionamos el botón “Eliminar”. Si no deseamos eliminar presionamos el botón “Cancelar” Si eliminamos el tipo de sensor, posteriormente, los sensores que pertenezcan a ese tipo de sensor conservará su tipo, pero no se podrán crear sensores del tipo eliminado.



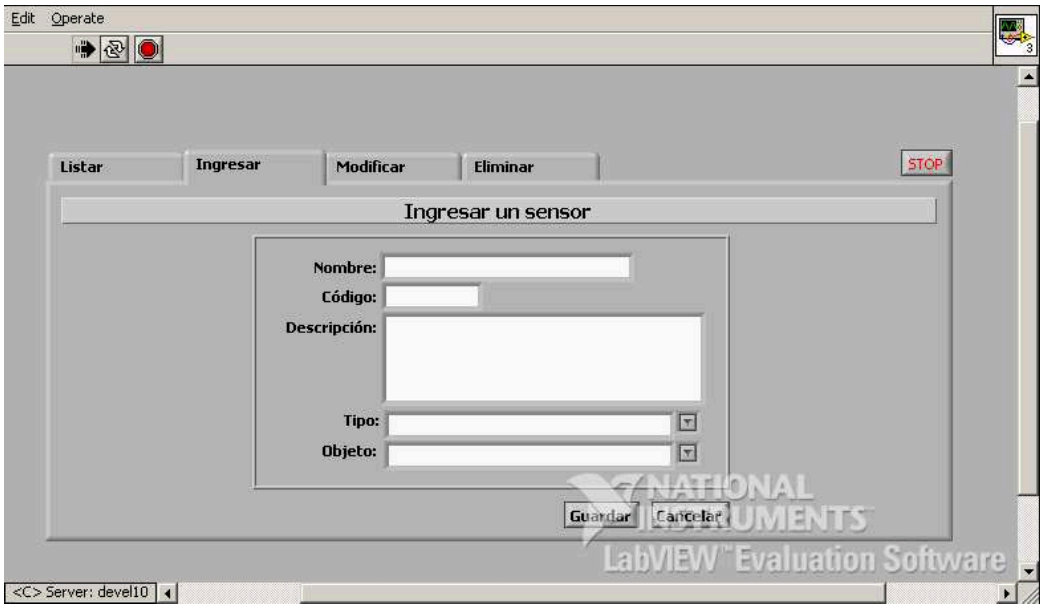


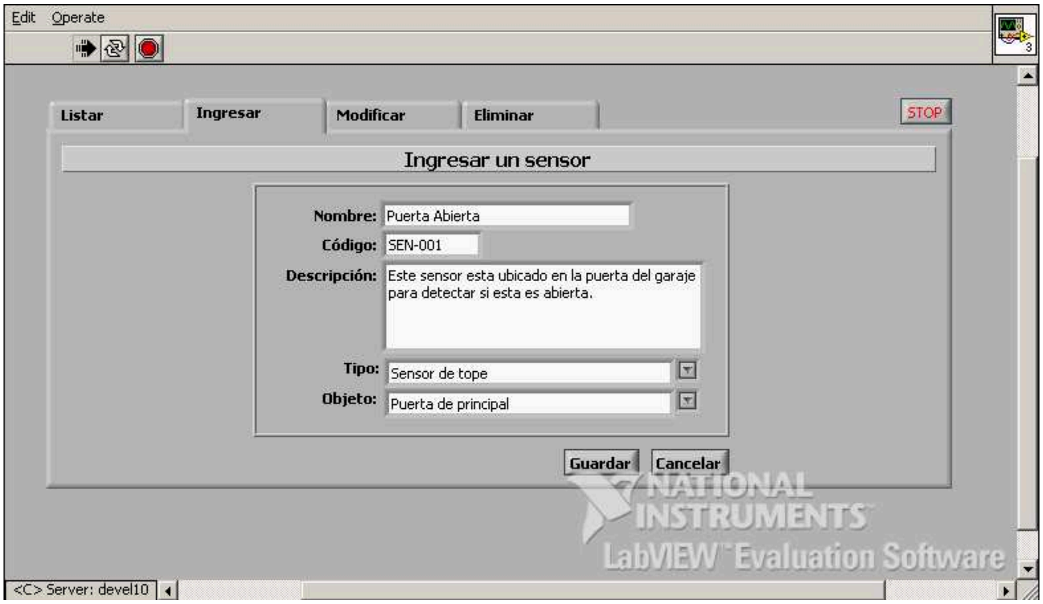
3.2. Sensor

Una vez que hemos definido los tipos de sensores, procedemos a ingresar los sensores pertenecientes a cada tipo de sensor.

3.2.1. Ingresar Sensores

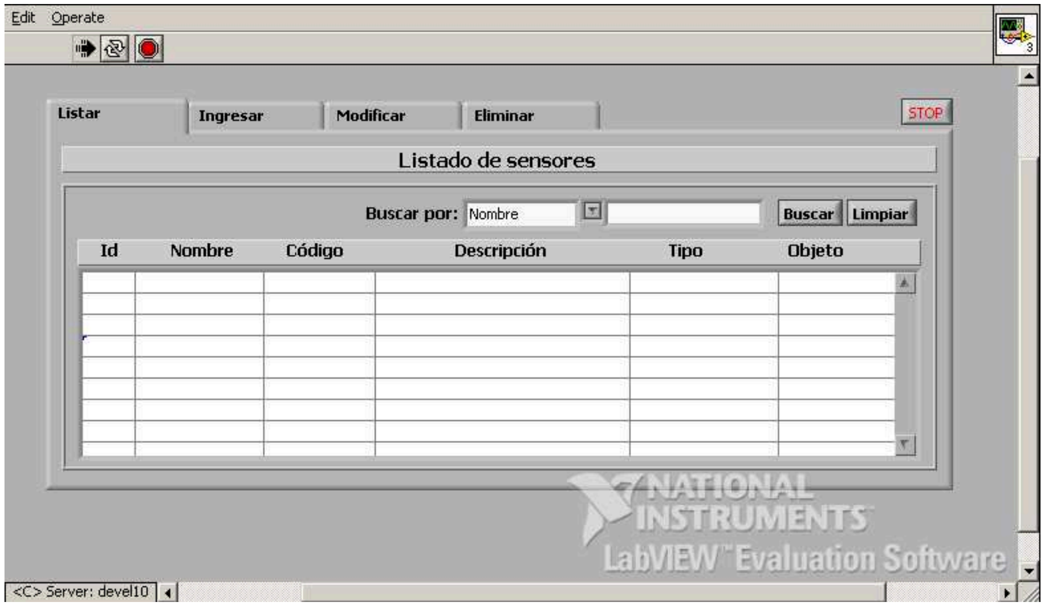
Dentro del menú Sensores, dar clic a la pestaña “Ingresar”, procedemos a llenar los campos correspondientes como nombre, código, descripción, escogemos el tipo de sensor y el objeto previamente creado y damos clic en el botón “Guardar”. Aparecerá un mensaje de confirmación exitosa de creación del sensor. Si no deseamos crear un sensor damos clci en el botón "Cancelar".

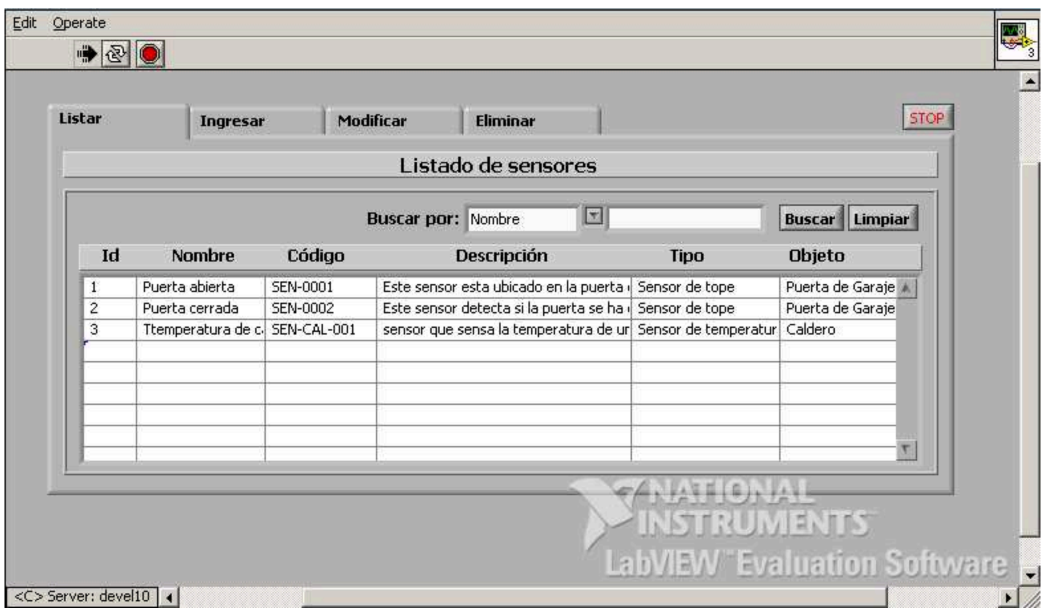




3.2.2. Listar Sensores

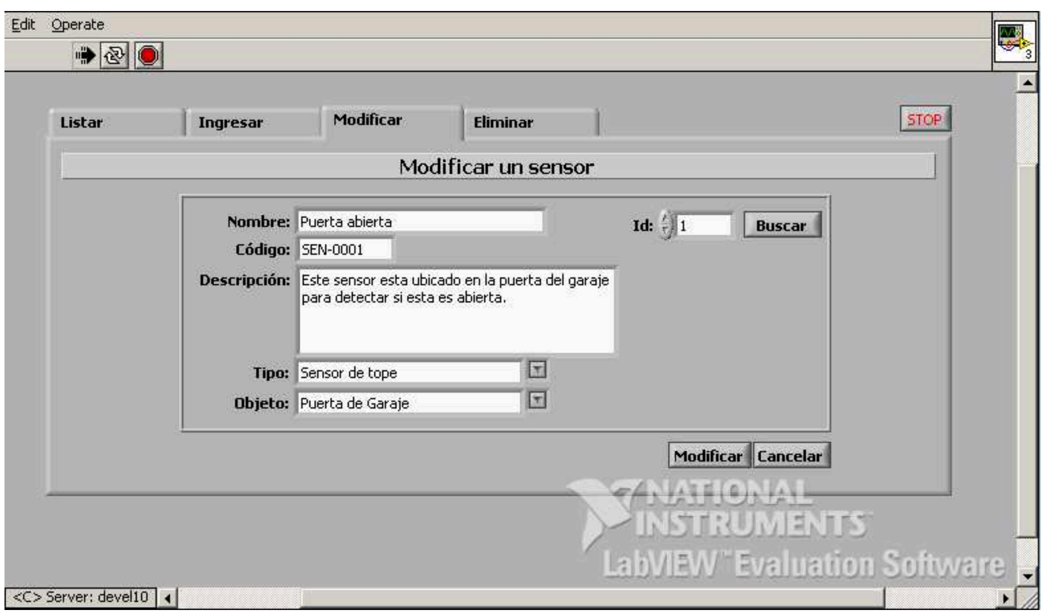
Para tener un reporte de todos los sensores que hemos creado, damos clic a la pestaña “Listar” y damos clic al botón “Buscar” automáticamente se cargará todos los sensores creados, con sus campos correspondientes como id, nombre, código, descripción, tipo y objeto. Podemos filtrar esta búsqueda por nombre, código, tipo y objeto, para esto escogemos el filtro por el cual se desea buscar, escribimos en el campo de texto vacío el dato para filtrar y presionamos el botón “Buscar” Para realizar una nueva búsqueda presionamos el botón “Limpiar” el cual borrará el listado de sensores y el filtro por el cual se realizó la búsqueda.


Edit Operate
Listar
Ingresar
Modificar
Eliminar
STOP
Listado de sensores
Buscar por:
Nombre
Buscar
Limpiar
Id
Nombre
Código
Descripción
Tipo
Objeto
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
<C> Server: devel10


Edit Operate
Listar
Ingresar
Modificar
Eliminar
STOP
Listado de sensores
Buscar por:
Nombre
Buscar
Limpiar
Id
Nombre
Código
Descripción
Tipo
Objeto
1
Puerta abierta
SEN-0001
Este sensor esta ubicado en la puerta
Sensor de tope
Puerta de Garaje
2
Puerta cerrada
SEN-0002
Este sensor detecta si la puerta se ha
Sensor de tope
Puerta de Garaje
3
Ttemperatura de c
SEN-CAL-001
sensor que sensa la temperatura de ur
Sensor de temperatur
Caldero
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
<C> Server: devel10

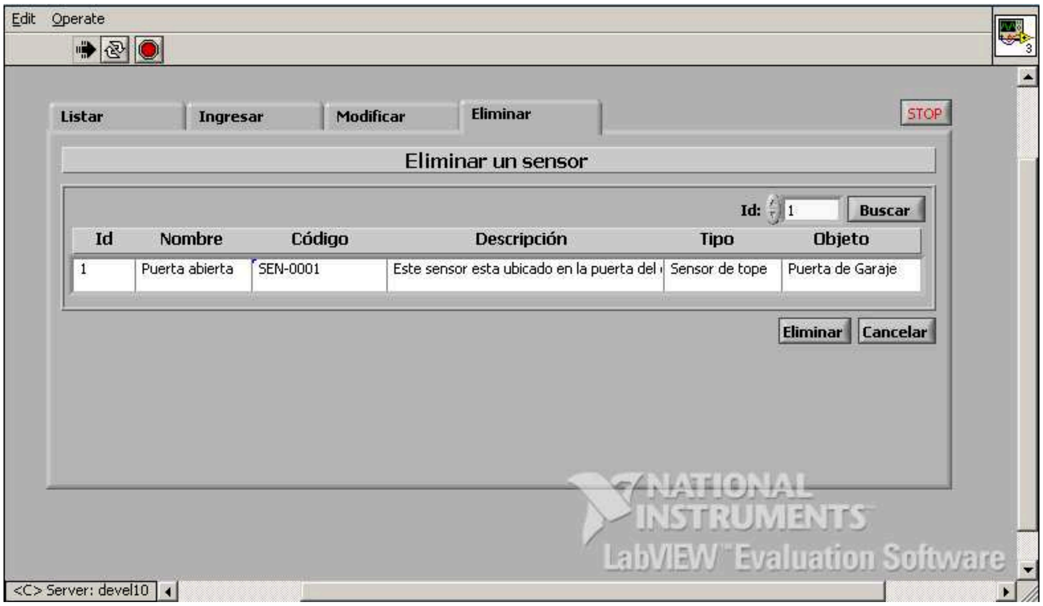
3.2.3. Modificar Sensores

Podemos modificar un sensor, para esto, damos clic en la pestaña “Modificar” y escribimos el identificador (id) del sensor que deseamos modificar y damos clic en el botón “Buscar” Automáticamente se cargará la información correspondiente al sensor, podemos modificar los campos como nombre, código, descripción y/o cambiar el tipo de sensor y el objeto al cual se le asignó el sensor, luego damos clic en el botón “Modificar”. Aparecerá un mensaje de confirmación exitosa de modificación del sensor. Si no deseamos modificar el sensor damos clic en el botón “Cancelar”.



3.2.4. Eliminar Sensores

Para eliminar un sensor, damos clic en la pestaña “Eliminar” y escribimos el identificador (id) del sensor a eliminar, damos clic en el botón “Buscar” Automáticamente se cargará los datos del sensor y luego damos clic en el botón “Eliminar” aparecerá un mensaje de confirmación exitosa de eliminación del sensor. Si no deseamos eliminar damos clic en el botón “Cancelar”.


Edit
Operate
Listar
Ingresar
Modificar
Eliminar
STOP
Eliminar un sensor
Id:
1
Buscar
Id
Nombre
Código
Descripción
Tipo
Objeto
1
Puerta abierta
SEN-0001
Este sensor esta ubicado en la puerta del
Sensor de tope
Puerta de Garaje
Eliminar
Cancelar
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
<C> Server: devel10

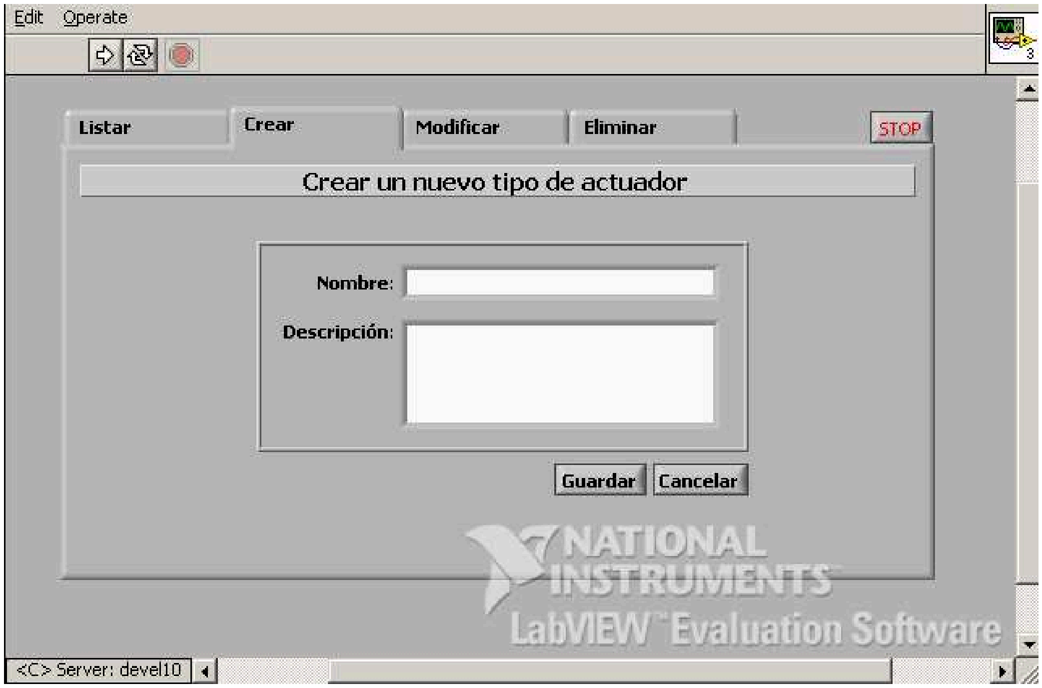
4. Menú Actuadores

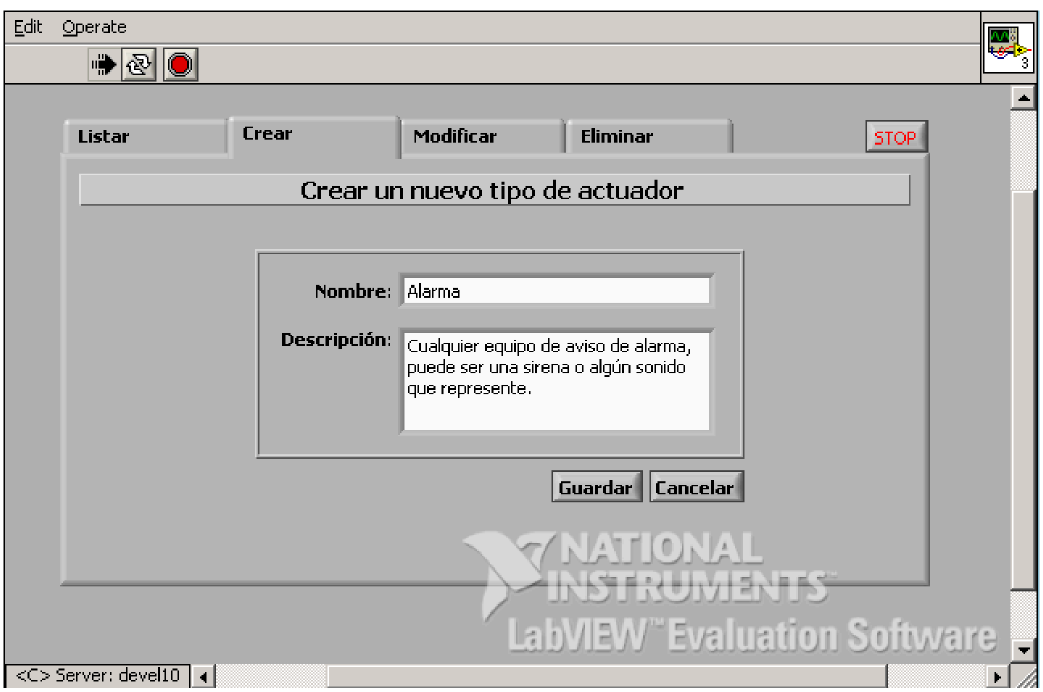
Una vez que hemos definido los objetos en sus localidades, procedemos a colocarles sus actuadores, en caso de ser necesarios, para esto, primero creamos los tipos de actuadores que pueden tener los objetos.

4.1. Tipo de Actuador

4.1.1. Crear un Tipo de Actuador

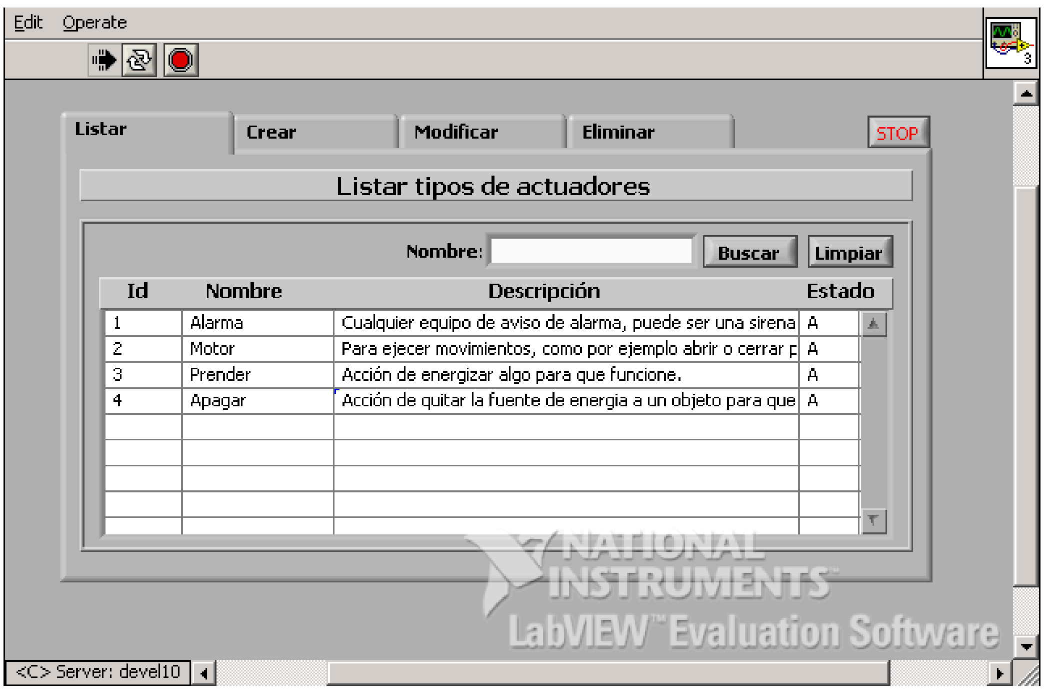
Para crear un nuevo tipo de actuador, damos clic en el menú “Actuador” y desplazamos verticalmente hacia abajo el scrollbar y damos clic en la pestaña “Crear” de la ventana inferior. Escribimos el nombre y la descripción del tipo de actuador y damos clic en el botón “Guardar”. Si no deseamos crear un nuevo tipo de actuador damos clic en el botón “Cancelar”. Internamente, el sistema asigna un identificador al tipo de actuador y se asigna un estado activo.


Edit Operate
Listar
Crear
Modificar
Eliminar
STOP
Crear un nuevo tipo de actuador
Nombre:
Descripción:
Guardar
Cancelar
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
<C> Server: devel10


Edit Operate
Listar
Crear
Modificar
Eliminar
STOP
Crear un nuevo tipo de actuador
Nombre:
Alarma
Descripción:
Cualquier equipo de aviso de alarma, puede ser una sirena o algún sonido que represente.
Guardar
Cancelar
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
<C> Server: devel10

4.1.2. Listar Tipo de Actuador

Podemos visualizar todos los tipos de actuadores dando clic en el botón "Buscar ", si deseamos ver algún tipo de actuador en especial, se da clic en el botón "Limpiar", se escribe el nombre del tipo de actuador y se da clic en el botón "Buscar". Se visualiza el id, nombre, descripción y estado.



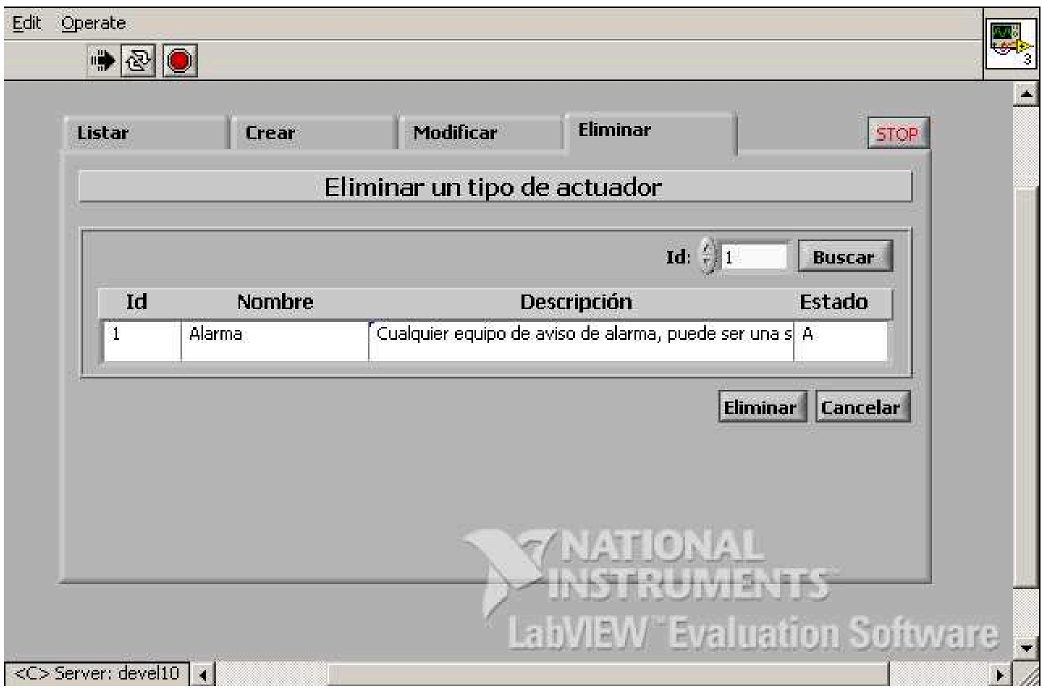
4.1.3. Modificar un Tipo de Actuador

Un tipo de actuador puede ser modificado,, para esto damos clic en la pestaña "Modificar", escribimos en el campo Id el identificador del tipo de actuador y damos clic en el botón "Buscar", se cargarán automáticamente la información correspondiente al tipo de actuador como son: Nombre, descripción y el estado, cambiamos lo que deseamos modificar y damos clic en el botón “Modificar”. Si no deseamos modificar el tipo de actuador damos clic en el botón “Cancelar”

4.1.4. Eliminar un Tipo de Actuador

Un tipo de actuador puede ser eliminado,, para esto damos clic en la pestaña “Eliminar”, escribimos en el campo Id el identificador del tipo de actuador y damos clic en el botón “Buscar”, se cargarán automáticamente la información correspondiente al tipo de actuador como son: Nombre, descripción y el estado, para eliminar el tipo de actuador damos clic en el botón “Eliminar”. Si no deseamos eliminar el tipo de actuador damos clic en el botón “Cancelar”

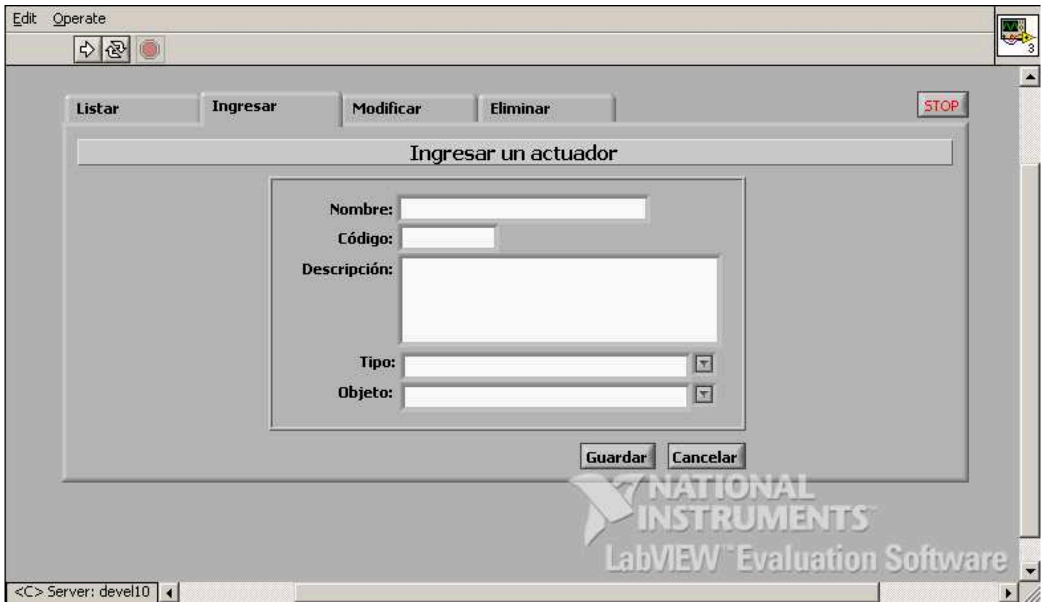
Una vez que eliminamos el tipo de actuador , los actuadores que fueron creados perteneciendo al tipo eliminado, mantendrán ese tipo, pero no se podrán crear nuevos actuadores con el tipo eliminado.

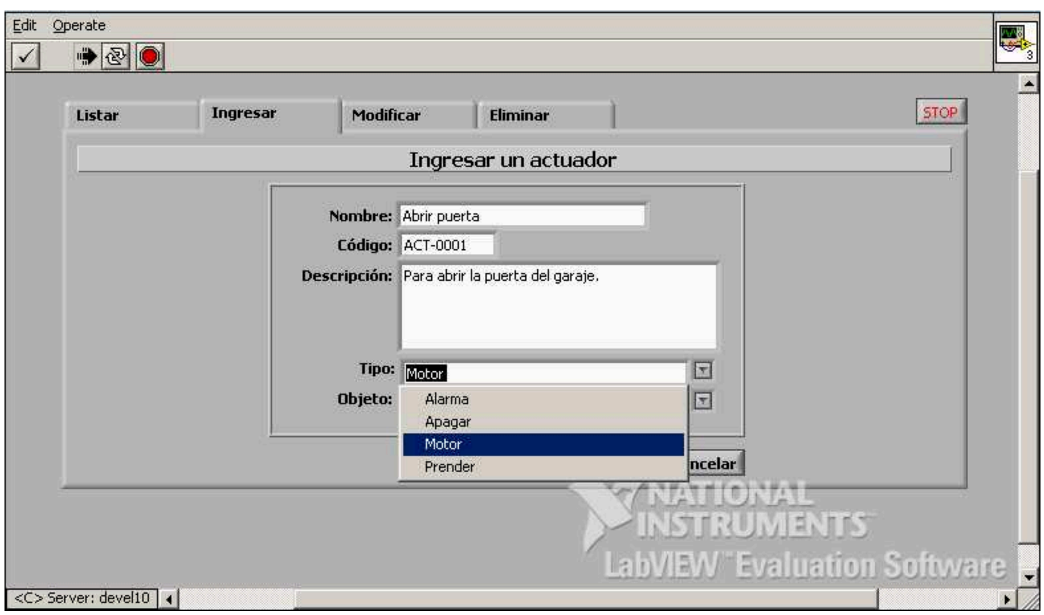


4.2. Actuadores

4.2.1. Ingresar un Actuador

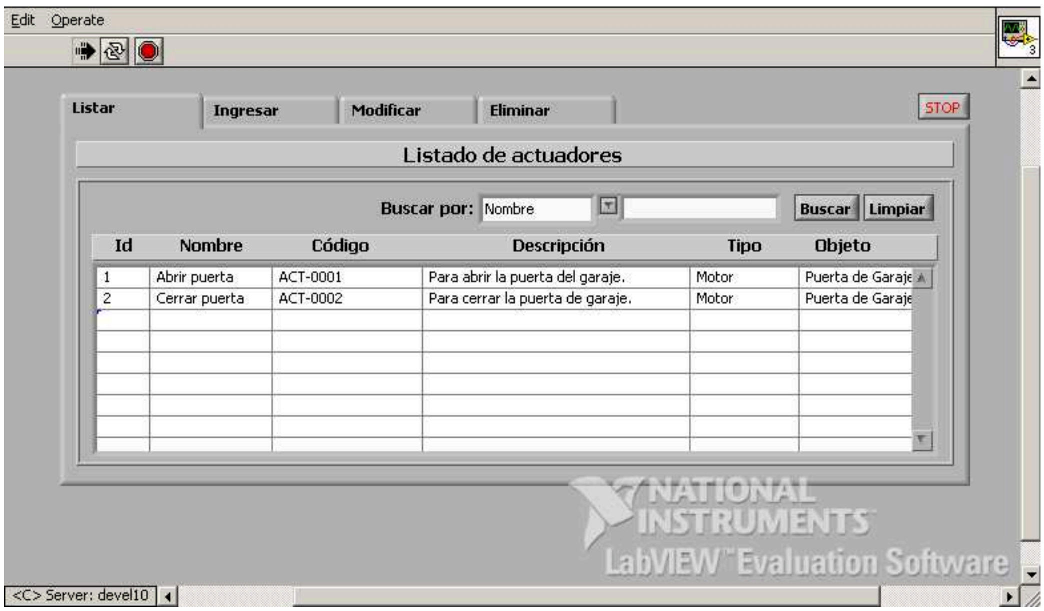
Para ingresar un actuador damos clic en la ventana “Ingresar” de la ventana superior del menú “Actuador”, llenamos los campos: Nombre, código, descripción, escogemos el tipo de actuador y el objeto, luego damos clic en el botón “Guardar” y se graba en la base de datos. Internamente, el sistema le asigna un identificador y le asigna un estado de activo al actuador.





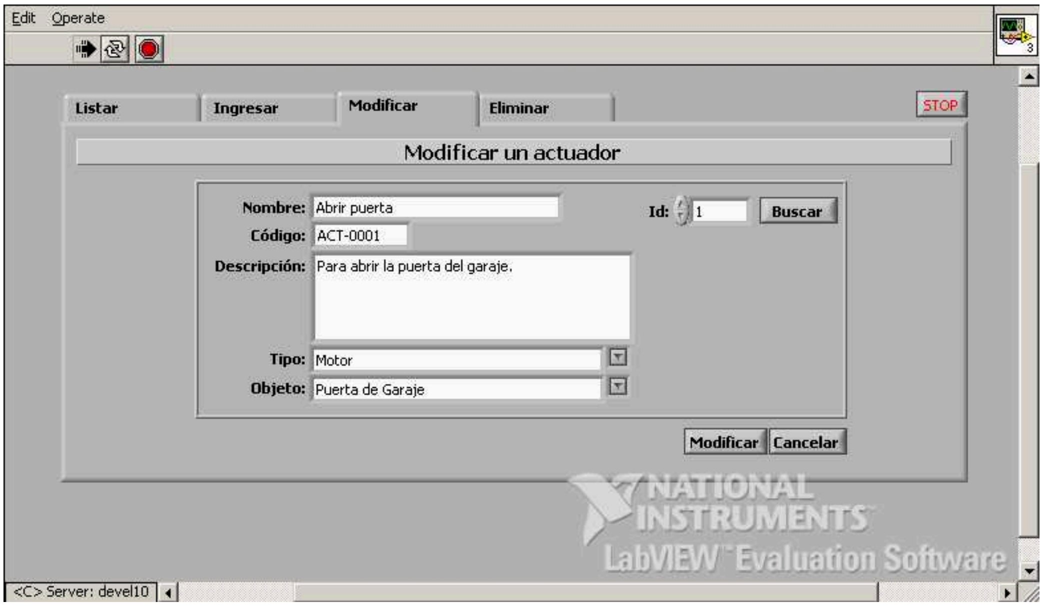
4.2.2. Listar un Actuador

Podemos ver todos los actuadores que han sido creados, damos clic en el botón “Buscar” si deseamos filtrar podemos buscar por Nombre, código, tipo y objeto.



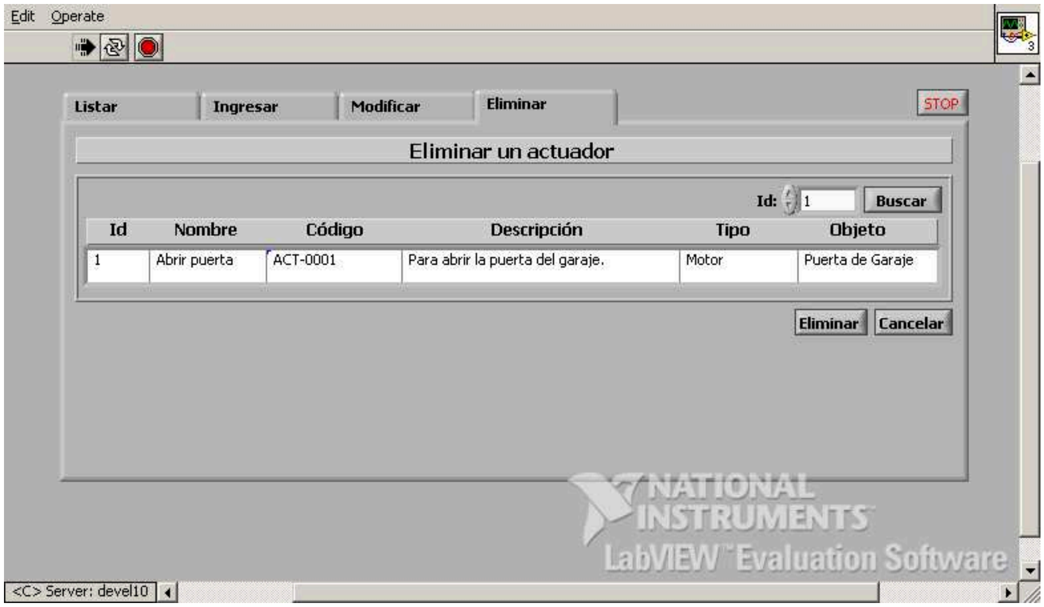
4.2.3. Modificar un Actuador

Si deseamos modificar un actuador, damos clic en la pestaña “Modificar”, escribimos el identificador del actuador que se desea modificar y se da clic en el botón “Buscar”. Automáticamente se carga la información correspondiente al actuador, y se puede cambiar el nombre, código, descripción, tipo de actuador y objeto, damos clic en el botón “Modificar”, aparecerá un mensaje de confirmación exitosa de modificación. Si no deseamos modificar damos clic en el botón “Cancelar”



4.2.4. Eliminar un Actuador

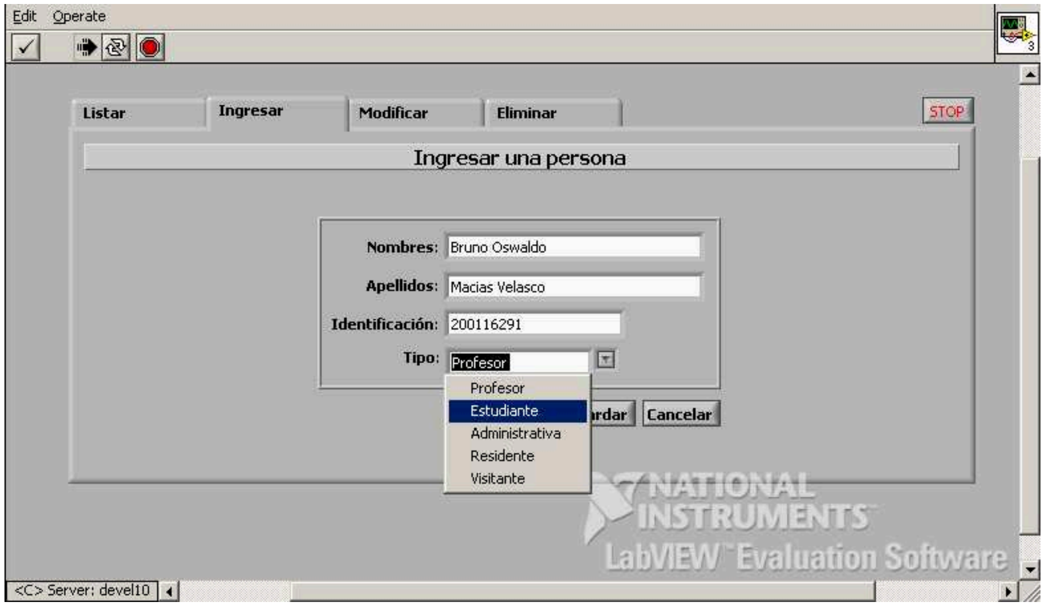
Podemos eliminar un actuador dando clic en la pestaña “Eliminar”, escribimos el identificador del actuador y damos clic en el botón “Buscar, se visualiza la información del actuador como id, nombre, código, descripción, tipo y objeto y damos clic en el botón “Eliminar” Si no deseamos eliminar damos clic en el botón “Cancelar”



5. Menú Personas

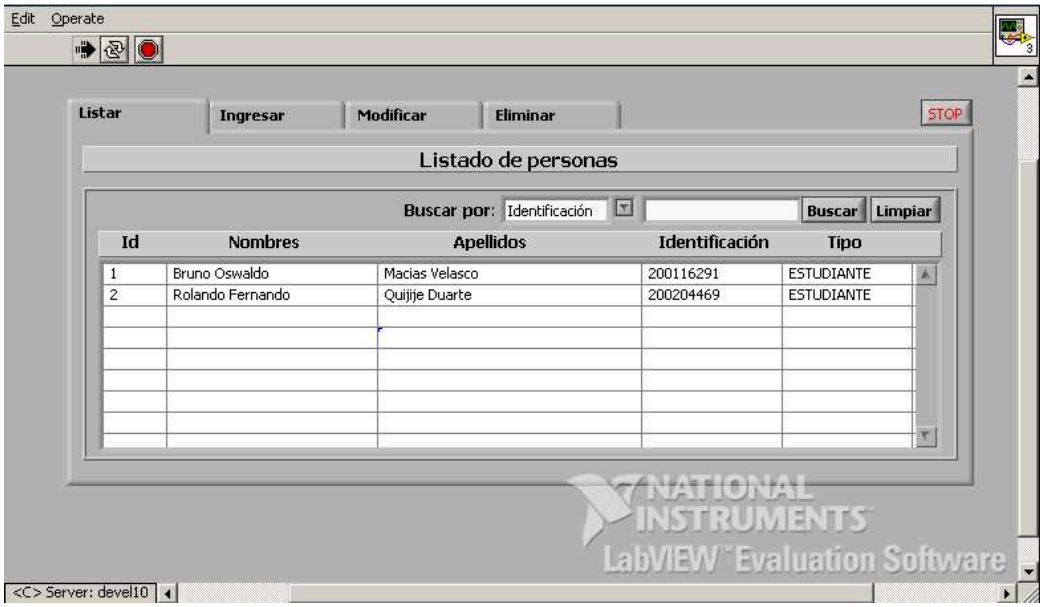
5.1. Ingresar una Persona

Para ingresar una persona, damos clic en el menú "Persona", escogemos la pestaña "Ingresar" y escribimos los nombres, apellidos, identificación y tipo de persona que puede ser Profesor, Estudiante, Administrativa, Residente, y Visitante y damos clic en el botón "Guardar", si no deseamos ingresar una persona damos clic en el botón "Cancelar"



5.2. Listar Personas

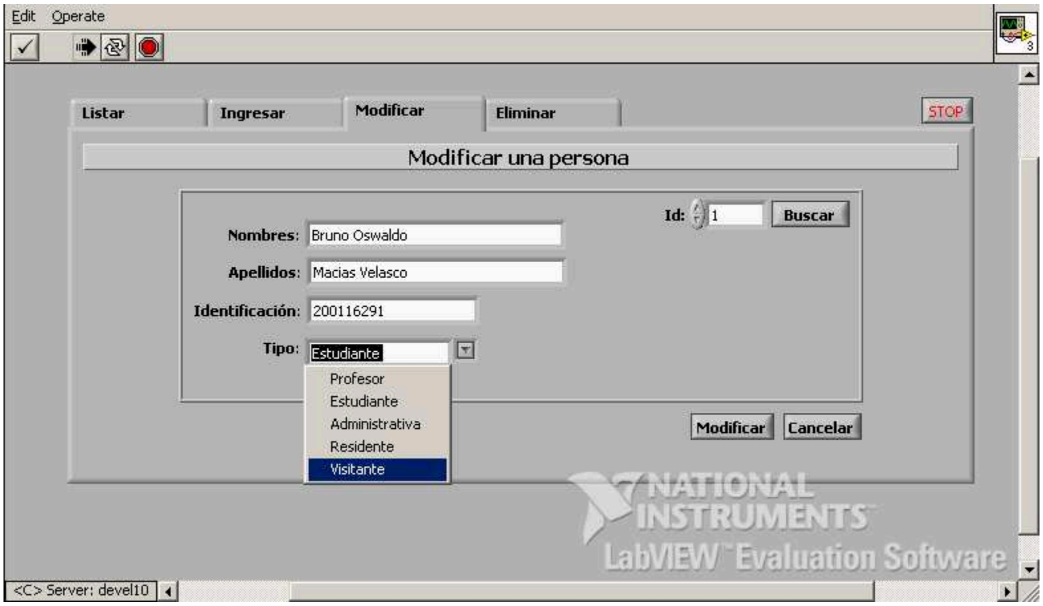
Podemos visualizar todas las personas dando clic en el botón “Buscar” de la pestaña “Listar”, podemos realizar la búsqueda por identificación, nombre, apellido y tipo para filtrar el listado.



5.3. Modificar una Persona

Podemos modificar una persona dando clic en la pestaña “Modificar” y escribimos el identificador de la persona y presionamos el botón “Buscar”

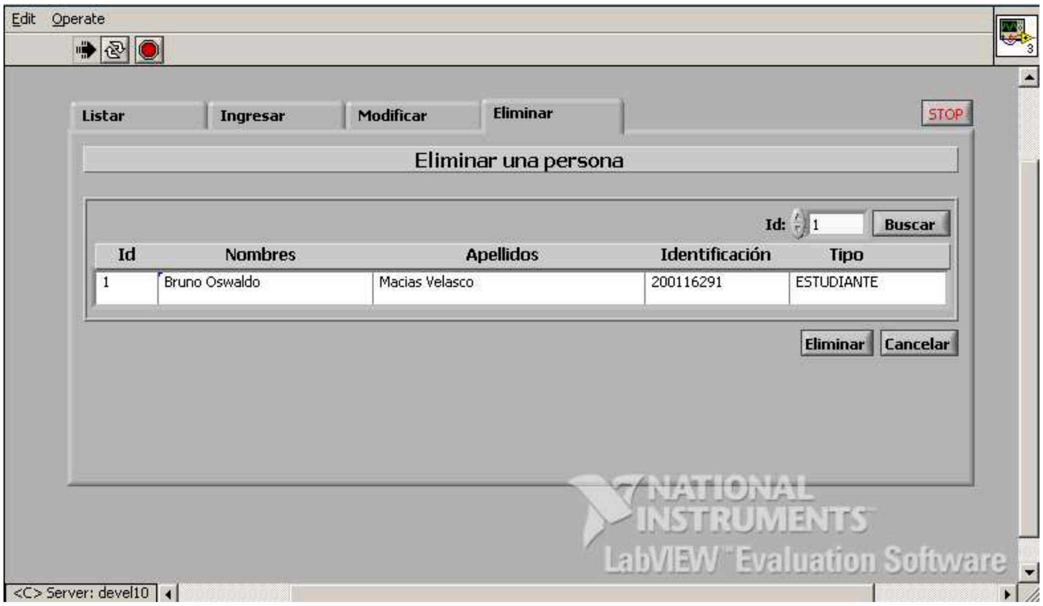
Se cargará automáticamente la información de la persona, como : Nombre, apellidos, identificación y tipo, .luego de cambiar alguno de estos campo damos clic en el botón “Modificar”, el sistema mostrará un mensaje de confirmación exitosa de modificación. Si no deseamos modificar la persona damos clic en el botón “Cancelar”



5.4. Eliminar una Persona

Podemos eliminar una persona dando clic en la pestaña “Eliminar” y escribimos el identificador de la persona y presionamos el botón “Buscar”

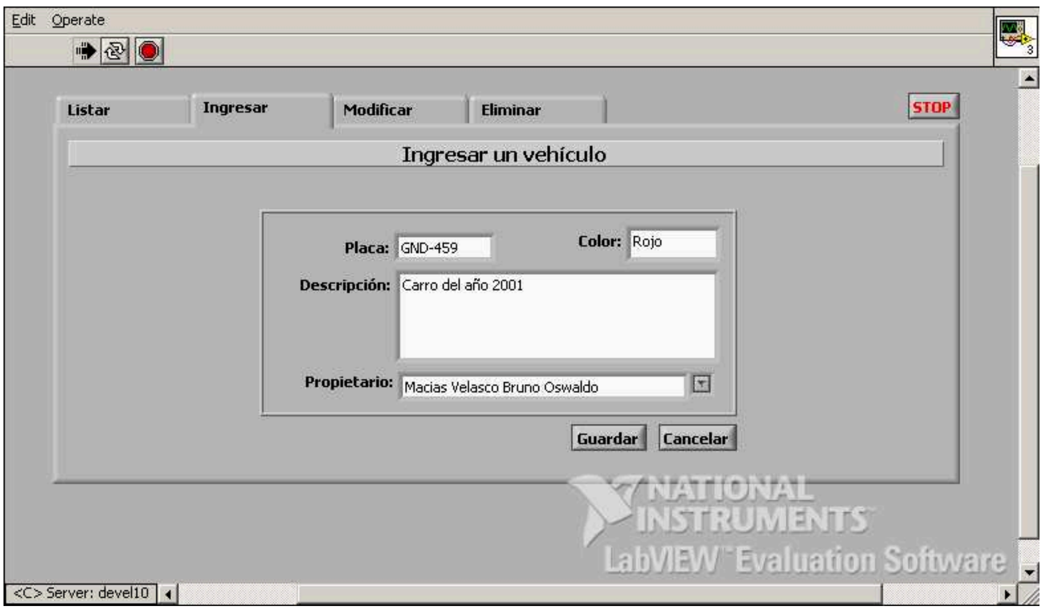
Se cargará automáticamente la información de la persona, como: id, Nombres, apellidos, identificación y tipo, .luego damos clic en el botón “Eliminar”, el sistema mostrará un mensaje de confirmación exitosa de Eliminación. Si no deseamos eliminar la persona damos clic en el botón “Cancelar”


Edit Operate
Listar
Ingresar
Modificar
Eliminar
STOP
Eliminar una persona
Id: 1
Buscar
Id
Nombres
Apellidos
Identificación
Tipo
1
Bruno Oswaldo
Macias Velasco
200116291
ESTUDIANTE
Eliminar
Cancelar
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
<C> Server: devel10

6. Vehículos

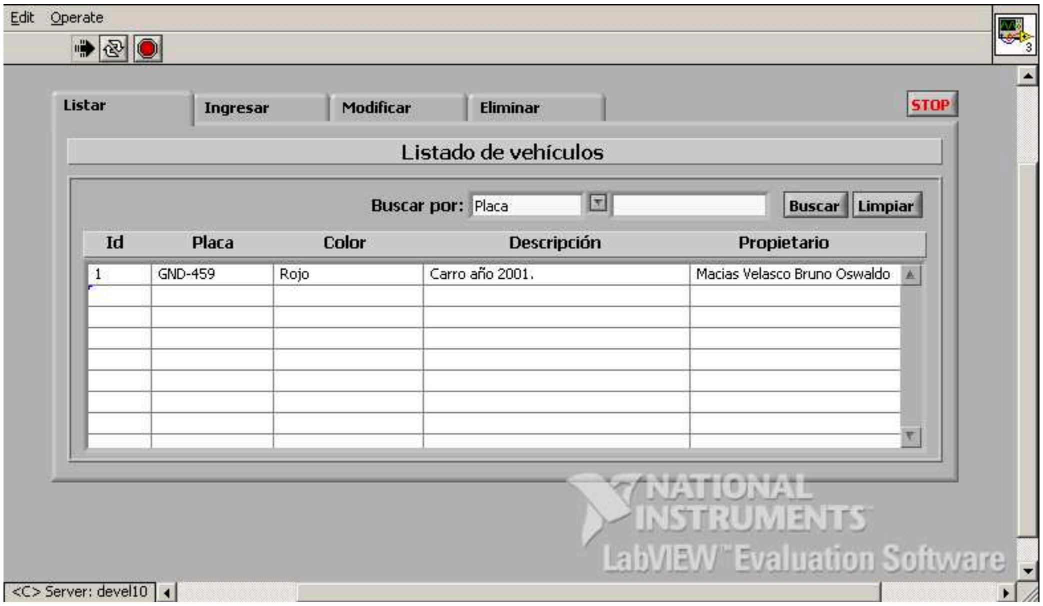
6.1. Ingresar Vehículos

Para ingresar un vehículo damos clic en la pestaña “Ingresar” del menú “Vehículos”, ingresamos la placa, el color, la descripción y propietario (persona) del vehículo y damos clic en el botón “Guardar”, si no deseamos ingresar damos clic en el botón “Cancelar”



6.2. Listar Vehículos

Podemos listar todos los vehículos creados, dando clic en el botón “Buscar”, si no deseamos ver todo el listado, podemos filtrar por la placa, color y propietario.



6.3. Modificar Vehículos

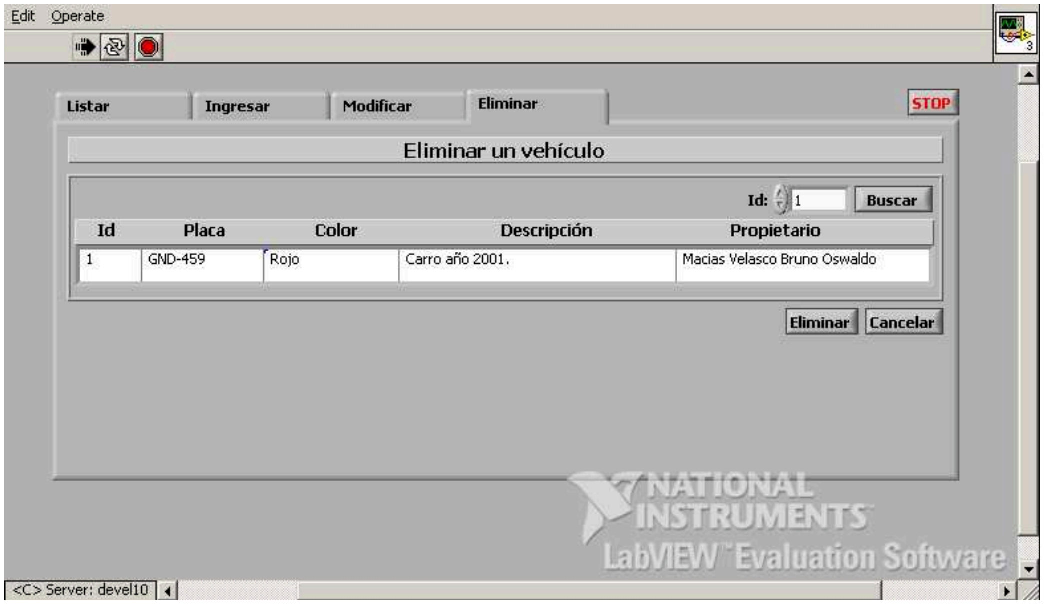
Podemos modificar un vehículo dando clic en la pestaña “Modificar” y escribimos el identificador del vehículo y presionamos el botón “Buscar”

Se cargará automáticamente la información del vehículo, como: Placa, color, descripción y propietario, .luego de cambiar alguno de estos campos damos clic en el botón “Modificar”, el sistema mostrará un mensaje de confirmación exitosa de modificación. Si no deseamos modificar el vehículo damos clic en el botón “Cancelar”.

6.4. Eliminar Vehículos

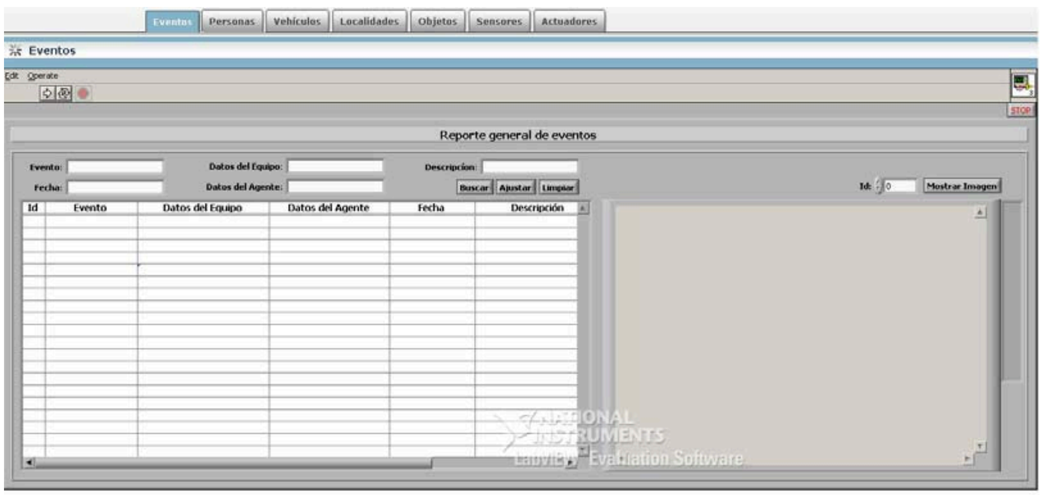
Podemos eliminar un vehículo dando clic en la pestaña “Eliminar” y escribimos el identificador del vehículo y presionamos el botón “Buscar”

Se cargará automáticamente la información del vehículo, como: Id, placa, color, descripción y propietario, .luego de cambiar alguno de estos campos damos clic en el botón “Eliminar”, el sistema mostrará un mensaje de confirmación exitosa de eliminación. Si no deseamos eliminar el vehículo damos clic en el botón “Cancelar”.


Edit Operate
Listar
Ingresar
Modificar
Eliminar
STOP
Eliminar un vehículo
Id: 1
Buscar
Id
Placa
Color
Descripción
Propietario
1
GND-459
Rojo
Carro año 2001.
Macias Velasco Bruno Oswaldo
Eliminar
Cancelar
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software
<C> Server: devel10

7. Eventos

Este menú permite consultar los eventos originados por los diversos sistemas de seguridades, en las cuales podemos realizar búsquedas por diferentes patrones como nombre del evento, datos del equipo, datos del agente, descripción y fecha.



El reporte muestra el id del evento, el nombre del evento, los datos del equipo, datos del agente, fecha del evento y descripción del evento, Además si tiene una imagen asociada se la puede consultar.

En los datos del equipo se muestra su localidad y en qué objeto se encuentra instalado con sus respectivos códigos.

En los datos del agente se muestra el tipo de agente (automóvil, estudiante, administrativo, visitante, etc.) con un detalle descriptivo del mismo.

En la descripción general es un campo auxiliar para poder información adicional a lo que un evento ha hecho.