UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DESARROLLO DE UNA LIBRERÍA EN LABVIEW PARA LA IMPLEMENTACIÓN DE SISTEMAS DE GESTIÓN DISTRIBUIDOS DE ALARMAS Y EVENTOS. APLICACIÓN A UN PROYECTO DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE DEFECTOS EN SOLDADURAS DE EJES DESTINADOS AL SECTOR DE LA AUTOMOCIÓN

AUTOR: Luis Dotor Alonso

TUTOR: Herrero Durá, Juan Manuel

Curso Académico: 2016-17

RESUMEN

El proyecto que se explica a continuación trata sobre el desarrollo de un toolkit en LabVIEW para la empresa Autis Ingenieros. El principal propósito de este es la gestión de alarmas y eventos en aplicaciones de ámbito industrial de manera efectiva y eficiente. Este toolkit consta de una librería de funciones y de un conjunto de herramientas integradas en el entorno de desarrollo de LabVIEW que lo dotan de un marco de trabajo (framework) cómodo y rápido para llevar a cabo los desarrollos.

Uno de sus principales objetivos es la inclusión de módulos de interfaz pre-programada (Paneles de visualización de alamas y eventos). Estos módulos consisten en aplicaciones flexibles con una interfaz gráfica completa para la clasificación, configuración y visualización de alarmas y eventos, además de otras funcionalidades. Gracias a su flexibilidad, estos módulos pueden cambiar fácilmente su apariencia sin afectar a su funcionalidad, permitiendo que cada proyecto se beneficie de una interfaz gráfica diferente. Además, están programados de manera escalable para facilitar su ampliación en el caso de que algún usuario necesite añadir alguna funcionalidad, o en el caso de que sea necesaria una actualización de la librería.

Además de lo anteriormente comentado, el toolkit presenta las siguientes prestaciones:

- Bases de datos de SQlite: el toolkit posee dos bases de datos personalizadas, una para eventos y otra para alarmas, con distintas funcionalidades para organizar la información de una manera concreta y específica a las necesidades de la empresa.
- Servidores de Alarmas/Eventos: aplicaciones integradas en el toolkit que gestionan todos los registros de alarmas y eventos, asegurando la concurrencia de todas las inserciones.
- Capacidad de implementación en sistemas distribuidos: los servidores anteriormente comentados permiten establecer comunicación con distintas aplicaciones al mismo tiempo utilizando el protocolo de comunicación TCP. Por ejemplo, se podrían utilizar los módulos de interfaz pre-programada en distintas aplicaciones dentro de una misma red, realizando peticiones a los servidores para consultar la información de la base de datos en paralelo. Además, el toolkit también dispone de funciones para realizar peticiones de inserción de alarmas y eventos de manera remota y distribuida.
- Asistentes de creación de bases de datos: aplicación autodocumentada e integrada en el entorno de desarrollo de LabVIEW para ayudar a los nuevos usuarios a configurar las bases de datos de la manera correcta.

- Asistentes de creación de nuevos paneles de Eventos/Alarmas: aplicaciones integradas en el entorno de desarrollo de LabVIEW para crear nuevos módulos pre-programados de Eventos y Alarmas. Estos asistentes permitirán la selección de distintas interfaces ya predefinidas en unas plantillas, así como la creación de nuevas plantillas para facilitar el trabajo en nuevos proyectos.
- Asistentes de configuración de paneles de Eventos/Alarmas: aplicaciones integradas en el entorno de desarrollo de LabVIEW para configurar los distintos módulos creados y así modificar algunas opciones de su comportamiento. Algunas de esas opciones son:
    - Cambio de Idioma.
    - Opciones de rendimiento.
    - Opciones de inicialización.
    - Opciones de actualización de la interfaz.
    - Opciones para la exportación de informes de históricos.

Por último, se expone su aplicación a Axle Welding. Un proyecto de visión artificial destinado al sector automovilístico.

ÍNDICE DE DOCUMENTOS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

Desarrollo de una librería en LabVIEW para la implementación de sistemas de gestión distribuidos de alarmas y eventos. Aplicación a un proyecto de visión artificial para la detección de defectos en soldaduras de ejes destinados al sector de la automoción.

Documento Nº 1: Memoria

AUTOR: Dotor Alonso, Luis

TUTOR: Herrero Durá, Juan Manuel

Curso Académico: 2016-2017

ÍNDICE

ÍNDICE DE FIGURAS

1. OBJETO DEL PROYECTO

El objetivo de este proyecto es el desarrollo de un toolkit de alarmas y eventos en LabVIEW que proporcione un marco de trabajo cómodo, sencillo y eficiente para desarrollar módulos de alarmas y eventos en distintas aplicaciones de carácter industrial para la empresa Autis Ingenieros S.L.U., disminuyendo así el tiempo de desarrollo de estos módulos para posteriores proyectos.

Además, se deberá demostrar su validez mediante su aplicación a Axle Welding, un proyecto de visión artificial en el que se detectan defectos en las soldaduras de ejes de transmisión destinados a la automoción.

2. MOTIVACIÓN

En la empresa Autis Ingenieros S.L.U. se desarrollan proyectos industriales de diversos tipos: control, visión artificial, monitorización de plantas, etc. En la gran mayoría de ellos se presenta la necesidad de gestionar alarmas/eventos intrínsecos al proyecto. Sin embargo, debido a que cada uno posee sus particularidades y que los tiempos de desarrollo son limitados, estos módulos no se pueden desarrollar con la escalabilidad necesaria para permitir su reutilización de manera cómoda en proyectos posteriores, resultando así en una considerable pérdida de tiempo y dinero en "reinventar la rueda". Debido a estos factores, la empresa se plantea la necesidad de desarrollar una librería/framework versátil y potente que permita desarrollar estos módulos de manera rápida y eficaz, dando solución a proyectos que demandan necesidades distintas.

Por otro lado, gran parte de la motivación proviene del reto que supone desarrollar en LabVIEW un complejo marco de trabajo que sirva para reducir el trabajo del resto de programadores en el desarrollo de futuros proyectos. Para poder implementar este desarrollo de manera exitosa, es necesario aplicar conceptos más avanzados de programación, y por ello ha sido necesario adquirir un conocimiento previo al desarrollo.

3. OBJETIVOS

Como bien se ha comentado anteriormente, el objetivo principal de este proyecto es el desarrollar, no solo una librería, sino también un marco de trabajo (framework) para ayudar a los programadores a abordar los módulos de alarmas y eventos. Parar alcanzar este propósito, el toolkit deberá estar provisto de:

- **Paleta de funciones:** el toolkit deberá disponer de un conjunto de VIs funcionales a fin de proveer a los programadores de una API de alto nivel para gestionar las alarmas y eventos de una forma personalizada. A continuación, se nombran las subpaletas a realizar:
  - Paleta de comunicación con el servidor.
  - Paleta de comunicación con los paneles de visualización.
  - Paleta de escritura.
  - Paleta de lectura.
  - Paleta de funciones remotas para sistemas distribuidos.
- **Servidores de Alarmas/Eventos:** procesos integrados en el toolkit que gestionan todos los registros de alarmas y eventos, asegurando la concurrencia de todas las inserciones. Debido a que SQLite no tiene un motor de base de datos cliente-servidor, se pretende proveer al toolkit de esta funcionalidad mediante LabVIEW.
- **Paneles de visualización de alarmas y eventos:** paneles pre-programados y configurables para visualizar y explotar la generación de alarmas y eventos.
- **Base de datos de eventos y alarmas:** diseñar dos bases de datos (Alarmas y eventos) flexibles, capaces de gestionar y albergar toda la información necesaria para los distintos proyectos.
- **Arquitectura distribuida (cliente-servidor):** debido a la gran diversidad de proyectos que se desarrollan en la empresa, en algunas ocasiones las aplicaciones de monitorización se instalan en más de un punto a la vez. Esto demanda que las aplicaciones tengan la capacidad de trabajar de manera distribuida con un punto central de información (Servidor).
- **Asistentes de creación de bases de datos:** el framework ha de disponer de una aplicación sencilla y manejable para ayudar a los programadores a definir una base de datos.
- **Asistente de creación de interfaces:** para fomentar la reutilización del trabajo que supone el diseño de una interfaz. El framework deberá proveer a los usuarios de una herramienta para definir nuevas interfaces que puedan ser reutilizadas para futuros proyectos. De esta manera, cuando se vaya a realizar un nuevo diseño, siempre se podrá partir de uno anterior, evitando así una cantidad considerable de esfuerzo e inspiración.
- **Asistentes de creación de nuevos paneles de Eventos/Alarmas:** aplicación que ha de dotar al framework de un asistente para la creación y gestión de paneles

en los proyectos. Deberá permitir dar de alta nuevos proyectos y en cada uno de ellos crear paneles de alarmas y eventos a partir de plantillas predefinidas.

- **Asistentes de configuración de paneles de Eventos/Alarmas:** aplicación de configuración integrada en el framework que permita modificar el comportamiento de los módulos de visualización de alarmas y eventos:
  - Cambio de Idioma.
  - Opciones de rendimiento.
  - Opciones de inicialización.
  - Opciones de actualización de la interfaz.
  - Opciones para la exportación de informes de históricos.
  - Opciones gráficas.

4. ANTECEDENTES DEL PROYECTO

4.1 LabVIEW

LabVIEW (acrónimo de Laboratory Virtual Instrumentation Engineering Workbench) (Wikimedia, LabVIEW, 2017) es una plataforma y entorno de desarrollo para diseñar sistemas con un lenguaje de programación visual gráfico. Este software fue diseñado por National Instruments en 1976 para el desarrollo de osciloscopios virtuales. No obstante, su gran potencial de programación le ha conducido a ser una plataforma recomendada para el desarrollo de sistemas (hardware y software) de control, diseño simulado o real. El lenguaje que utiliza este programa se denomina lenguaje G (Gráfico).

4.1.1 Formación: Cursos de LabVIEW (Cores)

Para poder abordar un proyecto de esta complejidad ha sido necesario un proceso de aprendizaje extenso que ha consistido en realizar una serie de cursos y en un proceso de asentamiento de los conceptos adquiridos participando en los desarrollos de aplicaciones de distinta naturaleza. A continuación, se muestran los cursos que se han realizado de manera completa:

- **LabVIEW Core 1:** este curso comienza con las bases de LabVIEW, ayudando a explorar el entorno de desarrollo, arquitecturas, flujo de datos etc.
- **LabVIEW Core 2:** este curso termina de asentar las bases del lenguaje G y enseña al programador a diseñar aplicaciones completas y autónomas con entorno de desarrollo gráfico integrado.
- **LabVIEW Core 3:** este tercer curso constituye el final del proceso de aprendizaje de lo que un desarrollador avanzado en LabVIEW debe dominar como mínimo (CLD, Desarrollador certificado en LabVIEW). En él se recoge la

formación necesaria para analizar los requisitos de una aplicación y seleccionar los patrones de diseño y estructuras de datos correctas. Además, se adquieren las habilidades necesarias para crear aplicaciones escalables, fáciles de leer y fáciles de mantener.

Además de estos últimos cursos, también se han tocado aspectos particulares de otros manuales de LabVIEW más específicos y avanzados:

- **Advanced Architectures in LabVIEW:** donde se ha aprendido a plantear aplicaciones basadas en "subpanels" y desarrollar Xcontrols.
- **LabVIEW Connectivity:** comunicación y transmisión de datos a través de distintos protocolos (TCP, UDP), tecnologías .NET, ActiveX, etc.
- **LabVIEW Performance:** conocer la gestión interna que hace LabVIEW de la programación "multi-hilo", así como de la declaración y reutilización de memoria cuando se manipulan las variables (Cables). Conocer estos conceptos más avanzados permite desarrollar las aplicaciones de una manera más óptima y eficiente.

4.1.2 Toolkits Privados

A continuación, se muestran los Toolkits que se han utilizado en el desarrollo de esta librería y que han sido desarrollados por la empresa Autis Ingenieros S.L.U.

1. **Modbus TCP**: toolkit privado que permite comunicar con cualquier dispositivo que utilice este protocolo de comunicación. Además, incorpora funciones de alto nivel para la transferencia de datos y la ejecución de VIs dinámicamente vía TCP.
2. **SQLite Autis API**: un Toolkit privado que funciona como una extensión de un toolkit público (el cual se comenta en el siguiente apartado). Básicamente, se utiliza para gestionar las distintas referencias (conexiones) a ficheros SQLite en aplicaciones donde es necesario gestionar varias bases de datos.
3. **Control de usuarios:** toolkit privado para gestionar usuarios con distintos privilegios en aplicaciones mediante una base de datos de control de usuarios.

4.1.3 Toolkits públicos

A continuación, se nombran las librerías públicas de LabVIEW que se han utilizado para el desarrollo de esta librería:

- **Gpower Toolsets:** librería de funciones de propósito general para LabVIEW.
- **OpenG Toolkit:** librería de funciones de propósito general para LabVIEW.
- **MGI Tools:** librería de funciones de propósito general para LabVIEW.
- **SQLite Library:** librería de funciones para acceder de manera sencilla a las dll de SQLite.
- **AMC (Asynchronous Message Communication):** es una API de uso general de LabVIEW para gestionar colas y enviar mensajes dentro de un proceso o entre procesos.

4.2 Bases de datos SQL

SQL (Structured Query Language) (Wikimedia, SQL, 2017) es un lenguaje específico de alto nivel que permite el acceso a bases de datos relacionales y que ofrece una gran cantidad de operadores para aplicar. Gracias a que posee una fuerte base teórica y a que está orientada al manejo de registros de manera conjunta (no de forma individual), permite una alta productividad con implicaciones muy notables en el desarrollo de código, sobre todo si lo comparamos con lenguajes de bajo nivel. Una de sus principales características es el manejo del cálculo relacional, lo cual le permite acceder de forma sencilla a la información de la base de datos mediante el uso de consultas o sentencias (Queries).

4.2.1 SQLite

SQLite (Wikimedia, SQLite, 2017) es un sistema de gestión de bases de datos relacionales proveniente de una librería compilada en C. A diferencia de muchos otros sistemas de gestión de bases de datos, SQLite no es un motor de base de datos cliente-servidor. Más bien, está embebido en el programa final.

4.3 Axle Welding

Axle Welding es un proyecto desarrollado por Autis Ingenieros S.L.U para Ford USA. Consiste en detectar mediante algoritmos de visión artificial defectos en las zonas de soldaduras de ejes de transmisión destinados al sector del automóvil.

4.3.1 Adquisición

La adquisición se fundamenta en un barrido de imágenes unidimensionales para finalmente, tras un ciclo de adquisición, obtener una imagen compuesta por cada sensor (ver Figura 1).

Los ejes que se inspeccionan constan de dos tipos de soldaduras: longitudinales y transversales a la dirección del eje. Es por ello que la adquisición consta de dos fases:

- **Adquisición longitudinal:** donde participan los sensores 7 y 8, los cuales tienen un rango de actuación más pequeño. En este caso, son los sensores los que se mueven durante el proceso de toma de imágenes y no el eje.
- **Adquisición trasversal:** en este caso es el eje el que rota sobre sí mismo mientras que los sensores 1, 2, 3, 4, 5 y 6 toman imágenes.

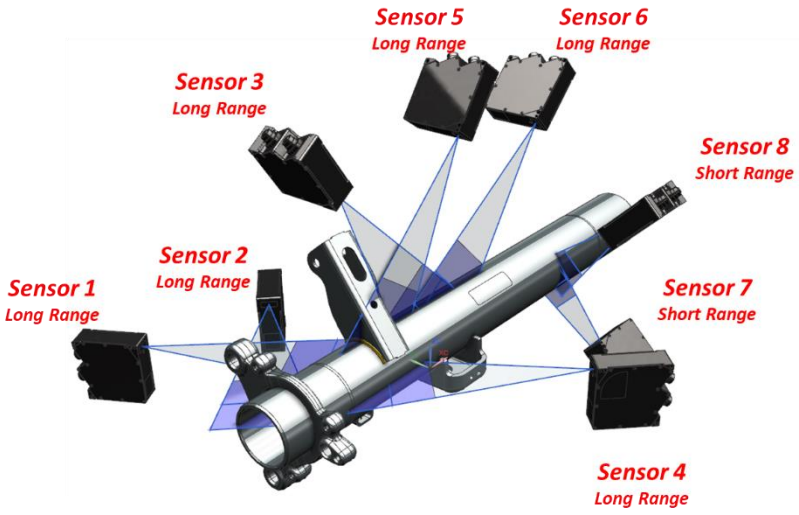

Figura 1: Sensores de adquisición. Fuente: (Axle Welding: Training, 2017)

4.3.2 Descripción del proceso

4.3.2.1 Adquisición

A continuación, se describe el proceso de inspección de los ejes. El cual consta de un tiemplo de ciclo total de 36 segundos.

1. PARADA: POSICIONAMIENTO DEL PALLET DEL CONVEYOR

Figura 2: Axle Welding (Parada). Fuente: (Axle Welding: Training, 2017)

Los ejes son transportados por la cinta encima de unos pallets hasta llegar a la estación de inspección. La parada se detecta gracias a un sensor de presencia situado en la estación.

2. CLAMPADO DEL EJE EN LA ESTACIÓN

Figura 3: Axle Welding (Clampado). Fuente: (Axle Welding: Training, 2017)

Una vez el eje llega a la estación, dos émbolos metálicos clampan el eje, asegurando su sujeción.

3. TRASLADO A LA ZONA DE INSPECCIÓN

Figura 4: Axle Welding (Posicionado en zona de inspección). Fuente: (Axle Welding: Training, 2017)

Se sube el eje de transmisión mediante un pórtico vertical a fin de situarlo en la zona de inspección para comenzar la adquisición.

4. INSPECCIÓN LINEAL: SENSORES 7 Y 8

Figura 5: Axle Welding (Inspección Lineal). Fuente: (Axle Welding: Training, 2017)

Se inicia una adquisición mediante el desplazamiento lineal de parte de la estructura metálica adquiriendo mediante 2 de los 8 sensores en las zonas de soldadura longitudinales al eje de inspección.

5. INSPECCIÓN LONGITUDITAL

Figura 6: Axle Welding (Inspección longitudinal). Fuente: (Axle Welding: Training, 2017)

Se produce un giro de 360 grados sobre el eje longitudinal de la pieza mientras adquieren el resto de sensores (1, 2, 3, 4, 5 y 6).

6. TRASLADO AL CONVEYOR

Figura 7: Axle Welding (Posicionamiento en la cinta). Fuente: (Axle Welding: Training, 2017)

Una vez finalizada la inspección, el eje baja por el pórtico hasta volver a posicionarse en el pallet de la cinta.

7. DESCLAPADO Y LIBERACIÓN DEL EJE

Figura 8: Axle Welding (Final del ciclo). Fuente: (Axle Welding: Training, 2017)

Finalmente, se desclampa el eje de transmisión y se libera el sistema para iniciar la salida del eje inspeccionado e iniciar el movimiento de la nueva unidad.

4.3.2.2 Procesado

Una vez se han adquirido todas las imágenes, el sistema informático analiza cada imagen para detectar mediante algoritmos de visión el estado de la soldadura. En caso de detectar alguna anomalía, el sistema comanda a un robot para coger el eje y situarlo en una cinta de ejes rechazados.

5. DESCRIPCIÓN DE LA SOLUCIÓN ELEGIDA

5.1 Introducción

En este apartado se abordará el desarrollo y se justificará el proyecto en sus distintas partes. Cabe destacar que esta librería trata de manera separada el desarrollo de alarmas y eventos. Esto es debido a que hay proyectos que incluyen módulos de alarmas, módulos de eventos o ambos. **Debido a que presentan muchas similitudes, gran parte de las explicaciones se realizarán de forma conjunta para ambos módulos excepto cuando sea necesario puntualizar.**

5.2 Justificación del uso de base de datos

Los módulos de alarmas y eventos están ligados a una generación de históricos a lo largo de la vida útil de los proyectos. Esto implica que sea necesario almacenar una gran cantidad de información proveniente de registros que posteriormente va a ser necesario explotar para obtener la máxima información posible. Debido a esta razón, es necesario un sistema de almacenamiento que proporcione herramientas y un lenguaje apropiado para desempeñar esta tarea, por ejemplo, las bases de datos relacionales (SQLite).

5.2.1 Elección de base de datos relacionales (SQL): SQLite

Actualmente, existen muchos modelos de bases de datos: jerárquicas, relacionales, multidimensionales, orientadas a objetos, documentales, etc. No obstante, los sistemas de bases de datos relacionales (SQL) son el modelo más utilizado actualmente para implementar las bases de datos ya planificadas. Esto implica la existencia de herramientas que tienen un mayor soporte y mejores suites de productos y add-ons para gestionarlas, algunas de libre acceso. Además de esto:

- Posee herramientas que garantizan evitar la duplicidad de registros.
- Garantiza la integridad referencial, así, al eliminar un registro, se eliminan todos los registros relacionados y dependientes.
- Favorece la normalización por ser más comprensible y aplicable.

Hoy por hoy, existen muchas alternativas para el uso de bases de SQL: MySQL, SQL Server, PostgreSQL etc. No obstante, en el proyecto que aquí se desarrolla, se ha optado por el uso de SQLite por diversas razones:

- Es realmente liviana, ya que únicamente consta de una librería compilada en C para acceder a la base de datos.
- Al ser una única biblioteca, trabaja de manera eficiente y le permite ser más rápida que otros sistemas SQL.

- Presenta gran portabilidad debido a que se puede ejecutar en muchas plataformas.
- SQLite es compatible con ACID, por tanto reúne los cuatro criterios de Atomicidad, Consistencia, Aislamiento y Durabilidad.
- Existe un gran bagaje en la empresa de Autis Ingenieros utilizando esta herramienta con buenos resultados, además de que existe un toolkit gratuito en LabVIEW que facilita mucho su uso.

5.3 Diseño de las bases de datos

Tanto la base de datos de eventos como la de alarmas están diseñadas como una combinación de tablas, vistas y triggers. Esto facilita su escalabilidad para realizar futuras modificaciones y permite de manera sencilla auto gestionar la base de datos. A continuación, se definen los conceptos anteriores:

- **Tablas**: hacen referencia al modelo de los datos y son similares a las hojas de cálculo. Su estructura principal se compone de:
  - **Registros (Filas):** que constituyen una nueva instancia del tipo de dato que almacena la tabla.
  - **Campos (Columnas):** cada uno de los tipos de datos que conforman las estructuras de datos de la tabla.
- **Vistas**: poseen exactamente la misma estructura que las tablas (filas y columnas) con la diferencia de que únicamente almacenan la definición pero no los datos. Las vistas obtienen los datos a base de combinar otras tablas ya existentes en la base de datos.
- **Triggers**: los triggers o disparadores son objetos que se asocian con las tablas y se almacenan en la base de datos. Se ejecutan cuando se produce algún evento en la tabla a la que van asociados. Los eventos más típicos para disparar los triggers son las operaciones de: INSERT, DELETE y UPDATE.

5.3.1 Base de datos de alarmas

Esta base datos está compuesta de: 5 tablas, 3 vistas y 5 triggers.

Su diseño está pensado para que en funcionamiento normal, todas las inserciones se realicen contra la tabla HISTORY_ALARMS para que posteriormente, mediante triggers, los datos se distribuyan a las otras tablas: HISTORY_DURATION_ALARMS y ACTIVE_ALARMS.

Para explotar los datos de las tablas HISTORY_DURATION_ALARMS, HISTORY_ALARMS y ACTIVE_ALARMS se han creado 3 vistas VIEW_HISTORY_DURATION_ALARMS, VIEW_HISTORY_ALARMS, y VIEW_ACTIVE_ALARMS, respectivamente. Todas ellas son la combinación de unir las 3 tablas anteriormente nombradas con la tabla CONFIGURATION_ALARMS, en la cual se tienen que dar de alta todas las alarmas que se vayan a utilizar en el proyecto.

5.3.1.1 Tablas y triggers

La Base de datos de alarmas está compuesta por 5 tablas:

CONFIGURATION_ALARMS

En esta tabla se han de configurar cada una de las alarmas intrínsecas de cada proyecto. En caso de no estar configuradas, las vistas de la base de datos aparecerán vacías. Cada fila constituye la configuración de una alarma y contiene toda aquella información que no es necesario repetir con cada registro. Además, cada alarma se identifica mediante una combinación unívoca de código y grupo, es decir, dos alarmas pueden tener el mismo código o el mismo grupo pero no ambos parámetros a la vez:

- **Código:** número identificador de la alarma.
- **Grupo:** tag identificador de la alarma. Se plantea para los proyectos donde es necesario categorizar más la configuración de alarmas.
- **Descripción:** campo descriptivo asociado al tipo de alarma.
- **Actuación:** campo descriptivo asociado al procedimiento a seguir cuando salta una alarma.
- **Habilitada_Activas:** campo para habilitar la visualización de la alarma en las vistas de alarmas activas.
- **Habilitada_Historicos:** campo para habilitar la visualización de la alarma en las vistas de históricos.

Esta tabla tiene asociado un trigger:

- **IGNORE_NEW_ALARM_WHEN_REPEATED:** en ningún caso debe haber dos alarmas con la misma combinación de código y grupo, es por ello que este trigger ignora todas las inserciones que repiten el código y grupo de una alarma ya insertada en esta tabla.

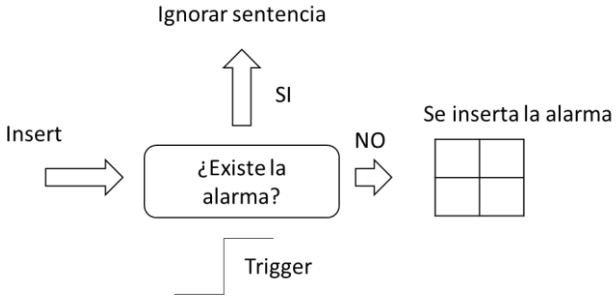

Figura 9: pseudocódigo de funcionamiento tabla CONFIGURATION_ALARMS

CONF_DATA_BASE

Esta tabla es usada para guardar la configuración de la base de datos. Actualmente, solo existe un campo de configuración para especificar el número de filas máximo de la tabla HISTORY_ALARMS y HISTORY_ALARMS_DURATION. En caso de sobrepasar esta cifra, el servidor de alarmas ejecutará una tarea para migrar el exceso de registros a ficheros de texto plano (Se comentará más adelante).

HISTORY_ALARMAS

Contiene el histórico de cambios que se producen en las alarmas. Es una de las tablas más importantes, debido a que es ella la que recibe todas las sentencias y a partir de esta, los datos se migran a las tablas "HISTORY_DURATION_ALARMS" y "ACTIVE_ALARMS" mediante Triggers. Los campos de cada registro son:

- **Código:** número identificador de la alarma.
- **Grupo:** tag identificador de la alarma.
- **Estado:** valor booleano para especificar si la alarma está activa o no.
- **Instante:** valor temporal en el que se ha producido la alarma.
- **Info:** campo adicional para guardar cualquier información relacionada con la generación de la alarma. Es un campo un poco flexible que puede ser usado de distintas maneras en función de las necesidades del proyecto.

Esta tabla contiene un total de 4 triggers:

- **UPDATE_ACTIVE_ALARMS_WHEN_NEW_ALARM_IS_ON**: este trigger inserta las alarmas de la tabla ACTIVE_ALARMS cuando anteriormente no existían en tabla. Además, genera una inserción en la tabla HISTORY_DURATION_ALARMS dejando vacíos los campos: Instante_final y duración.
- **UPDATE_ACTIVE_ALARMS_WHEN_NEW_ALARM_IS_OFF:** este trigger elimina las alarmas de la tabla ACTIVE_ALARMS cuando anteriormente estaban activas y en la nueva inserción pasan a estar desactivadas.
- **IGNORE_REPEATED_ALARMS:** este trigger ignora todas las alarmas repetidas que se insertan en HISTORY_ALARMS. Una alarma se considera repetida cuando coincide en código, grupo y estado.

- **UPDATE_HISTORY_DURATION_ALARMS:** este trigger actualiza los registros ya creados en la tabla HISTORY_DURATION_ALARMS para completar los campos instante final y duración cuando se produce el flanco de bajada de una alarma.

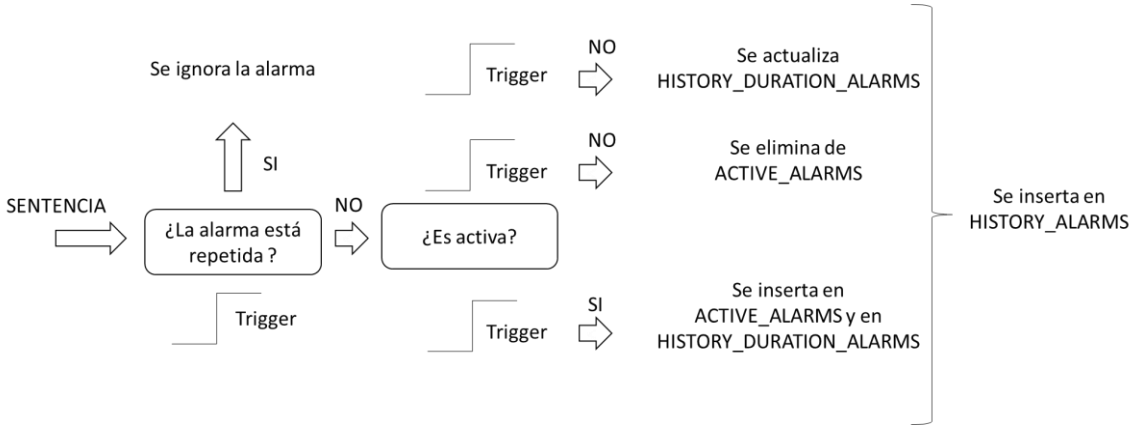

Figura 10: Diagrama funcionamiento de trigger en HISTORY_ALARMS

HISTORICOS_DURATION_ALARMS

Esta tabla de histórico contiene una fila por cada inicio y fin de una alarma. Cuando llega el Inicio de una alarma, se genera un nuevo registro. Sin embargo, cuando llega el fin de una alarma se actualiza el registro asociado a su inicio. Los campos que posee esta tabla son:

- **Instante_inicial:** instante inicial en el que se ha producido la alarma.
- **Instante_final:** instante final en el que se ha producido la alarma.
- **Duración:** duración en segundos de la alarma.
- **Estado:** valor booleano para especificar si la alarma está activa o no.
- **Código:** número identificador de la alarma.
- **Grupo:** tag identificador de la alarma.
- **Info:** campo adicional para guardar cualquier información (igual que en el caso de las HISTORY_ALARMS).

ACTIVE_ALARMS

Esta tabla contiene todas las alarmas activas que se han insertado en la base de datos. Cada vez que se inserta una alarma en HISTORY_ALARMS, esta se inserta en la tabla ACTIVE_ALARMS cuando antes no estaba activa. Del mismo modo, cuando se inserta una alarma desactivada, esta se borra de la tabla ACTIVE_ALARMS si antes estaba activa. Los campos que posee esta tabla son:

- **Código:** número identificador de la alarma.
- **Grupo:** tag identificador de la alarma.
- **Estado:** valor booleano para especificar si la alarma está activa o no.
- **Instante**: valor temporal en el que se ha producido la alarma.
- **Info:** campo adicional para guardar cualquier información.

5.3.1.2 Vistas

La base de datos de alarmas está provista de 3 vistas:

- **VIEW_ACTIVE_ALARMS:** es el producto de combinar las tablas CONFIGURATION_ALARMS y ACTIVE_ALARMS. Las alarmas registradas en ACTIVE_ALARMS solo aparecerán en la vista siempre y cuando estén contempladas en la tabla CONFIGURATION_ALARMS y estén habilitadas en las opciones de visualización (HABILITADA_ACTIVAS). Los campos resultantes para la visualización son: instante, código, descripción, estado, actuación, grupo e info.
- **VIEW_HISTORY_ALARMS:** es el producto de combinar las tablas CONFIGURATION_ALARMS y HISTORY_ALARMS. Las alarmas registradas en HISTORY_ALARMS solo aparecerán en la vista siempre y cuando estén dadas de alta en la tabla CONFIGURATION_ALARMS y estén habilitadas en las opciones de visualización (HABILITADA_HISTORICOS). Los campos resultantes para la visualización son: instante, código, descripción, estado, actuación, grupo e info.
- **VIEW_HISTORY_DURATION_ALARMS:** es el producto de combinar las tablas CONFIGURATION_ALARMS y HISTORY_DURATION_ALARMS. Las alarmas registradas en HISTORY_DURATION_ALARMS solo aparecerán en la vista siempre y cuando estén dadas de alta en la tabla CONFIGURATION_ALARMS y estén habilitadas en las opciones de visualización (HABILITADA_HISTORICOS). Los campos resultantes para la visualización son: Instante_inicial, Instante_final, duración, estado, descripción, actuación, código, grupo e info.

5.3.2 Base de datos de eventos

Esta base datos está compuesta de: 5 tablas, 1 vistas y 4 triggers. Su diseño está pensado para que, en funcionamiento normal, todas las inserciones se realicen contra la tabla "HISTORY_EVENTS". Previamente a la utilización de la base de datos, se han de definir los eventos y configurarlos. Para cada evento el usuario deberá configurar:

- **Tipo de evento (Tabla CONFIGURATION_EVENT_TYPES):** destinado a definir la severidad del evento. Por ejemplo, eventos de error, informativos, de advertencia, etc.
- **Grupo del evento (Tabla CONFIGURATION_EVENT_GROUPS):** destinado a una categorización personalizada por el usuario e intrínseca del proyecto.
- **Configuración propia del evento (Tabla CONFIGURATION_EVENTS):** es en esta tabla donde se configura cada evento por separado y donde se le asigna un grupo y un tipo (definidos anteriormente).

Para explotar los datos de la tabla HISTORY_EVENTS se ha creado una vista llamada VIEW_HISTORY_EVENTS, fruto de la combinación de las tablas HISTORY_EVENTS, CONFIGURATION_EVENT_TYPE, CONFIGURATION_EVENT_GROUP y CONFIGURATION_EVENTS.

5.3.2.1 Tablas y triggers

CONF_DATA_BASE

Esta tabla es usada para guardar la configuración de la base de datos. Actualmente, solo existe un campo de configuración para especificar el número de filas máximo de la tabla HISTORY_EVENTS. Similar a la tabla existente en la base de datos de alarmas.

CONFIGURATION_EVENTS

Esta tabla es similar a la tabla CONFIGURATION_ALARMS en la base de datos anterior. En ella, el usuario deberá dar de alta todos los eventos que se vayan a utilizar en el proyecto. La información que contiene es, además de la identificación de evento, toda aquella información referente a un evento que solo es necesario almacenar una vez. A continuación, se definen las columnas de la tabla:

- **Name:** nombre descriptivo del evento.
- **Code:** código identificador del evento. **No pueden haber dos eventos con el mismo código.**
- **Group_Code:** código de grupo al que pertenece el evento (los grupos se configuran en la tabla CONFIGURATION_EVENT_GROUPS).
- **Parameters:** son los argumentos que va a tener un evento y se colocan entre corchetes uno detrás de otro. Cuando se configuran los argumentos es importante poner un nombre descriptivo en la configuración. Este nombre es importante ya que posteriormente se va a utilizar como tag para filtrar los eventos. Por ejemplo: Para un evento de nombre "Nuevo_Coche_Entrando", la configuración de los argumentos podría ser [COLOR][MODELO][VARIANTE].
- **Description:** campo reservado para describir el evento.
- **Enable:** campo booleano que habilita y deshabilita el evento para ser insertado en la base de datos.
- **Visible:** campo booleano que habilita y deshabilita el evento para ser visible en la vista.

La actual tabla posee un trigger. La principal funcionalidad de este es garantizar que el usuario no realice una mala configuración de los eventos:

- **IGNORE_CONF_EVENT_WHEN_GROUP_OR_TYPE_NOT_EXIST:** Ignora la sentencia cuando el grupo de evento o tipo de evento no existe en la tabla CONFIGURATION_EVENT_GROUPS y CONFIGURATION_EVENT_TYPES respectivamente.

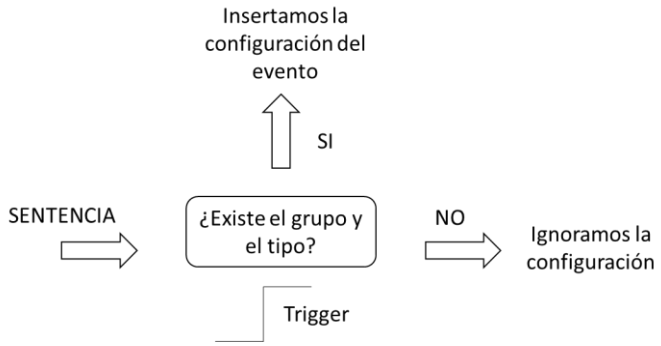

Figura 11: Funcionamiento triggers tabla CONFIGURATION_EVENTS

CONFIGURATION_EVENTS_GROUPS

En esta tabla, los usuarios deberán dar de alta los grupos de eventos. Estos afectan a la manera en la que se van a visualizar los eventos en los paneles de visualización de eventos. Las columnas que posee esta tabla son:

- **Name:** campo descriptivo del nombre del grupo.
- **Color:** código RGB correspondiente al color con el que se va a visualizar el grupo en el panel de visualización de eventos.
- **Enable:** campo booleano que habilita y deshabilita todos los eventos del grupo para ser insertados en la base de datos.
- **Visible:** campo booleano que habilita y deshabilita todos los eventos del grupo para ser visibles en la vista.

CONFIGURATION_EVENTS_TYPES

En esta tabla, los usuarios deberán dar de alta los tipos de eventos. Al igual que en los grupos, estos afectan a la manera en la que se van a visualizar los eventos en los paneles de visualización de eventos. Las columnas que posee esta tabla son:

- **Name:** campo descriptivo del tipo de evento.
- **Icon:** índice de icono asociado al evento. Los iconos se encuentran en el directorio raíz del toolkit y afectan a la visualización en el panel de visualización de eventos, ya que hacen referencia a los iconos que va a cargar el árbol resumen de eventos. El usuario puede añadir iconos a esta carpeta añadiéndoles un índice nuevo y cargándolo en la base de datos (el índice es el número que aparece al principio del nombre del icono).
- **Severity:** orden de importancia del evento. Afecta al orden en el que aparecen los tipos de eventos en el árbol resumen del panel de eventos.
- **Enable:** campo booleano que habilita y deshabilita todos los eventos del tipo correspondiente para ser insertado en la base de datos.
- **Visible:** campo booleano que habilita y deshabilita el todos los eventos del tipo correspondiente para ser visible en la vista.

Por defecto, la base de datos de eventos se crea con 3 tipos de eventos: Error, Advertencia e Información. No obstante, el usuario puede eliminarlos y/o crear todos los que necesite. La Figura 12 muestra los iconos disponibles, los cuales se pueden encontrar en el directorio: …\\LabVIEW 20XX\vi.lib\Autis Ingenieros S.L.U\Alarmas_y_Eventos\config\Symbols.

Figura 12: Iconos disponibles para la configuración del árbol de eventos

HISTORY_EVENTS

Esta es la tabla de histórico de eventos, la cual contiene todos los eventos que se insertan a lo largo de la vida del proyecto. En funcionamiento normal (una vez configurada la base de datos) todos los registros se realizan en esta tabla. Las columnas que posee son:

- **Code:** código del evento.
- **Date:** instante en el que se ha producido el evento.
- **Parameters:** valor de los argumentos.

Esta tabla tiene asociados 3 triggers:

- **IGNORE_EVENT_TYPE_WHEN_NOT_ENABLE:** ignora las inserciones de los eventos pertenecientes a un determinado tipo cuando está deshabilitado en la tabla CONFIGURATION_EVENT_TYPES.
- **IGNORE_EVENT_WHEN_NOT_ENABLE:** ignora las inserciones de los eventos pertenecientes a un determinado grupo cuando está deshabilitado en la tabla CONFIGURATION_EVENT_GROUPS.
- **IGNORE_GROUP_WHEN_NOT_ENABLE:** ignora las inserciones de un determinado evento cuando está deshabilitado en la tabla CONFIGURATION_EVENTS.

A continuación, la Figura 13 muestra el diagrama conceptual de los triggers de esta tabla.

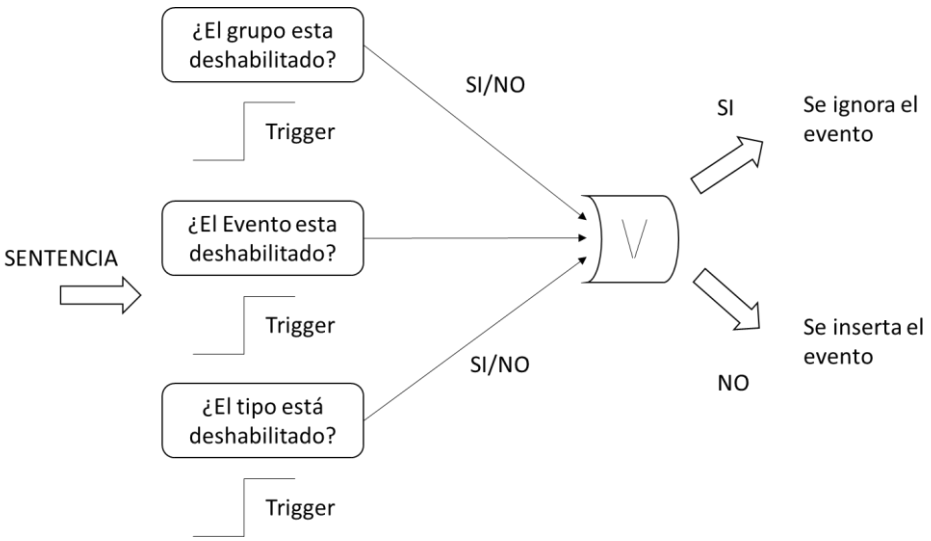

Figura 13: Funcionamiento triggers en tabla HISTORY_EVENTS

5.3.2.2 Vistas

La base de datos de Eventos posee 1 vista:

- **VIEW_HISTORY_EVENTS**: es el producto de combinar las tablas HISTORY_EVENTS, con CONFIGURATION_EVENT_TYPE, CONFIGURATION_EVENT_GROUP y CONFIGURATION_EVENTS. Los eventos registrados en HISTORY_EVENTS solo aparecerán en la vista siempre y cuando estén contemplados y habilitados en las 3 tablas de configuración (eventos, tipos y grupos). Las columnas resultantes de la vista son:
  - **Name:** nombre del evento.
  - **Date:** instante en el que se ha producido el evento.
  - **Group_Name:** nombre del grupo al que pertenece el evento.
  - **Group_Code**: código del grupo al que pertenece el evento.
  - **Color:** código RGB del grupo del evento.
  - **Event_type:** nombre del tipo de evento al que pertenece.
  - **Icon:** índice del icono de tipo de evento.
  - **Description:** descripción del evento.
  - **Parameter:** valor de los argumentos cuando se insertó el evento.

5.4 Desarrollo en LabVIEW

5.4.1 Introducción: Arquitectura Cliente-Servidor

El principal objetivo de este toolkit es facilitar el desarrollo de los módulos de alarmas y eventos en distintos proyectos (con módulo se hace referencia a todas las partes de la aplicación que interactúan de alguna forma con las alarmas y eventos), facilitando desde la creación de la base de datos hasta el diseño de la interfaz gráfica.

De todos los aspectos y funcionalidades que posee el toolkit los más importantes son:

- Servidor de alarmas y eventos para gestionar el acceso a la base de datos y asegurar la concurrencia de las inserciones.
- Funciones de escritura de alto nivel que comuniquen con el servidor para hacer peticiones de escritura.
- Paneles de visualización de alarmas y eventos con todas las funcionalidades programadas para explotar toda la información de la base de datos. (Estos paneles utilizan las funciones de lectura del toolkit).
- Base de datos bien estructura y diseñada.

Se destacan estos últimos puntos ya que mediante ellos se puede abordar la mayor parte de funcionalidad de los módulos en la mayoría de los proyectos.

Desde la empresa Autis Ingenieros se realizan una gran cantidad de aplicaciones con gran variedad de arquitecturas. En general, podemos destacar dos tipos de aplicaciones:

- Aplicaciones monolíticas: donde todo está en la misma aplicación. Este caso funciona bien para aplicaciones pequeñas con pocos subprocesos.
- Aplicaciones distribuidas: generalmente aplicaciones grandes donde se desacoplan los procesos críticos o donde se trabaja con varios Servidores o PCs al mismo tiempo por cuestión de necesidades de Hardware.

El toolkit necesita poder amoldarse a los dos estilos de aplicaciones existentes (monolíticas y distribuidas). La manera de poder conseguir esto es gracias a una arquitectura Cliente-Servidor TCP "Flexible", es decir, despendiendo de cómo se configure el toolkit, el usuario puede de manera sencilla poner un puerto TCP a la escucha, o no (cuando no sea necesario, por ejemplo en sistemas de un solo fichero ".exe"). De esta manera, cuando los sistemas son distribuidos, las aplicaciones que contengan los paneles de visualización de alarmas y eventos (Clientes) únicamente tendrán de realizar peticiones TCP de información a las aplicaciones que contengan el servidor de alarmas/eventos (Servidores), pudiendo estar las aplicaciones 100% separadas en un mismo PC o en PCs distintos de una misma red. Otra de las ventajas es que se pueden tener varios puntos de visualización, es decir, varios clientes con un único punto de inserción de alarmas (el servidor). Este último ejemplo es el caso de la aplicación de este toolkit sobre la Control_Room, un proyecto que monitoriza todo el túnel de proveedores de Ford Valencia, donde hay varias pantallas de monitorización y un servidor central.

A continuación, se muestra un esquema del funcionamiento de la arquitectura.

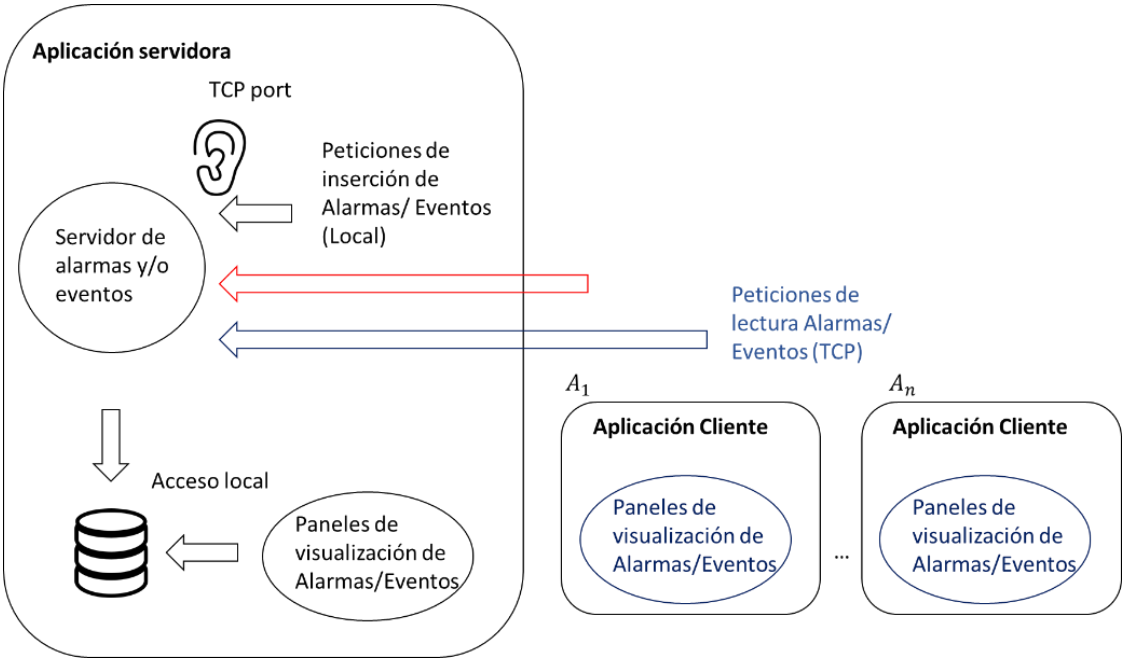

Figura 14: Esquema de funcionamiento de la arquitectura del toolkit

5.4.2 Servidor

Tanto el servidor de alarmas como el de eventos no son más que subprocesos independientes o hilos paralelos a la ejecución de la aplicación que se lanzan dinámicamente mediante las funciones del VI SERVER. Una vez lanzado el servidor, este se queda indefinidamente en segundo plano hasta que se le ordena el cierre.

Internamente está programado mediante una máquina de estados con colas donde cada estado desempeña una función en particular. Algunos de ellos se llaman desde fuera utilizando la paleta funciones de comunicación con el servidor. Estas funciones acceden a la referencia de la cola del servidor para posteriormente encolar los estados demandados.

5.4.2.1 Funciones del servidor: Estados

INICIALIZACIÓN VI DINÁMICOS Y SERVIDOR TCP

Un VI dinámico no es más que el nombre que se le da a una función que puede ser ejecutada de forma dinámica por el servidor TCP del toolkit ModBus TCP. La ventaja es que estas funciones pueden ser llamadas por aplicaciones externas para recibir información. Estas funciones se explicarán más adelante.

El servidor de alarmas y eventos utiliza las funciones del toolkit ModBus TCP para abrir a su vez un servidor TCP en caso de que se requiera. En muchas ocasiones, la aplicación que alberga el servidor de Alarmas/Eventos ya posee un servidor TCP de este Toolkit, en cuyo caso el servidor de Alarmas/Eventos únicamente tiene que dar de alta una serie de VI dinámicos para ese servidor TCP.

ESTABLECER CONEXIÓN CON LA BASE DE DATOS

Esto se realiza mediante el uso de tres estados:

- **Set_BBDD_Path:** Estado utilizado para que el usuario le comunique al servidor la ruta de la base de datos que se está utilizando.
- **Open_BBDD:** Establece una conexión con la base de datos y la mantiene en memoria.
- **Close_BBDD:** Cierra la conexión con la base de datos.

RECIBIR DATOS Y REGISTRARLOS MEDIANTE SENTENCIAS DE ESCRITURA

Mediante la funciones de escritura de la paleta, los usuarios son capaces de llamar a los estados “Insert Alarms” o “Insert_Events”. Estos estados reciben la información de las alarmas y eventos que se desean insertar para posteriormente generar las sentencias necesarias para ello. Una vez generadas, los servidores encolan internamente un estado llamado “Execute Query” que ejecuta todas las sentencias generadas.

Cada vez que se inserta una alarma o evento en la base de datos, el servidor actualiza una FGV (Variable Global Funcional) utilizada para comunicar a los distintos paneles de visualización que se ha modificado la base de datos y que, por tanto, se puede hacer una actualización de su interfaz.

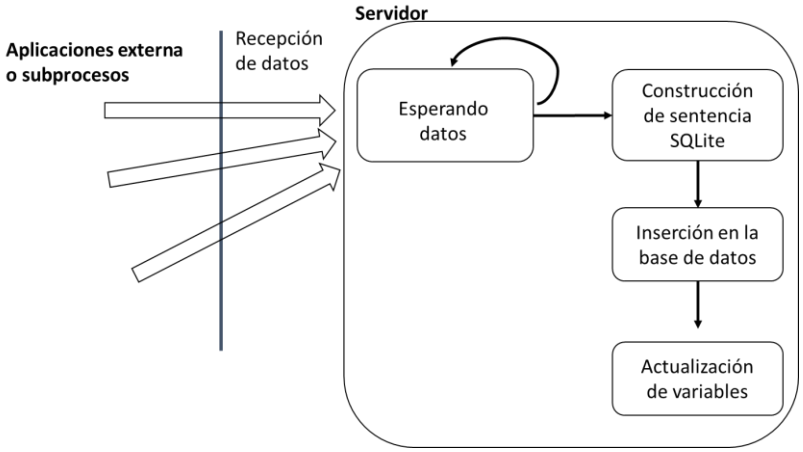

Figura 15: Pseudocódigo de recepción de datos y registro en la base de datos

MIGRAR LOS DATOS DE LA BASE DE DATOS A FICHEROS DE TEXTO

Para evitar que la base de datos crezca indefinidamente, el servidor ejecuta un estado cada cierto tiempo para migrar el exceso de históricos, en el caso de que sea necesario (como se comentó anteriormente, este límite es configurable desde la base de datos tanto para eventos como para alarmas). Básicamente, este estado coge el exceso de registros más viejos sobre el límite establecido y los copia a un fichero de texto por meses (los meses en los que se han producido esas alarmas/eventos).

5.4.3 Paneles de visualización de alarmas y eventos

En el toolkit encontramos dos paneles de visualización, uno para alarmas y otro para eventos. En este apartado se procederá a explicar aspectos conceptuales de estructura y programación y no tanto de uso, debido a que eso ya se realiza en la guía de usuario. Se recomienda al lector leer primero el apartado 2 de este último documento para entender mejor ciertos aspectos del apartado que aquí se desarrolla.

Debido a que los paneles de alarmas y eventos presentan similitudes, se explicarán conjuntamente los factores comunes, puntualizando en las diferencias cuando sea necesario en cada caso.

5.4.3.1 Introducción

Los paneles de alarmas y eventos son aplicaciones independientes con interfaz gráfica integrada para explotar la información de la base de datos de alarmas y eventos.

Estos paneles pueden modificar su interfaz y aspectos de su funcionamiento (configuración) sin la necesidad de realizar ninguna modificación en el código. Esto les proporciona una gran flexibilidad para adaptarse a la gran mayoría de proyectos. No obstante, estos paneles no dejan de ser una plantilla que se puede llegar a modificar en caso de que algún proyecto lo requiera.

5.4.3.2 Funcionamiento: Concepto

Ambos paneles de alarmas y eventos se basan en explotar la información presente en las vistas de la base de datos. De hecho, en la base de datos de alarmas existen tres vistas (VIEW_ACTIVE_ALARMS, VIEW_HISTORY_ALARMS, VIEW_HISTORY_DURATION_ALARMS) las cuales corresponden a las tres tablas del panel de visualización de alarmas (Activas, históricos e Históricos Duración). Sin embargo, en el panel de visualización de eventos existe solo una tabla de históricos correspondiente a la única vista de la base de datos (VIEW_HISTORY_EVENTS).

Ambos paneles poseen controles de diversos tipos que les permiten aplicar filtros de unas maneras u otras (fecha, descripción, código, etc.) para cambiar la información visible en estas tablas, sacándoles así el mayor provecho posible. A continuación, se muestran un par de imágenes con estas interfaces:

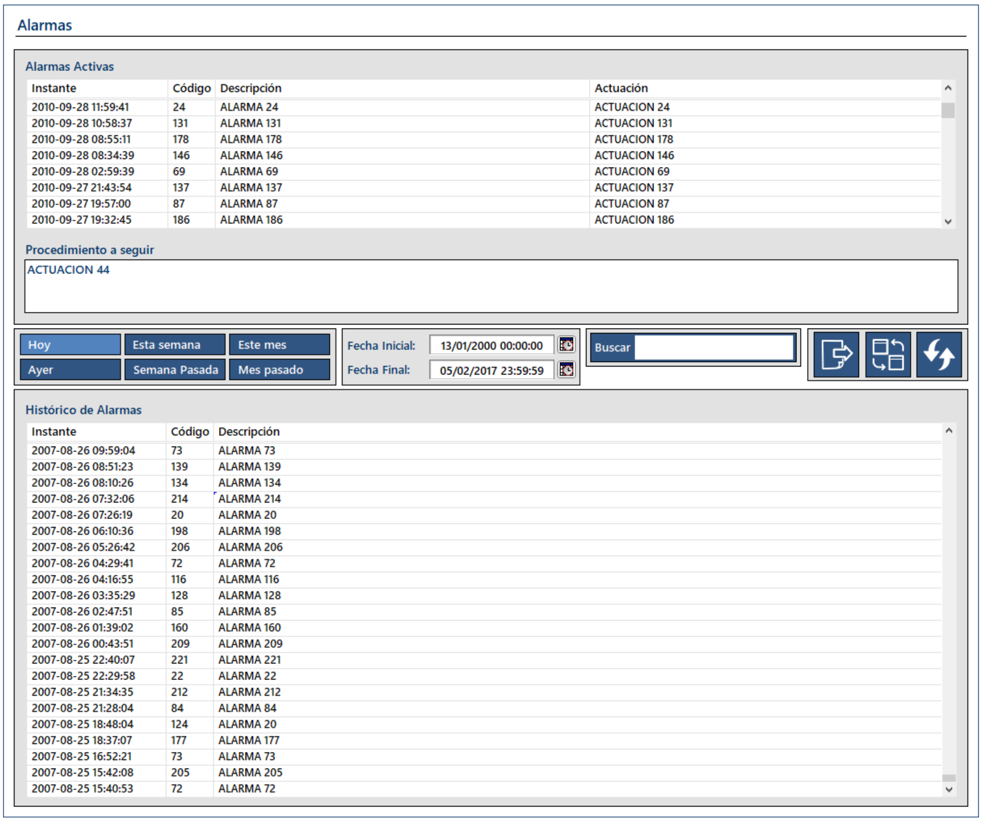

Figura 16: Interfaz gráfica del panel de alarmas

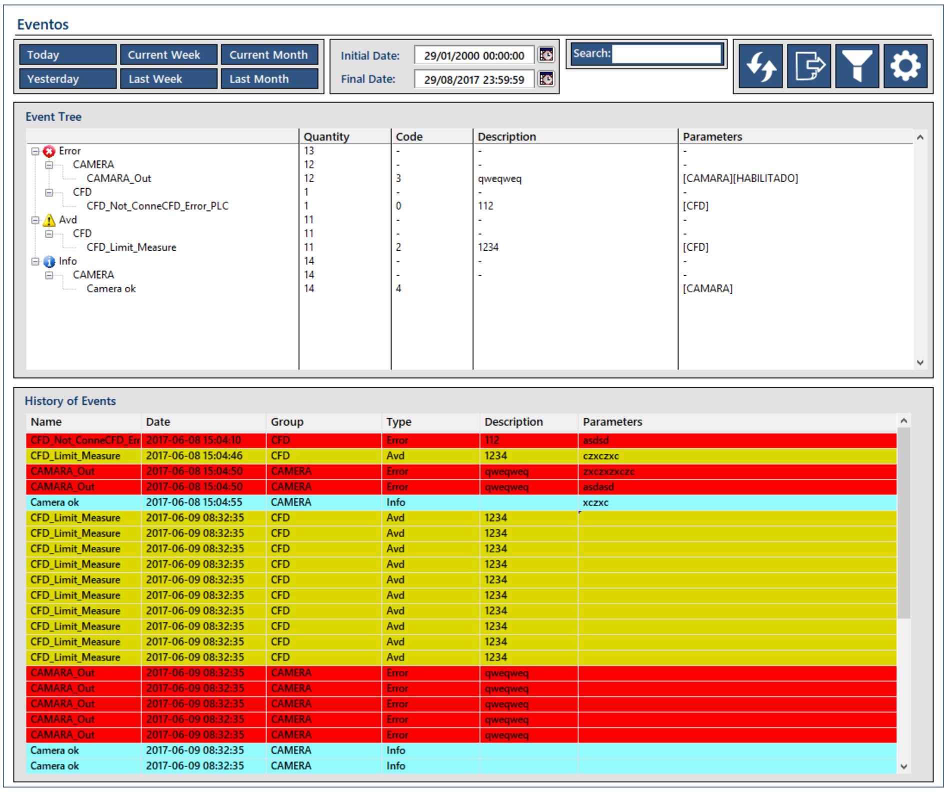

Figura 17: Interfaz gráfica del Panel de eventos

5.4.3.3 Estructura y funciones

Tanto el panel de alarmas como el de eventos presentan una estructura productor-consumidor combinado con máquina de estados. Por un lado, el bucle consumidor atiende a todos los eventos de interfaz de usuario encolando los estados necesarios en el consumidor, el cual alberga la máquina de estados. La Figura 18 muestra una imagen de la estructura del código.

La funcionalidad principal de estos paneles se divide en varias partes fundamentales, las cuales se comentan a continuación.

INICIALIZACIÓN Y CONFIGURACIÓN

A priori estos paneles están pensados para ser ejecutados dinámicamente mediante las funciones del VI SERVER (aunque no es estrictamente necesario). Una vez arrancados los paneles:

1. Se lanzan los servicios del panel: dos procesos que se ejecutan en paralelo con la intención de no ser bloqueantes y cumplir una determinada función:
    - **Gestión de errores:** este servicio recibe los errores que se puedan producir y los registra en un fichero de texto. Principalmente está orientado a la depuración del panel.
    - **Comprobación de conexión**: se encarga de comprobar el bit de vida con el servidor de alarmas y de las peticiones de actualización.

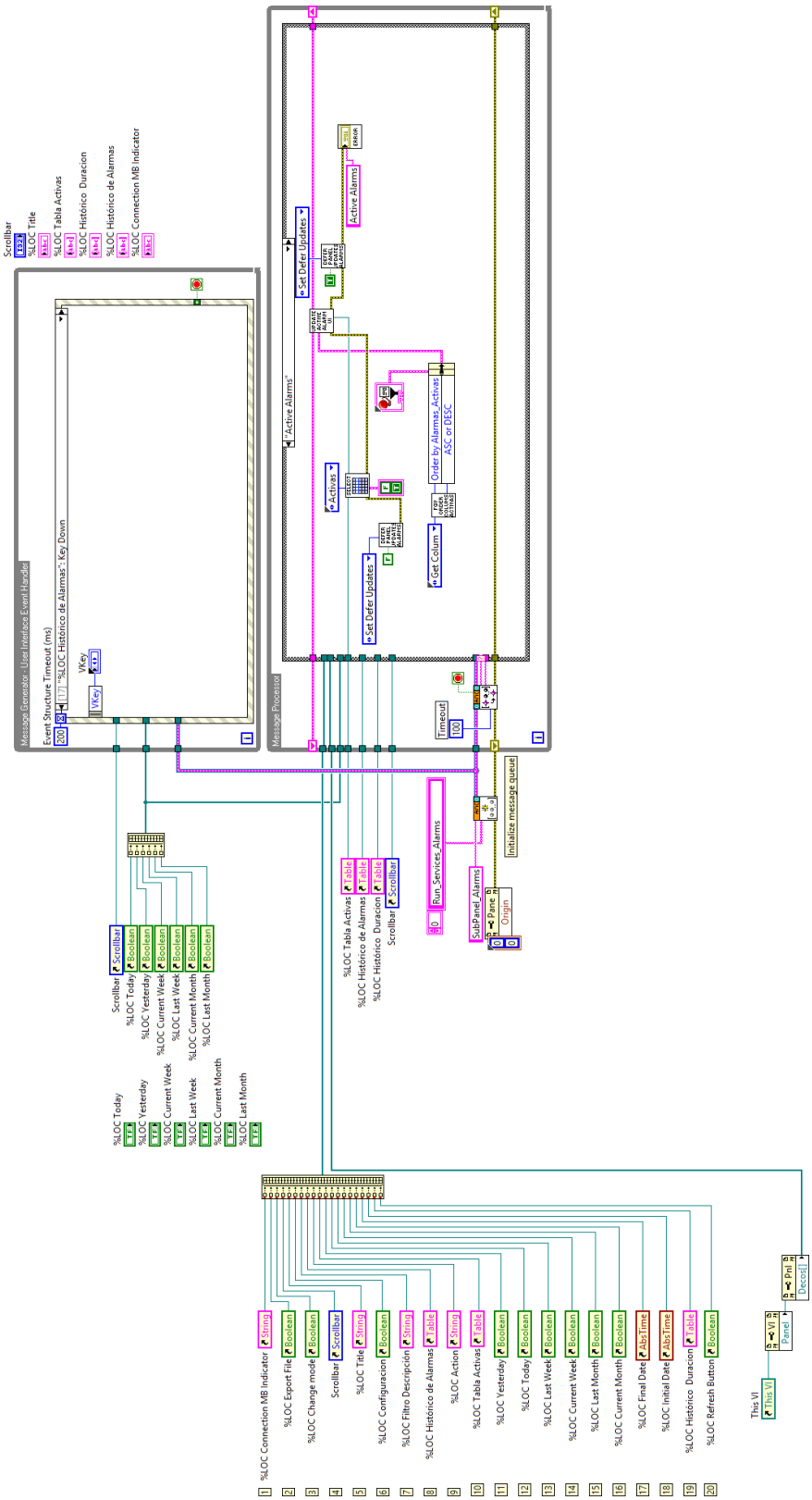

Figura 18: Estructura panel de alarmas y eventos

2. El panel se queda esperando a recibir la configuración del usuario, el cual las envía haciendo uso de las funciones de la paleta. Una vez el panel recibe la información de configuración, establece una secuencia de estados para ponerse en funcionamiento:
    - **Inicialización:** carga el resto de parámetros de configuración de un fichero ".ini" e inicializa las referencias necesarias.
    - **Inicialización de controles:** actualiza los controles de filtro a los valores correspondientes o por defecto.
    - **Recopilación de datos:** consulta a la base de datos la información de Alarmas/Eventos.
    - **Interfaz:** actualiza las tablas de históricos (y el árbol en el caso de eventos).

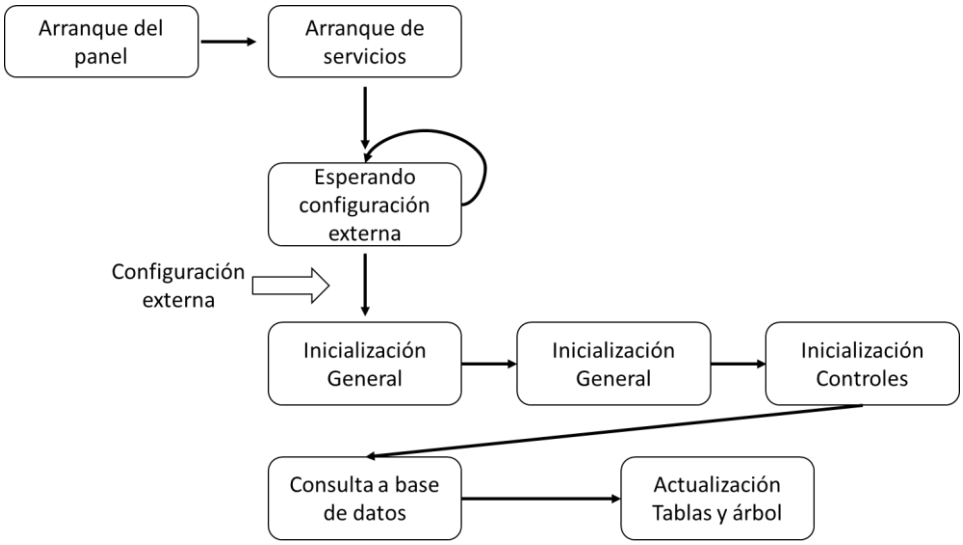

Figura 19: Pseudocódigo: Inicialización

GESTIÓN DE ACTUALIZACIÓN DE INTERFAZ

Todos los controles que actúan como filtros para la actualización de las tablas se registran en la estructura de eventos. De esta manera, el panel no necesita estar consumiendo CPU cuando no se está modificando nada, ya que solo se actualizará cuando se produzca un cambio en alguno de los controles.

Otra forma de que se produzca la actualización de la interfaz es que el servicio que comunica con el servidor reciba la orden de actualizar (vía local o TCP) porque se ha producido un cambio en la base de datos.

Cada vez que se da cualquiera de las dos opciones anteriores se encolan una secuencia de estados para la actualización de las tablas:

- Lectura de controles.
- Construcción de sentencia.
- Consulta a la base de datos.
- Actualización de tablas y árbol.

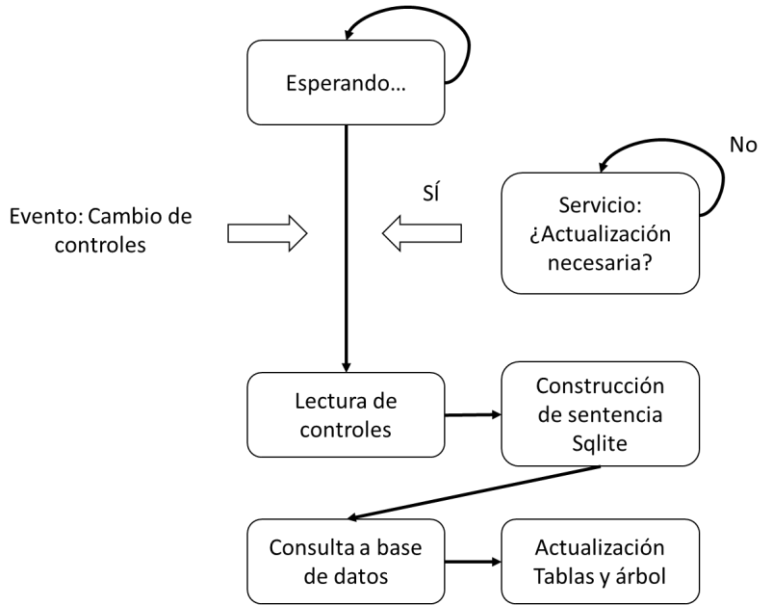

Figura 20: Pseudocódigo: gestión de filtros

GESTIÓN DE OFFSET DE TABLAS

Las tablas de históricos deben de estar preparadas para grandes cantidades de datos. En ocasiones las vistas de la base de datos pueden llegar a tener una gran cantidad de registros. En caso de realizar una consulta a la base de datos de varios miles de filas, la actualización de la interfaz gráfica puede llegar a ser realmente lenta, ya que entre otras cosas requiere el pintado de las filas. Es por ello que es necesaria una gestión de offset, es decir, actualizar en las tablas únicamente lo que se está visualizando.

Para ello es necesario hacer uso de controles de tipo scrollbar, ya que es fundamental diferenciar y desligar la lectura de datos de la actualización de la interfaz. Por un lado, como bien se ha dicho antes, cada vez que cambia un control se hace una consulta a la base de datos. En ese momento el panel ha de guardar los datos en una variable y evaluar su magnitud. En función del tamaño de los datos leídos, el panel aumenta o disminuye el rango de los controles “scrollbars” de las tablas para hacerlos acordes con el tamaño de los datos. Al registrar el scrollbar en la estructura de eventos, cuando se produce un cambio sobre este, el panel es capaz de actualizar en las tablas únicamente la ventana de datos (de ancho el número de filas de las tablas) sobre el índice de scroll, aumentado increíblemente la fluidez y el rendimiento cuando se está trabajando con grandes cantidades de datos.

GESTIÓN CAMBIO DE INTERFAZ

Esta parte del código no se utiliza durante el funcionamiento normal de la aplicación. De hecho, no se puede acceder a él utilizando los objetos del panel frontal. Esta funcionalidad es usada de manera externa por la herramienta de gestión de paneles del framework (la cual se comenta en el apartado 5.4.5.2) cada vez que se crea un panel por primera vez.

Internamente, el panel utiliza todas las referencias de los objetos del panel frontal para modificar su interfaz (mediante propiedades) en función de la información de un fichero de configuración gráfica. La ruta de este fichero es enviada desde la herramienta al panel, de tal manera que este únicamente lee su contenido y modifica su interfaz de manera acorde cuando la herramienta se lo ordene.

5.4.4 Paleta de funciones

Este apartado desarrolla a un nivel general la estructura de la paleta y cada una de sus partes. No se entra a definir y explicar cada función en particular, ya que eso se realiza en la “Guía de Usuario”.

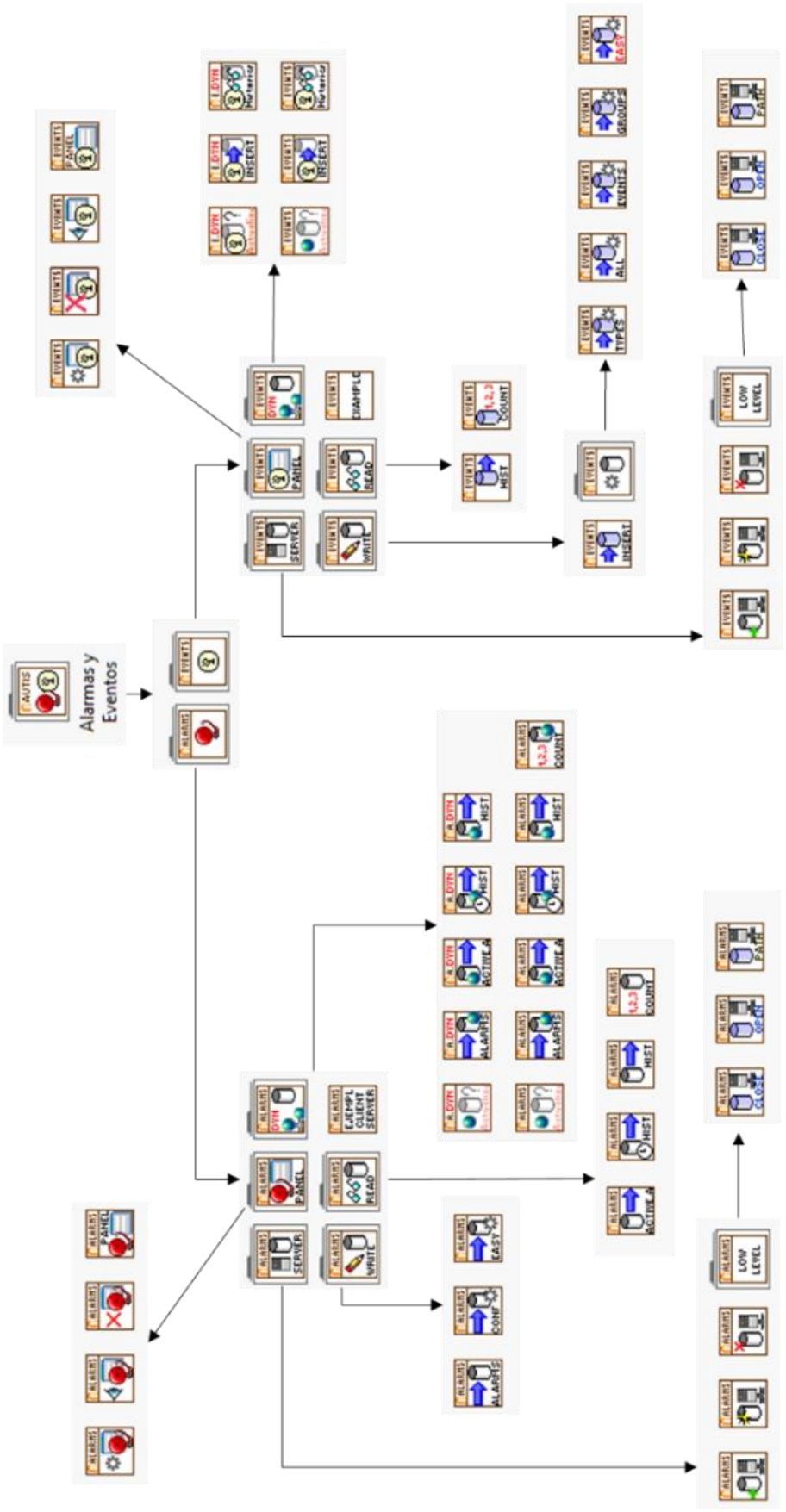

Figura 21: estructura de la paleta

La paleta del toolkit se divide completamente en dos partes, la de alarmas y la de eventos. Ambas paletas se estructuran de la misma forma y constan de los mismos grupos de funciones. A continuación, se procede a explicar cada uno de estos grupos.

PALETA DE FUNCIONES: SERVIDOR

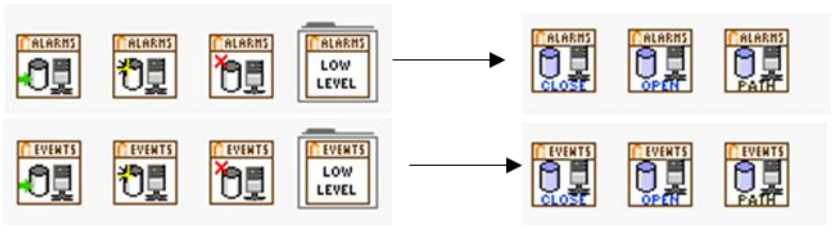

Figura 22: Paleta (Servidor)

Estos VIs se utilizan para comandar las acciones del servidor de Alarmas/Eventos y enviarles información. Principalmente, utilizan las referencias de las colas para enviar los datos necesarios al servidor una vez que este está ejecutándose.

PALETA DE FUNCIONES: PANELES

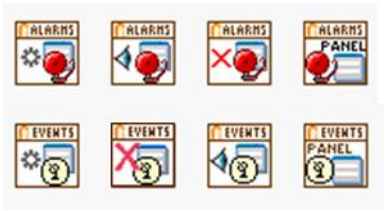

Figura 23: Paleta (Paneles)

Funcionan de manera similar a las anteriores, pero en este caso utilizan la referencia del panel de Alarmas/Eventos.

PALETA DE FUNCIONES: ESCRITURA

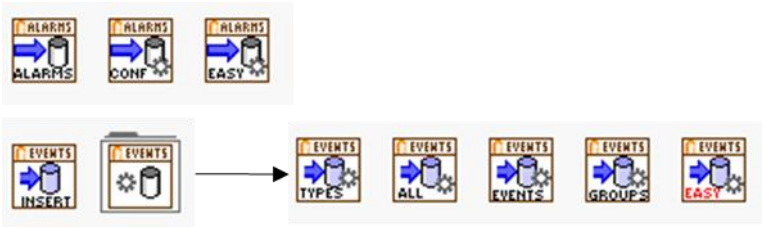

Figura 24: Paleta (Escritura)

En este caso se pueden distinguir dos tipos de funciones:

- **Funciones Online:** Funciones de inserción de Alarmas/Eventos pensadas para ser utilizadas durante la ejecución normal de la aplicación. Estas funciones presentan las mismas estructuras que las funciones de la paleta del servidor o paneles. Utilizan la referencia del servidor (por tanto, necesitan que el servidor esté arrancado) para comunicar la información, y es el servidor el que se encarga de gestionar esa información.
- **Funciones Offline (o de inicialización):** Son todas las referentes a la configuración de la base de datos. Estas funciones sí que desempeñan su función de manera independiente. Los pasos que realizan son los siguientes:
    - Recogida de información de las entradas.
    - Análisis de esa información: gestión de datos duplicados, vacíos, etc.
    - Creación de Sentencias.
    - Inserción en la base de datos.

  Debido a que estas funciones se ejecutan de manera independiente a los servidores de alarmas y eventos, es necesario gestionar la conexión con la base de datos (cuando se ejecute de manera offline al servidor). Para ello es necesario utilizar las funciones del Toolkit SQLite Autis API para abrir y cerrar la conexión.

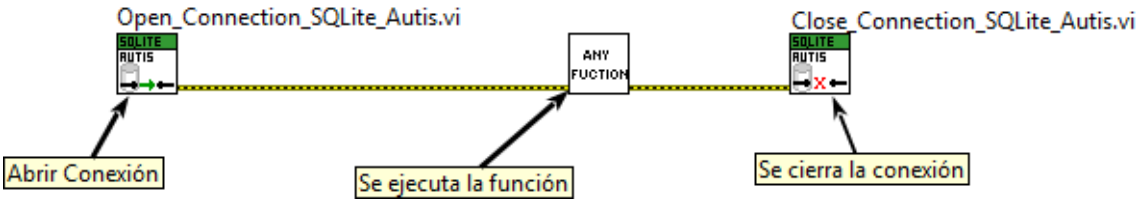

Figura 25: Ejemplo ejecución offline

PALETA DE FUNCIONES: LECTURA

Figura 26: Paleta (Escritura)

Estas funciones se asemejan a las funciones de escritura (offline), es decir, se pueden ejecutar de manera independiente al servidor de eventos y alarmas (siempre que tengan acceso a la base de datos). No obstante, la secuencia de operaciones que realizan es ligeramente diferente:

- Recogida de información de las entradas.
- Construcción de sentencias SQLite.
- Lectura de la base de datos.
- Transformación de la información en los tipos de datos correspondientes.

PALETA DE FUNCIONES: VIs Dinámicos (DYN)

Figura 27: Paletas (VIs dinámicos cliente-Servidor)

El toolkit contempla una serie de funciones reentrantes agrupadas en parejas (cliente y servidor). Estas funciones no son más que una réplica de algunas de las funciones de lectura y escritura anteriormente descritas en formato de peticiones TCP. Por tanto, estas funciones deberán de usarse cuando el servidor de Alarmas/Eventos esté en una aplicación distinta de donde se desea realizar la escritura y/o lectura.

Las funciones cliente utilizan las funciones del toolkit "ModBUS TCP" para hacer peticiones al servidor remoto y llamar a las funciones DYN (o servidor), las cuales hacen uso de los VI de lectura/escritura anteriormente descritos para devolver los resultados al cliente. **A nivel de usuario, se utilizarán las funciones "Cliente", ya que las funciones "DYN" se integran en el servidor de alarmas por defecto** (únicamente con abrir el servidor se están cargando estas funciones en memoria).

5.4.5 Framework: herramientas

El Framework consta de una serie de herramientas que facilitan a los usuarios el desarrollo de los módulos de alarmas y eventos en los distintos proyectos. El uso detallado de estas queda patente en la "Guía de Usuario", por tanto, el actual apartado se centrará en explicar la estructura interna de las aplicaciones, agrupando de nuevo la explicación para las aplicaciones de alarmas y eventos y puntualizando cuando sea necesario. No obstante, para un mejor entendimiento, se recomienda leer previamente a este apartado el punto 4 de la "Guía de Usuario".

Cabe destacar que las herramientas que aquí se comentan han sido integradas en el entorno de desarrollo de LabVIEW y no en la paleta de funciones. Esto se debe a que LabVIEW posee un menú destinado a incluir este tipo de desarrollos, resultando en un acceso notablemente más práctico e intuitivo.

5.4.5.1 Asistente de configuración Gráfica

Se ha diseñado esta aplicación para proveer al framework de una herramienta para definir nuevas interfaces que puedan ser reutilizadas para futuros proyectos. De esta manera, cuando se vaya a realizar un nuevo diseño, siempre se podrá partir de uno anterior, evitando así una cantidad considerable de esfuerzo e inspiración.

Su funcionamiento se basa en combinar las herramientas de edición del propio entorno de desarrollo de LabVIEW con las funciones del VI Server y con las propiedades y métodos de los objetos del panel frontal. A continuación, se comentan sus partes fundamentales:

ARRANQUE

Cada vez que se llama a esta herramienta se crea una copia del VI original. La justificación es que el usuario de esta herramienta va a poder editar este VI sin estar en ejecución, por tanto se abre la posibilidad de modificar código, eliminar elementos, etc.

La manera de arrancar esta herramienta es utilizando las funciones del VI Server para realizar una copia limpia de la plantilla original (dejando esta última a salvo de cualquier mal uso) y lanzando la copia dinámicamente y mostrando su panel frontal.

ESTRUCTURA: FUNCIONAMIENTO

La estructura de esta aplicación es un Productor-Consumidor basado en colas y con máquina de estados. Estos estados son ejecutados cada vez que salta un evento asociado al bucle consumido. Estos eventos son cambios en los controles que se producen durante la ejecución de la herramienta. Entre los estados se distinguen 3 tipos:

- **Estado de cambio de visibilidad de los objetos del panel frontal:** accediendo a las propiedades de los objetos manipulados.
- **Estado de carga de interfaz:** este estado carga de un fichero de texto (ubicado en el directorio raíz del toolkit) toda la información de la interfaz gráfica de otro panel de Alarmas/Eventos para posteriormente aplicarlo haciendo uso de las propiedades de los objetos del panel frontal.
- **Estado de guardado de interfaz:** este estado realiza la operación contraria al anterior. Hace uso de todas las propiedades necesarias para extraer la información de los objetos del panel frontal y así guardarla en un fichero de texto.

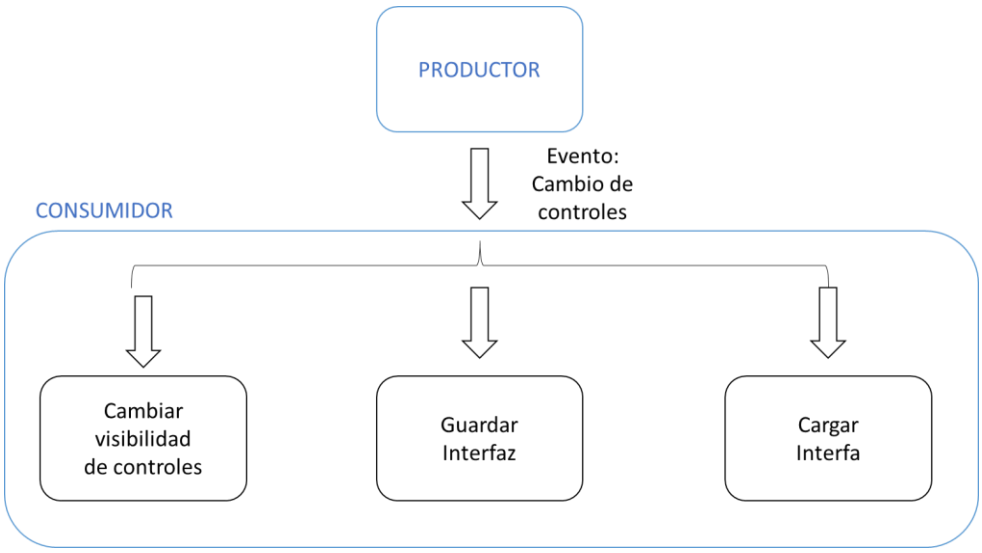

Figura 28: Estados (Pseudocódigo)

EDICIÓN

Como se ha comentado anteriormente, los usuarios pueden acceder a esta herramienta sin estar en ejecución. Esto es debido a que la manera más fácil y cómoda para un desarrollador de editar una interfaz gráfica es usar el propio entorno de LabVIEW (ya que es lo que está acostumbrado a utilizar).

5.4.5.2 Ventana de gestión de paneles

Esta herramienta está orientada a asistir a los usuarios a la hora de crear nuevos paneles de alarmas y eventos. Estos asistentes permiten la selección de distintas interfaces ya predefinidas mediante el "Asistente de configuración Gráfica". Además, permiten el mantenimiento y gestión de los paneles ya existentes, bien sea en un mismo proyecto o en varios proyectos a la vez.

ESTRUCTURA

Está basada en una estructura Productor-Consumidor con máquina de estados. Todos los controles se registran en la estructura de eventos del productor para llamar consecuentemente los estados del consumidor. Existen diversos estados agrupados en diversas funcionalidades:

- **Gestión de proyectos:** mediante los controles adecuados, los usuarios pueden dar de alta nuevos proyectos o eliminar proyectos antiguos. Para ello, la herramienta accede a estos estados para guardar y cargar la información de los proyectos en unos ficheros de configuración que se almacenan en el directorio raíz del toolkit.
- **Creación de paneles**: estos estados se llaman cuando se requiere guardar o pre visualizar un panel de Alarmas/Eventos con una determinada interfaz. Para ello se hace uso de las funciones del VI Server:
    - **Guardado:** se modifica la interfaz del panel lanzándolo dinámicamente y posteriormente se guarda.
    - **Pre visualizar:** se modifica la interfaz del panel lanzándolo dinámicamente y se muestra su panel frontal.

  Cada vez que se crea un panel, se genera un fichero de configuración con toda la información de ese panel, el cual se guarda en el directorio raíz del proyecto que se está manipulando.
- **Configuración de paneles:** este estado únicamente accede a los ficheros de configuración de paneles anteriormente comentados y lanza otra herramienta para su modificación, la cual se comenta a continuación.

5.4.5.3 Ventana de configuración de paneles

Esta herramienta se ha desarrollado para configurar los distintos paneles creados y así modificar algunas opciones de su comportamiento. Principalmente está planteada para ser utilizada desde el framework (desde la ventana de gestión de paneles), no obstante, también puede ser utilizada directamente desde los paneles de alarmas y eventos en caso de que los usuarios lo requieran. A continuación, se muestra una imagen de estas herramientas.

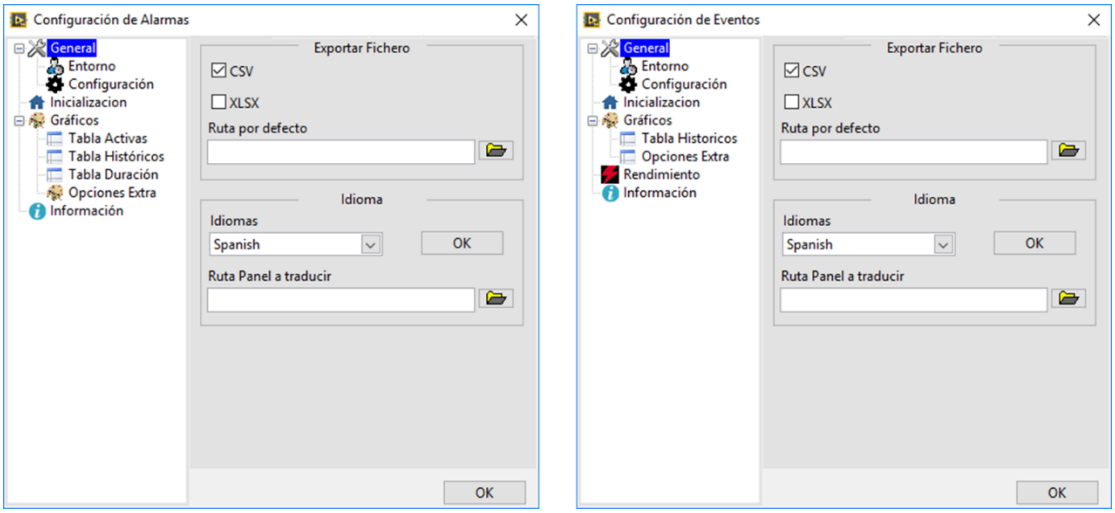

Figura 29: Ventanas de configuración de alarmas y eventos

ESTRUCTURA

Está basada en una estructura Productor-Consumidor donde todos los controles se registran en una estructura de eventos del bucle productor. El funcionamiento de esta herramienta se puede dividir en 4 partes:

- **Arranque:** debido a que esta herramienta siempre se abre desde una aplicación externa, siempre se le pasa el nombre del panel asociado a la configuración que se está modificando. De esta manera, la herramienta abre el archivo de configuración actualizando todos los controles con los últimos valores modificados.

- **Modificación de parámetros**: cada vez que se modifica un control, se dispara un evento asociado al mismo y se actualiza una zona de memoria con la nueva información. Algunos de los parámetros que se pueden modificar son:
    - Opciones de rendimiento.
    - Opciones de inicialización.
    - Opciones de actualización de la interfaz.
    - Opciones para la exportación de informes de históricos.
    - Campos mostrados en las tablas de históricos.
- **Traducción:** este módulo utiliza la funciones del VI server para buscar la referencia de todos los controles del panel de Alarmas/Eventos al que apunta. Posteriormente carga un fichero de idioma con toda la información de la traducción. Este fichero se construye con la información de los nombres de los controles que posteriormente se van a buscar, de esta manera se puede relacionar cada campo de la traducción con el control a traducir.
- **Cierre:** se guarda en el fichero de configuración del panel toda la información modificada guardada en memoria. Así la próxima vez que se abra el panel de Alarmas/Eventos cargará la nueva configuración, actualizando su comportamiento en consecuencia.

5.4.5.4 Asistente de configuración de base de datos.

Se trata de una aplicación autodocumentada e integrada en el entorno de desarrollo de LabVIEW para ayudar a los nuevos usuarios a configurar las bases de datos de la manera correcta.

Esta herramienta solo se ha desarrollado para la parte de alarmas. La razón principal es que normalmente la base de datos de alarmas se plantea en base a un listado de alarmas definidas por un PLC y no por el desarrollador en LabVIEW. Al tener las alarmas definidas, resulta cómodo utilizar esta aplicación para generar la base de datos. Sin embargo, la base de datos de eventos es definida por el usuario a lo largo del desarrollo del software del proyecto donde se está usando el toolkit y es muy susceptible de sufrir cambios constantes en la configuración, es por ello que en este caso resulta más conveniente y práctico utilizar las funciones de configuración de la paleta.

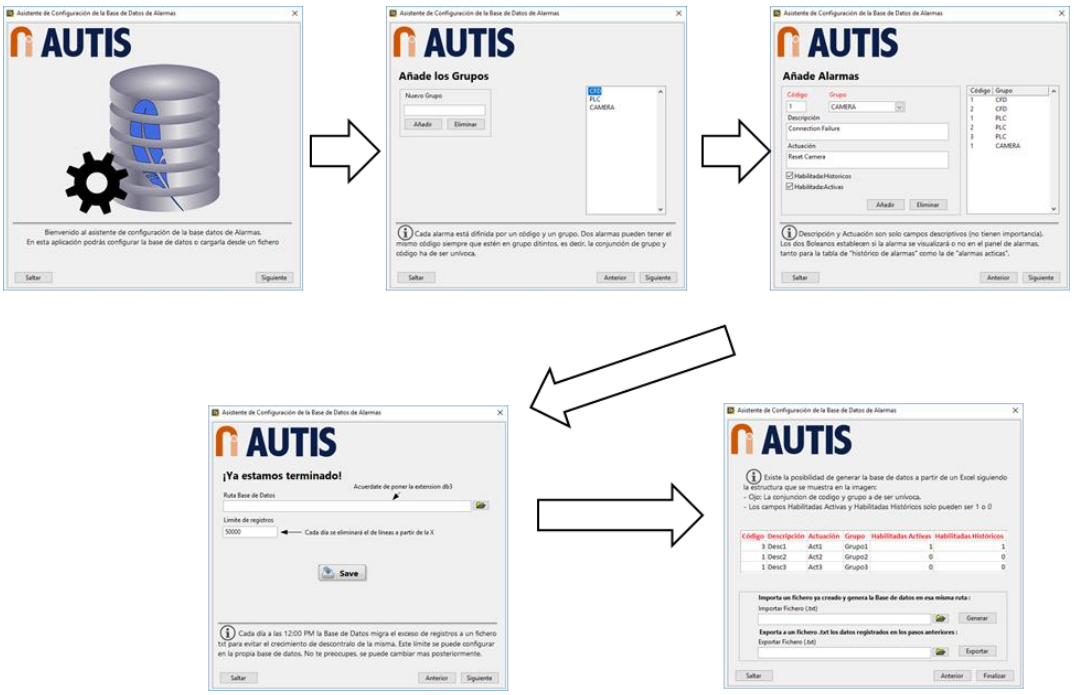

Figura 30: Interfaz del asistente de configuración de la base de datos de alarmas

ESTRUCTURA

Está basada en una estructura Productor-Consumidor donde todos los controles se registran en una estructura de eventos del bucle productor. El funcionamiento de esta herramienta se puede dividir en 4 partes:

- **Recopilación de datos:** mediante una serie de ventanas consecutivas, la aplicación va recaudando todos los datos necesarios almacenándolos en memoria. En cada una de estas ventanas se recoge una parte concreta de los datos activando eventos para la lectura de controles y encolando estados para el registro de datos.
- **Generación de la Base de datos:** una vez recogida toda la información, la aplicación genera todas las sentencias necesarias para generar la base de datos con toda la configuración necesaria.
- **Comunicación con Fichero de texto:** este módulo permite exportar e importar la información recogida a un fichero de texto con la finalidad de almacenar en disco una copia de seguridad de los datos y poder modificarla desde cualquier programa de edición de hojas de cálculo de una manera cómoda y sencilla.

6. INTEGRACIÓN DE LA APLICACIÓN AXLE WELDING

En este apartado se explica el modo en el que se ha utilizado el toolkit de alarmas y eventos en el proyecto Axle Welding. Cabe destacar que **el alcance de este apartado ha sido la integración del módulo de alarmas y eventos en el programa para su posterior utilización y la verificación de su correcto funcionamiento**. Esto quiere decir que no ha sido responsabilidad de este proyecto:

- **La configuración de las bases de datos**: Cada alarma y evento están relacionados con información intrínseca del proyecto, el cual conocen los desarrolladores del mismo. Por tanto, son ellos los que deciden qué eventos y alarmas son relevantes para el proyecto.
- **El uso reiterado de funciones de inserción y lectura en el programa:** Al fin y al cabo, esta es una herramienta que se ha creado para los desarrolladores de software y son ellos los que la usan. Por tanto, es responsabilidad de los programadores el decidir los puntos de programa en los que necesitan realizar la escritura y lectura de datos.

6.1 Introducción: Estructura de la aplicación

El sistema computacional de Axle Welding consta de 3 partes fundamentales:

- **Instancias de adquisición**: 8 aplicaciones idénticas (con configuraciones diferentes), una por sensor, para realizar la adquisición y guardar todas las imágenes en disco.
- **Controller o kernel:** es el núcleo del procesado del proyecto. Esta aplicación recoge las imágenes de adquisición y las procesa para determinar si las soldaduras están en buen estado o no. Además, comanda la aplicación HMI en cada inspección para actualizar la interfaz.
- **Aplicación HMI:** es la cara del sistema computacional. No realiza ninguna función vital para el proceso productivo, sin embargo, ella es la que presenta todos los resultados y notifica de las anomalías.

6.2 Programación de alarmas y eventos en: Axle Welding

A la hora de realizar la programación de los módulos se han de tomar dos decisiones fundamentales:

1. Ubicación del servidor de alarmas y el servidor de eventos

En el caso de este proyecto, lo lógico es instalar estos dos subprocesos en el núcleo del sistema computacional, es decir, el controller. Para ello, tanto en la inicialización como en el cierre de esta aplicación se utilizan las funciones de la paleta del servidor para abrir y cerrar tanto el servidor de alarmas como el de eventos.

Una vez montada la estructura de la apertura y el cierre de los servidores de alarmas y eventos, se procede a la inserción de alarmas y eventos mediante las funciones de escritura. En este aspecto, la decisión de dónde y cuándo usar estas funciones es en parte decisión del programador del proyecto, no obstante, se ha orientado su uso a un nivel general:

- **Inserción de alarmas:** en este proyecto las alarmas están definidas con un conjunto de señales que se leen del PLC del sistema. La lectura de estas señales del PLC se realiza en un proceso interno de la aplicación controller, por tanto, los programadores únicamente han de usar las funciones de inserción locales, ya que el servidor y la inserción de alarmas forman parte de la misma aplicación.
- **Inserción de eventos:** este caso es un tanto distinto debido a que parte del proceso computacional transcurre en las instancias de adquisición y parte en el procesado del controller. Es por ello que se han definido eventos asociados a cada una de estas aplicaciones. Por tanto, los eventos originados en el procesado han de hacer uso de las funciones de escritura locales de la paleta, mientras que los eventos producidos en las instancias de adquisición, deberán utilizar las funciones de escritura dinámicas para realizar las inserciones vía TCP.

A continuación, se muestra una imagen con el diagrama estructural del uso de los módulos de alarmas y eventos en el proyecto de Axle Welding.

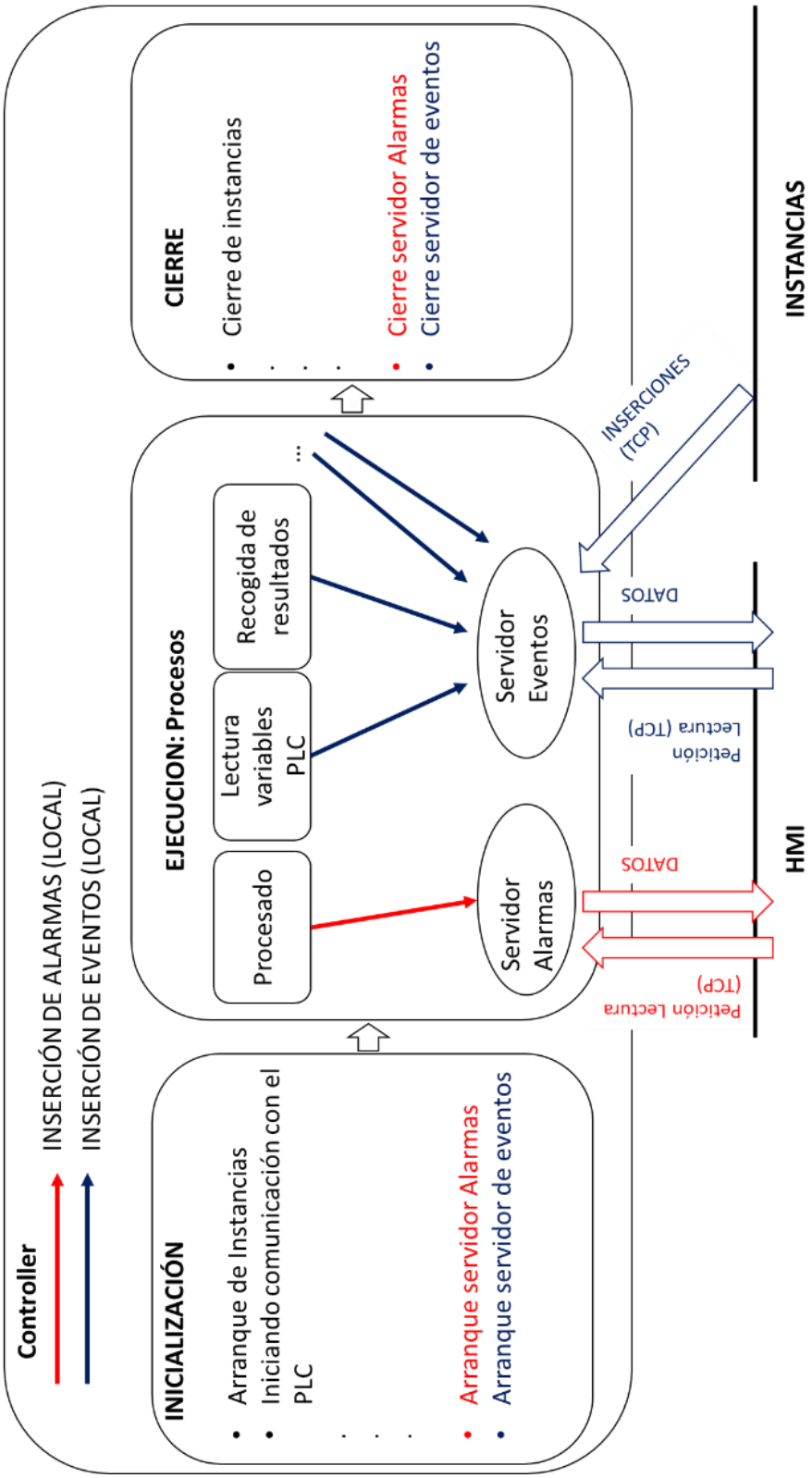

Figura 31: Integración de servidores en la aplicación controller de Axle Welding

2. Ubicación de los paneles de visualización de alarmas y eventos.

En este caso la decisión ha sido sencilla, ya que el proyecto plantea la aplicación HMI para la explotación de todos los datos generados en el sistema. La integración de los paneles en este caso es directa y consta de 4 pasos:

- **Dar de alta los paneles:** utilizando las herramientas del framework se crean un par de paneles (Alarmas y Eventos) y se configuran para que el comportamiento sea el deseado.
- **Arranque del panel:** lanzamiento dinámico de los VIs creados mediante las funciones del VI server en el proceso de inicialización del HMI.
- **Parametrización del HMI:** mediante la paleta de comunicación con el panel, configurando el panel en remoto para que toda la lectura se realice mediante peticiones TCP (de manera distribuida).
- **Cierre del panel:** durante el procedimiento de cierre de la aplicación utilizando de nuevo las funciones de comunicación con el panel.

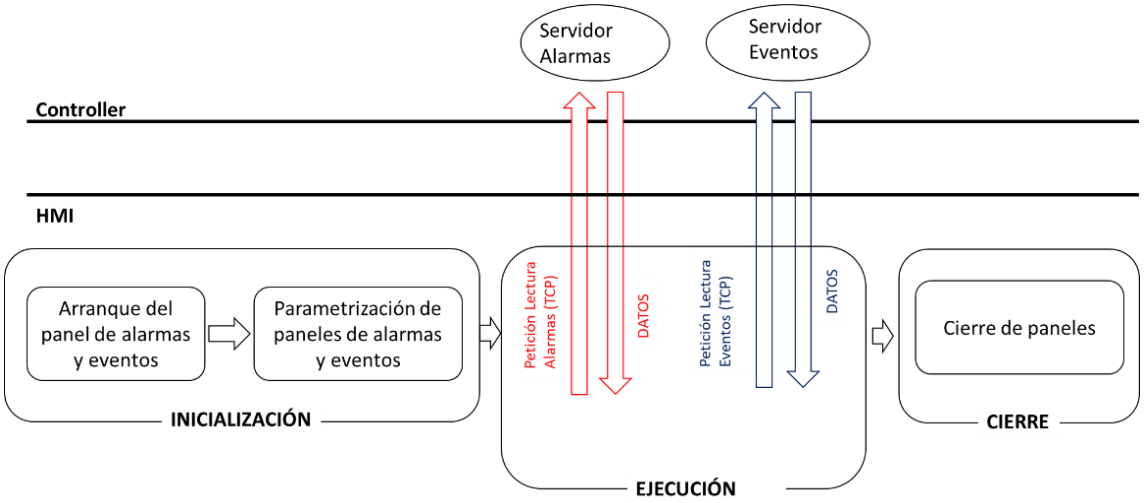

Figura 32: Estructura de la integración de paneles en la aplicación HMI de Axle Welding

Una vez montada la estructura, el propio panel funciona de manera autónoma sin que los desarrolladores tengan que preocuparse de programar nada, únicamente de cambiar la configuración desde las herramientas del framework, en caso de que se desee realizar alguna modificación. A continuación, se muestra la interfaz de la aplicación con los paneles de visualización de alarmas y eventos.

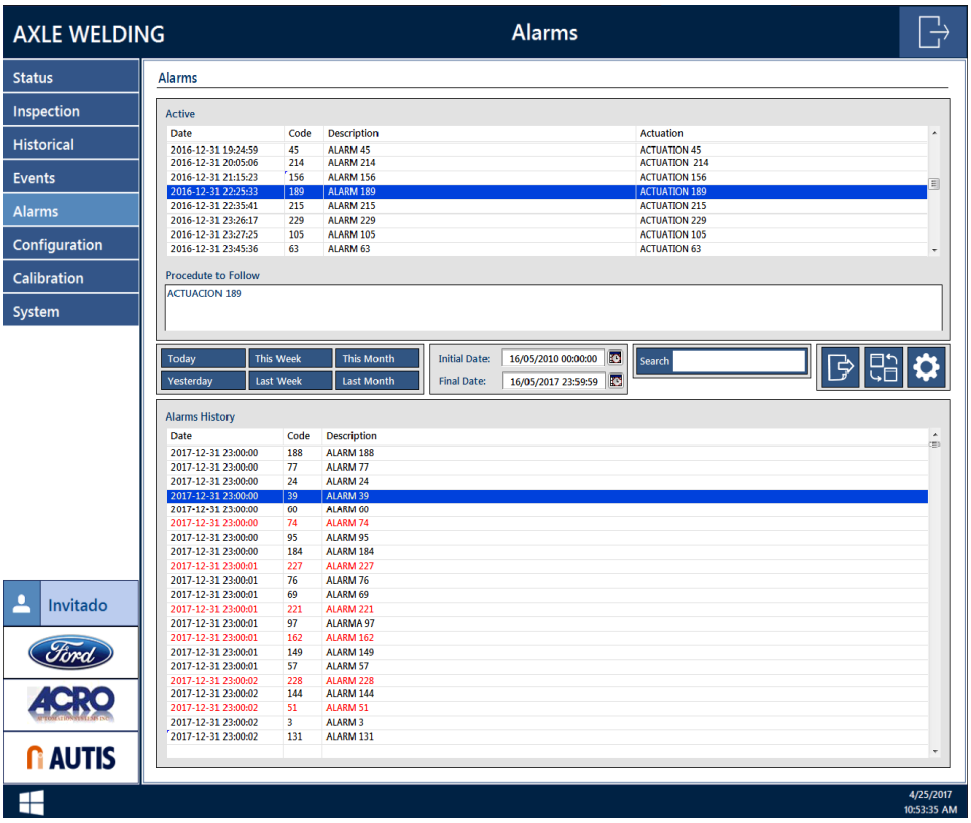

Figura 33: Panel de alarmas (Axle Welding)

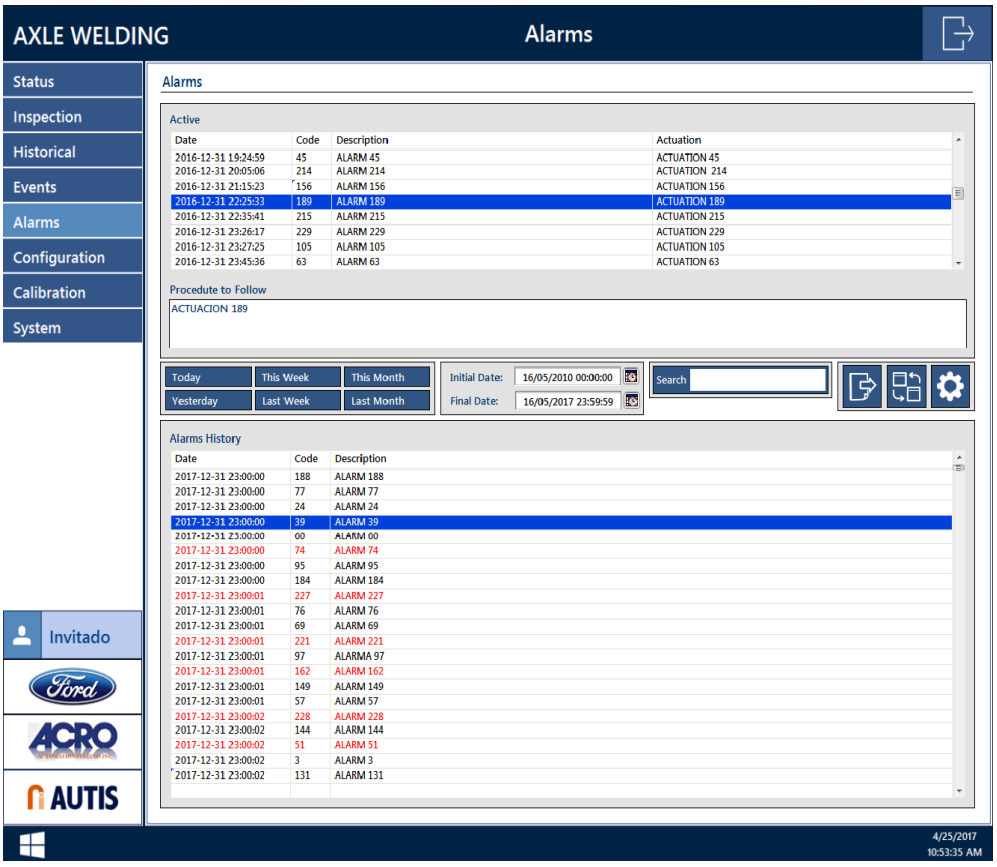

Figura 34: Panel de eventos (Axle Welding)

7. CONCLUSIONES

Finalmente, tras mucho tiempo de desarrollo del toolkit y su validación final en el proyecto Axle Welding, se puede concluir que se han cumplido los siguientes objetivos:

- Se ha desarrollado una paleta de funciones completa y funcional como API entre los procesos y los desarrolladores.
- Se han diseñado dos bases de datos a medida capaces de albergar toda la información necesaria para los diversos proyectos de la empresa.
- Se ha dotado al toolkit de unos servidores capaces de otorgar robustez y potencia al desarrollo de los módulos de alarmas y eventos.
- Se han desarrollado dos paneles de visualización de alarmas y eventos capaces de explotar al 100% la información de la base de datos, además de ser completamente configurables y flexibles con el fin de adaptarse a cualquier proyecto.
- Gracias a la arquitectura de los paneles y los servidores se ha conseguido desarrollar estructuras distribuidas donde los datos y la explotación de los mismos están separados.
- Se ha conseguido desarrollar diversas herramientas con la finalidad de facilitar el trabajo de los desarrolladores:
    - Asistente de creación de base de datos (en el caso de alarmas).
    - Asistente de creación de nuevos paneles: Para ofrecer un punto de partida e inspiración a los diseños gráficos.
    - Ventana de gestión de paneles: Para crear nuevos paneles de configuración y gestionar los viejos en diversos proyectos simultáneamente.
    - Ventana de configuración de paneles: Para modificar el comportamiento de las interfaces a fin de que se adapten lo mejor posible a cada proyecto.

En definitiva, se ha conseguido crear un marco de trabajo completamente integrado capaz de conseguir desarrollos de manera ágil, fácil y eficaz para reducir tiempos y ahorrar costes. Además, se ha conseguido su aplicación de manera satisfactoria a un proyecto de visión artificial de gran magnitud como es Axle Welding.

8. BIBLIOGRAFÍA

[1] *Advanced Architectures in LabVIEW.* (2012). National Instruments.

[2] *LabVIEW Connectivity Course Manual.* (2012). National Instruments.

[3] *LabVIEW Core 1 Course Manual.* (2012). National Instruments.

[4] *LabVIEW Core 2 Course Manual.* (2012). National Instruments.

[5] *LabVIEW Core 3 Couerse Manual.* (2012). National Instruments.

[6] *LabVIEW Performance Course Manual.* (2012). National Instruments.

[7] Puig, V. (2017, Junio 15). Axle Welding: Training. *Axle Welding: Training*. Gandia: Autis Ingenieros S.L.U.

[8] Wikimedia, F. (2017, Abril 22). *LabVIEW.* Retrieved from Wikipedia: https://es.wikipedia.org/wiki/LabVIEW

[9] Wikimedia, F. (2017, Septiembre 6). *SQL.* Retrieved from Wikipedia: https://es.wikipedia.org/wiki/SQL

[10] Wikimedia, F. (2017, Septiembre 2). *SQLite.* Retrieved from Wikipedia: https://es.wikipedia.org/wiki/SQLite

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

Desarrollo de una librería en LabVIEW para la implementación de sistemas de gestión distribuidos de alarmas y eventos. Aplicación a un proyecto de visión artificial para la detección de defectos en soldaduras de ejes destinados al sector de la automoción.

Documento Nº 2: Manual de usuario

AUTOR: Dotor Alonso, Luis

TUTOR: Herrero Durá, Juan Manuel

Curso Académico: 2016-2017

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

En el presente documento se pretende describir y explicar el funcionamiento del toolkit. Esta guía está pensada como un manual explicativo para que un programador experimentado en LabVIEW pueda entenderlo sin problemas. Por tanto, se presupone que el lector conoce los conceptos de:

- VI.
- Subpanel.
- VI Server: Lanzamiento dinámico de VI.
- Pane Control.

2. PALETA DE FUNCIONES

En este apartado se procede a explicar cada una de las funciones presentes en el toolkit. Debido a que existen muchas similitudes, se procederá a explicar ambas conjuntamente, matizando cuando sea necesario.

Para acceder a los VIs, el usuario deberá abrir la paleta de funciones del "Block diagrama": Functions → Autis Ingenieros S.L.U. → Alarmas y Eventos.

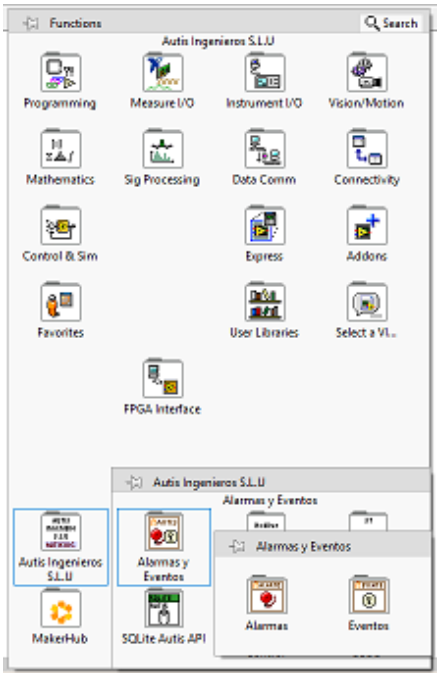

Figura 1: Acceso a la paleta de funciones

Una vez dentro la paleta se divide en 2 partes principales, la de alarmas y la de eventos. A continuación, se muestra una imagen de la estructura de la paleta.

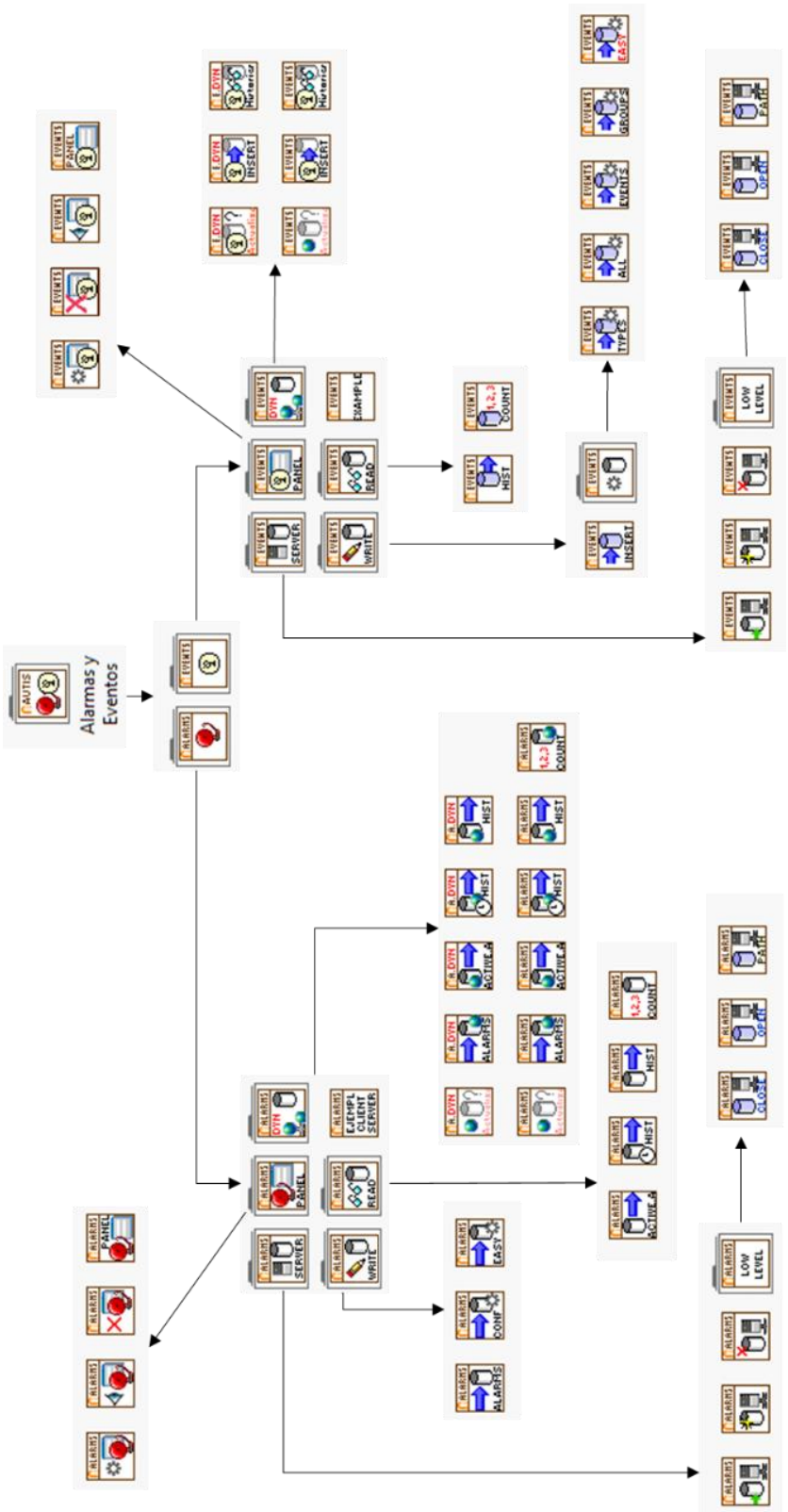

Figura 2: Estructura de la paleta de funciones

2.1 Paleta: Servidor

El principal uso de esta paleta es comandar las acciones del servidor de alarmas y eventos.

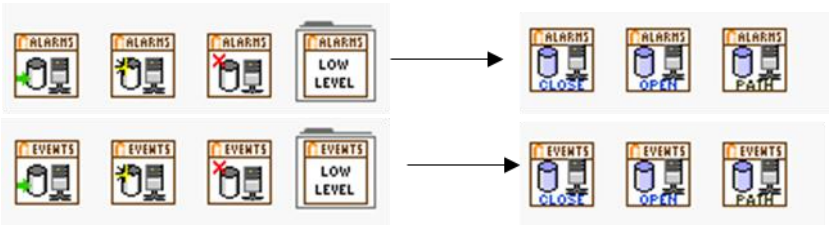

Figura 3: Paleta del Servidor

Open_SQLite_Server_Alarms/Events.vi:

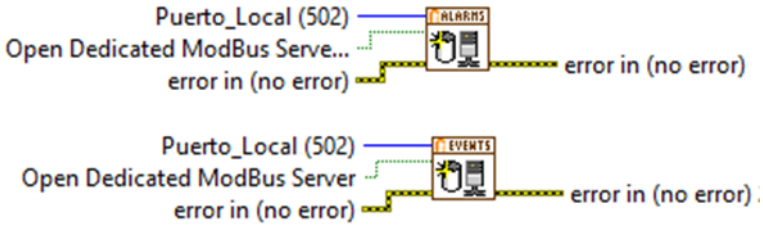

Figura 4: Open_SQLite_Server_Alarms/Events

Esta función abre dinámicamente el servidor de Alarmas/Eventos. Siempre que la entrada "Open Dedicated Modbus Server" esté a true, el servidor de Alarmas/Eventos abrirá un Servidor TCP en el puerto local especificado en el terminal "Puerto local (502)". Normalmente, se usará en la inicialización de la aplicación.

Load_BBDD_in_SQLite_Server_Alarms/Events.vi

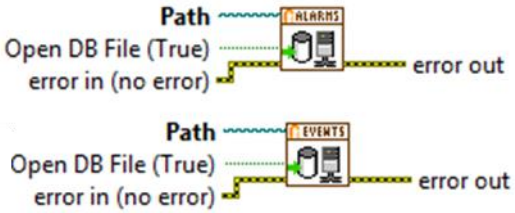

Figura 5: Load_BBDD_in_SQLite_Server_Alarms/Events

Este VI no es más que una conjunción de los VIs:

- "Set_Path_DDBB_in_SQLite_Server_Alarms".
- "Open_DDBB_in_SQLite_Server_Alarms/Events".

Se utiliza para indicarle al servidor que establezca conexión con la base de datos especificada en la entrada “Path”. En caso de que la entrada “Open DB File” esté a false, se establece la ruta, pero no la conexión.

Stop_SQLite_Server_Alarms/Events.vi

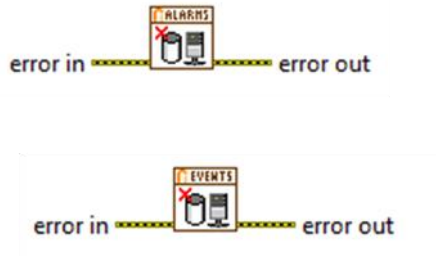

Figura 6: Stop_SQLite_Server_Alarms/Events

Este VI cierra la base de datos y posteriormente cierra la cola del servidor de alarmas. Normalmente, se utilizará esta función durante el procedimiento de cierre del programa.

Set_Path_DDBB_in_SQLite_Server_Alarms/Events.vi

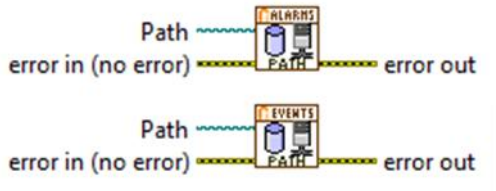

Figura 7: Set_Path_DDBB_in_SQLite_Server_Alarms/Events

Este VI le indica al servidor el directorio de la base de datos que va a ser usada para registrar las Alarmas/Eventos.

Open_DDBB_in_SQLite_Server_Alarms/Events.vi

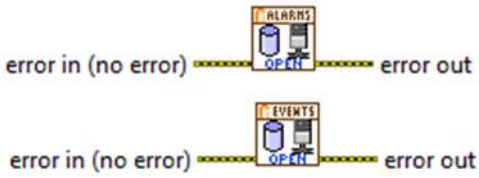

Figura 8: Open_DDBB_in_SQLite_Server_Alarms/Events

Esta función abre (o crea, si no existe) la base de datos especificada en el VI anterior “Set_Path_DDBB_in_SQLite_Server_Alarms/Events.vi”.

Close_DDBB_in_SQLite_Server_Alarms/Events.vi

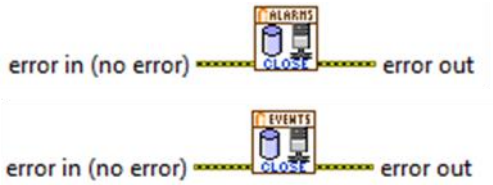

Figura 9: Close_DDBB_in_SQLite_Server_Alarms/Events

Este VI cierra la base de datos del servidor de alarmas pero no cierra el servidor. Se utilizará esta función cuando se quiera abrir una base de datos distinta a lo largo de la ejecución del programa.

Normalmente no se usan estos últimos 3 VIs, no obstante, se incluyen por si se quiere programar alguna funcionalidad adicional. En una situación normal utilizaremos:

- **"Open_SQLite_Server_Alarms/Events":** para abrir el servidor.
- **"Load_BBDD_in_SQLite_Server_Alarms/Event":** para configurar la base de datos.
- **"Stop_SQLite_Server_Alarms/Events":** para cerrar el servidor.

2.2 Paleta: Panel

El principal uso de esta paleta es comandar las acciones del panel de alarmas y eventos externamente.

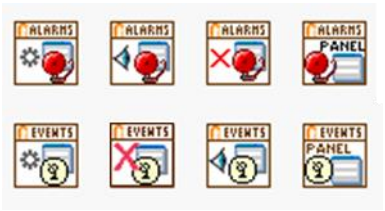

Figura 10: VIs de Paneles

Ninguna de estas funciones abre el panel de alarmas o evento. Esto se ha proyectado así debido a que en cada proyecto puede interesar abrir el panel de una forma u otra. La forma más habitual es lanzarlo dinámicamente:

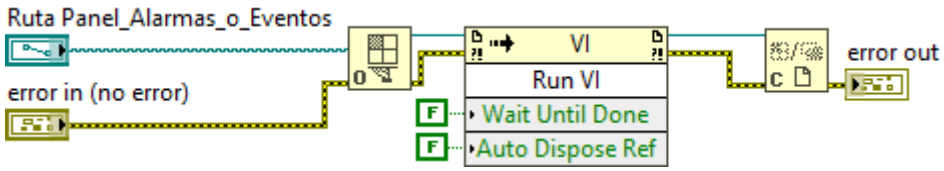

Figura 11: Ejemplo lanzamiento dinámico Vis

Config_SubPanel_Alarms/Events.vi

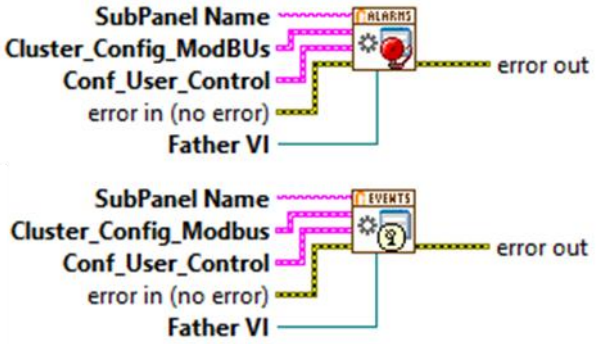

Figura 12: Config_SubPanel_Alarms/Events

Esta función se utiliza para parametrizar el panel de Alarmas/Eventos. Posee 4 entradas de configuración:

- **Subpanel Name:** nombre que sirve para especificar al panel el nombre del fichero configuración el cual define parte del comportamiento del mismo. Este fichero se crea cuando el usuario da de alta un panel de alarmas o eventos. En caso de querer saber cómo crear estos fichero, leer el apartado 4.3 Ventana de gestión de Paneles.
- **Cluster_Config_Mobbus:** cluster de configuración para especificar al panel la configuración del servidor TCP:
  - **T: Local SV F: Remote SV (F):** entrada para especificar si el servidor de Alarmas/Eventos está en la misma aplicación que el panel (local), o en otra aplicación (Remoto).
  - **IP:** dirección IP del dispositivo donde se ubica la aplicación del servidor de Alarmas/Eventos.
  - **Port**: puerto TCP del servidor de Alarmas/Eventos
  - **ID_Connect (Número + texto):** identificador de la conexión. Esta información debe ser unívoca para todas las conexiones simultáneas que se realicen al servidor. Por tanto, es responsabilidad del usuario utilizar unos valores que no vayan a ser utilizados por otras aplicaciones o procesos.
- **Conf_User_Control:** cluster de configuración que solo tiene sentido ingresar en caso de incluir el botón de configuración en la interfaz del panel de Alarmas/Eventos. Posee 2 entradas:
  - **Path UsersControl:** ruta de la base de datos de control de usuarios.
  - **User_Type:** numérico para indicar el grado de acceso a partir del cual se debe de poder acceder al árbol de configuración.
- **Father VI**: referencia del VI principal de la aplicación que utiliza el panel de Alarmas/Eventos.

Stop_SubPanel_Alarms/Events.vi

Figura 13: Stop_SubPanel_Alarms/Events

Esta función encola un estado de salida en el panel de Alarmas/Eventos. Normalmente, se usará este VI en el proceso de cierre de la aplicación.

Set_or_Unset_Focus.vi

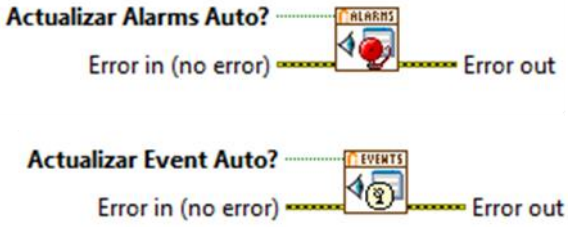

Figura 14: Set_or_Unset_Focus

Esta función permite programáticamente activar/desactivar la actualización automática del panel. Se utiliza principalmente en aplicaciones donde es necesario gestionar el foco de la interfaz debido a que generalmente son aplicaciones que constan de una gran cantidad de código y en las que es necesario poner los subprocesos en un estado de reposo cuando no se estén usando.

Panel de Alarmas/Eventos

Figura 15: paneles de alarmas/Eventos

Estas funciones son los propios paneles de alarmas y eventos. Se incluyen en la paleta pero no deben de ser utilizadas por los usuarios, ya que son las plantillas que posteriormente se usan para generar otros paneles. La razón de que se incluyan es para hacer alguna consulta esporádica al código por parte del desarrollador del mismo (una razón puramente práctica de acceso al código). Esto se puede hacer debido a que este toolkit es un desarrollo interno de la empresa Autis Ingenieros S.L.U y no un desarrollo comercial, en cuyo caso, no debería ser accesible desde la paleta.

2.3 Paleta: VIs de escritura

En este punto sí que es importante diferenciar entre los VIs de alarmas y eventos.

2.3.1 Alarmas

Figura 16: Write VIs (Alarmas)

Register_Alarms_Locally.vi

Figura 17: Register_Alarms_Locally

Esta función permite insertar un array de alarmas en la base de datos a través del servidor. Cada alarma estará definida por un cluster en el array de entrada, en los cuales se deberá rellenar la siguiente información: instante, código, grupo y estado **(True: activa, false: no activa).** Para que funcione este VI es necesario tener abierto el servidor de alarmas previamente.

Configure_Alarms_Locally.vi

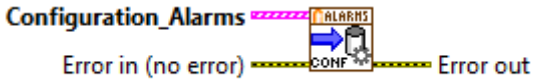

Figura 18: Configure_Alarms_Locally

Esta función inserta un array de configuración de alarmas con cada uno de los campos de la tabla "CONFIGURATION_ALARMAS": código, grupo, descripción, actuación, Habilitada_Activas, Habilitada_Historicos.

Easy_Configure_BBDD.vi

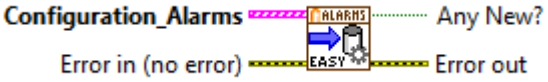

Figura 19: Easy_Configure_BBDD

Esta función es similar a la anterior pero gestiona internamente que no se inserte una configuración de alarmas repetida.

2.3.2 Eventos

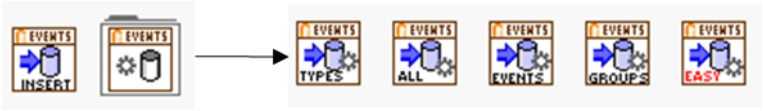

Figura 20: Write VIs (Eventos)

Register_Events_Locally.vi

Figura 21: Register_Events_Locally

Esta función permite insertar un array de eventos en la base de datos a través del servidor. Cada evento estará definido por un cluster en el array de entrada, en los cuales se deberá rellenar la siguiente información: instante, código, parámetros. Para que funcione este VI es necesario tener abierto el servidor de eventos previamente.

Configure_Event_Types_Locally.vi

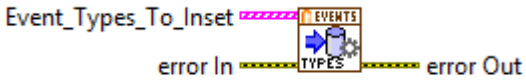

Figura 22: Register_Events_Locally

Esta función permite insertar un array de configuración de tipos de evento. Cada elemento del array está formado por un cluster con una variable por cada campo de la tabla "CONFIGURATION_EVENT_TYPES": Name, serverity, icon, enable, visible.

Configure_Group_Events_Locally.vi

Figura 23: Configure_Group_Events_Locally

Esta función permite insertar un array de configuración de grupos de evento. Cada elemento del array está formado por un cluster con una variable por cada campo de la tabla "CONFIGURATION_GROUPS": name, color, code, enable, visible.

Configure_Events_Locally.vi

Figura 24: Configure_Events_Locally

Esta función permite insertar un array de configuración de eventos. Cada elemento del array está formado por un cluster con una variable por cada campo de la tabla "CONFIGURATION_EVENTS": name, code, code_group, parameters, description, Enable y visible.

Configure_Events_Groups_Types.vi

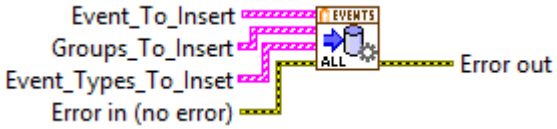

Figura 25: Configure_Events_Groups_Types

Esta función es simplemente la conjunción de las tres funciones anteriores: Configuración de tipos de evento, grupos de evento, y eventos.

Easy_Configure_BBDD.vi

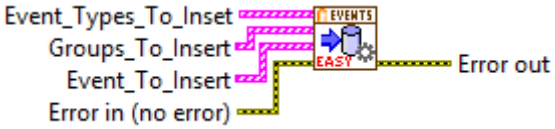

Figura 26: Easy_Configure_BBDD

Esta función es similar a la anterior pero gestiona la inserción para descartar los eventos, tipos y grupos repetidos en la base de datos.

2.4 Paleta: VIs de lectura

Esta paleta también presenta diferencias en la versión de alarmas y eventos.

2.4.1 Alarmas

Figura 27: Read VIs (Alarmas)

Read _Active _Alarms _Locally.vi

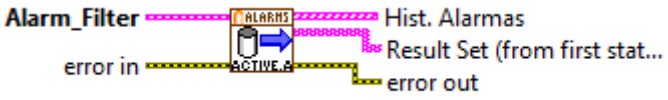

Figura 28: Read _Active _Alarms _Locally

Está función devuelve la lectura de todas las alarmas activas de la base de datos que cumplan con los filtros establecidos en el cluster de entrada. La lectura se devuelve en dos formatos distintos:

- **Tabla:** como si se hiciese una lectura directa de la base de datos.
- **Array de cluster:** donde cada elemento es una alarma y cada variable del cluster es un campo de la base de datos.

El cluster de entrada permite establecer:

- **Filtros:** códigos, grupos, rango de fechas y descripción.
- **Orden:** ascendente o descendente de todos los campos disponibles en la vista VIEW_ACTIVE_ALARMS.

Read_History_Alarms_Locally.vi

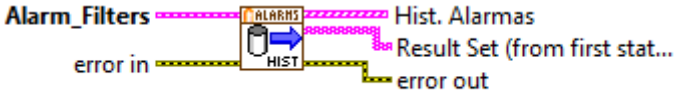

Figura 29: Read_History_Alarms_Locally

Esta función se utiliza de la misma forma que "Read_Active_Alarms_Locally" cambiando la lectura de activas por históricos de alarmas (VIEW_HISTORY_ALARMS).

Read_Duration_Alarms_Locally.vi

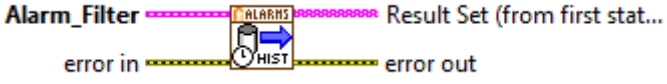

Figura 30: Read_Duration_Alarms_Locally

Esta función se utiliza de la misma forma que "Read_Active_Alarms_Locally" cambiando la lectura de activas por históricos de duración de alarmas (VIEW_HISTORY_DURATION_ALARMS).

Get_Count_Alarms_Locally.vi

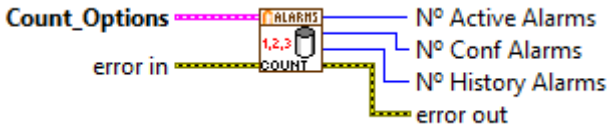

Figura 31: Get_Count_Alarms_Locally

Esta función devuelve el número de alarmas activas, el número de históricos de alarmas y el número de alarmas configuradas en la base de datos de alarmas. En la entrada (Count_Options) el usuario ha de especificar, mediante 3 booleanos, cuáles de los últimos 3 parámetros se desea leer. La finalidad de la entrada es evitar tener que contar de la base lo que no sea de interés para el usuario.

2.4.2 Eventos

Figura 32: Read VIs (Eventos)

Get_History_Events_Locally.vi

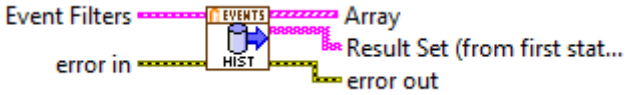

Figura 33: Get_History_Events_Locally

Esta función devuelve la lectura de todo el histórico de eventos de la base de datos que cumpla con los filtros establecidos en cluster de entrada. La lectura se devuelve en dos formatos distintos:

- **Tabla:** como si se hiciese una lectura directa de la base de datos.
- **Array de cluster:** donde cada elemento es un evento y cada variable del cluster es un campo de la vista VIEW_HISTORY_EVENTS.

El cluster de entrada permite establecer:

- **Filtros:** códigos de evento, códigos de grupo, nombre de grupo, tipos de evento, rango de fechas y descripción de eventos y parámetros.
- **Orden**: ascendente o descendente de todos los campos disponibles en la vista VIEW_HISTORY_EVENTS.

Get_Count_Events_Locally.vi

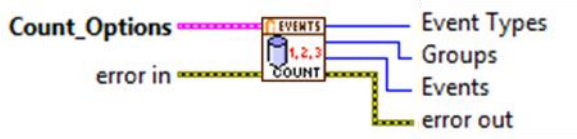

Figura 34: Get_Count_Events_Locally

Esta función devuelve número de eventos, grupos de evento y tipos de evento configurados en la base de datos de eventos. En la entrada (Count_Options) el usuario ha de especificar, mediante 3 booleanos, cuáles de los últimos 3 parámetros se desea leer. La finalidad de la entrada es evitar tener que contar de la base lo que no sea de interés para el usuario.

2.5 Paleta: VIs Dinámicos (DYN)

El toolkit contempla una serie de funciones reentrantes agrupadas en parejas (cliente y servidor). Estas funciones no son más que una réplica de algunas de las funciones de lectura y escritura anteriormente descritas en formato de peticiones TCP. Por tanto, estas funciones deberán de usarse cuando el servidor de Alarmas/Eventos esté en una aplicación distinta de donde se desea realizar la escritura y/o lectura.

Las funciones cliente hacen la petición al servidor remoto y las funciones DYN (o servidor) representan la parte de código que el servidor de alarmas ejecuta cuando se hace la petición. **A nivel de usuario, se utilizaran las funciones "Cliente", ya que las funciones "DYN" se integran en el servidor de alarmas por defecto** (únicamente con abrir el servidor se están cargando estas funciones en memoria).

De la misma forma que ocurre con los VIs propios del panel de alarmas y eventos, estos VIs dinámicos (DYN) se incluyen en la paleta por una practicidad de acceso al código para los desarrolladores.

Todas estas funciones presentan 3 entradas comunes intrínsecas de la conexión con el servidor TCP:

- **Configuración Socket:** cluster formado por dos variables:
  - **IP:** dirección IP del servidor TCP con el que se realiza la conexión.
  - **Port:** puerto del servidor con el que se realiza la conexión.
- **Connection Options:** cluster formado por dos elementos identificadores de la conexión TCP con el servidor:
  - **ID:** numérico identificador de conexión local. Este número no se ha de repetir con ninguna de las conexiones simultáneas que se hagan desde la aplicación local.
  - **Function name:** identificador de conexión con el servidor. Este string no debe coincidir con ninguna de las conexiones simultáneas que se hagan con el servidor desde cualquier aplicación.
- **Timeout:** tiempo máximo en milisegundos para realizar la petición al servidor.

2.5.1 Alarmas

Figura 35: VIs Dinámicos (Alarmas)

Client_VI_Actualize_Alarms.vi

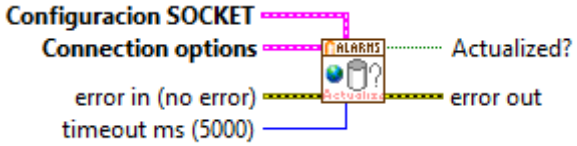

Figura 36: Client_VI_Actualize_Alarms

Esta función es usada por el panel de alarmas para conocer si es necesario un refresco de la interfaz debido a que se ha insertado una alarma en la base de datos. Normalmente no será necesario utilizar esta función, no obstante, se incluye por si el usuario necesitara utilizarla.

Client_VI_Insert_Alarms.vi

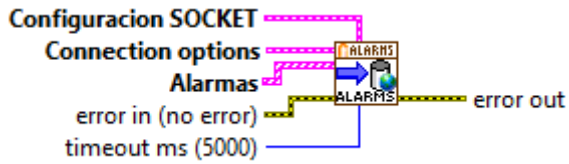

Figura 37: Client_VI_Insert_Alarms

Esta función es la versión TCP del VI Register_Alarms_Locally.

Client_VI_Read_Historic_Alarms.vi

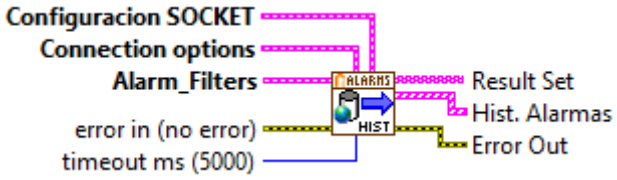

Figura 38: Client_VI_Read_Historic_Alarms

Esta función es la versión TCP del VI Read_History_Alarms_Locally.

Client_VI_Read_Duration_Alarms.vi

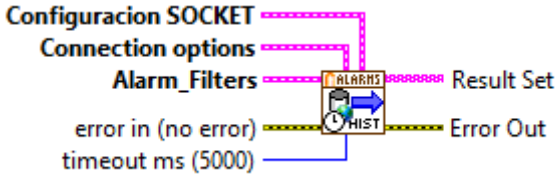

Figura 39: Client_VI_Read_Duration_Alarms

Esta función es la versión TCP del VI Read_Duration_Alarms_Locally.

Client_VI_Read_Active_Alarms.vi

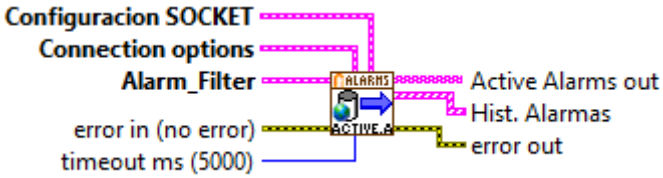

Figura 40: Client_VI_Read_Active_Alarms

Esta función es la versión TCP del VI Read _Active _Alarms _Locally.

Client_VI_Get_Count_Alarmas.vi

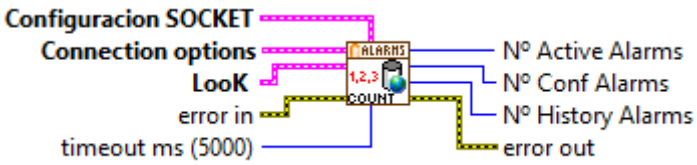

Figura 41: Client_VI_Get_Count_Alarmas

Esta función es la versión TCP del VI Get_Count_Alarms_Locally.

2.5.2 Eventos

Figura 42: VIs Dinámicos (Eventos)

Client_VI_Actualize_Events.vi

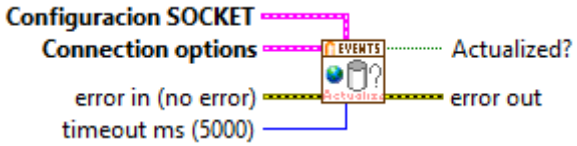

Figura 43: Client_VI_Actualize_Events

Esta función es usada por el panel de eventos para conocer si es necesario un refresco de la interfaz debido a que se ha insertado un evento en la base de datos. Normalmente, no será necesario utilizar esta función, no obstante se incluye por si el usuario necesitara utilizarla.

Client_VI_Insert_Events.vi

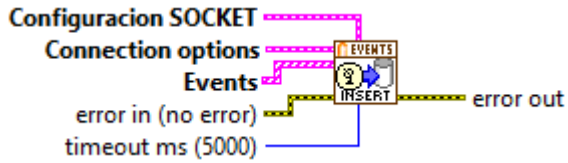

Figura 44: Client_VI_Insert_Events

Esta función es la versión TCP del VI Register_Events_Locally.

Client_VI_Read_Historic_Events.vi

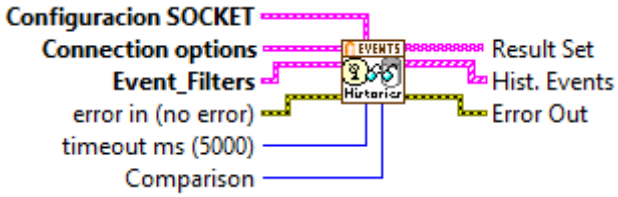

Figura 45: Client_VI_Read_Historic_Events

Esta función es la versión TCP del VI Get_History_Events_Locally.

2.6 Ejemplos

Figura 46: Ejemplos

Tanto la paleta de alarmas como la de eventos presentan un par de ejemplos de uso de las funciones descritas anteriormente. Básicamente muestran como lanzar un servidor de Alarmas/Eventos junto con un panel de visualización de Alarmas/Eventos, además de mostrar cómo realizar inserciones en la base de datos. A continuación, se explica el funcionamiento de este código sobre el ejemplo de alarmas debido a que el de eventos es muy similar:

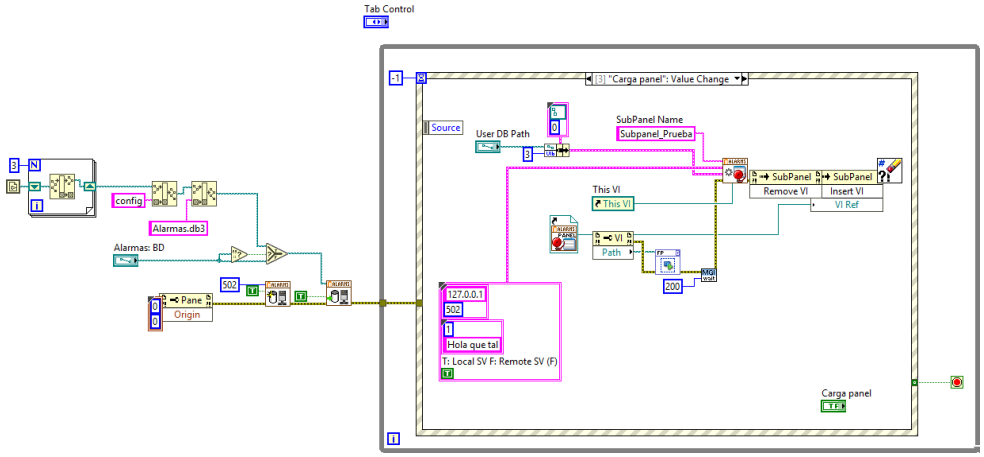

Figura 47: Código del ejemplo de alarmas

INICIALIZACIÓN:

Lanzamiento de servidores: Utilizando la paleta del servidor se abre el servidor y se configura utilizando las funciones: “Open_SQLite_Server_Alarms” y “Load_BBDD_in_SQLite_Sever_Alarms”.

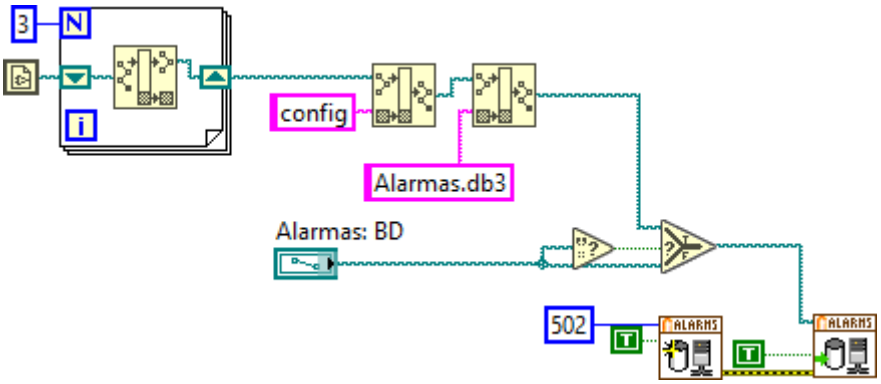

Figura 48: Arranque del servidor

EJECUCIÓN: EVENTOS

Mediante un par de casos del "Event estructure", se muestra al usuario como realizar tanto la inicialización de un panel de visualización, como la inserción de alarmas y eventos en la base de datos:

- Lanzamiento dinámico del panel:
  - Se obtiene la referencia del VI del panel a arrancar.
  - Se arranca utilizando las funciones del VI Server.
  - Se parametriza utilizando la función "Config_SubPanel_Alarms"
  - Se inserta la referencia en un "Pane Control" de LabVEIW para poder visualizar la información gráfica del VI.

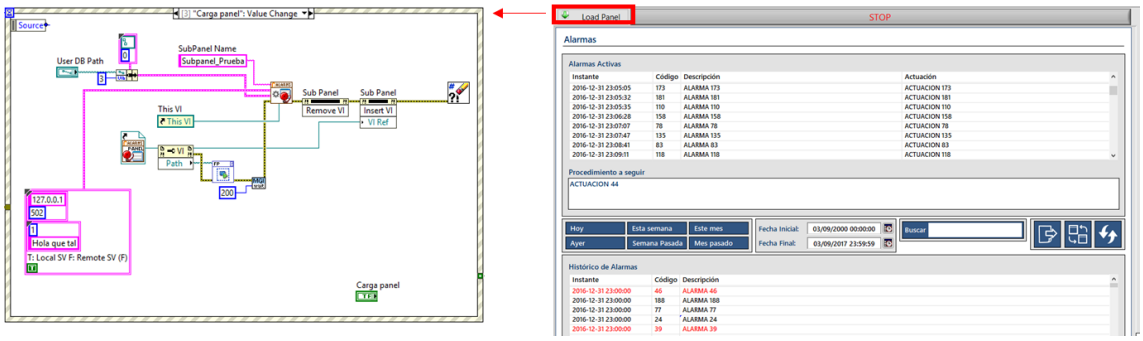

Figura 49: Arranque dinámico del panel

- Inserción: Simplemente haciendo uso de la función "Register_Alarms_Locally".

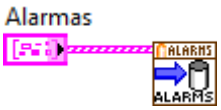

Figura 50: Ejemplo de inserción de alarmas

CIERRE

Por último, se muestra el uso de las funciones para cerrar el panel y del servidor cuando se pulsa el botón "Stop". (Stop_Subpanel_Alarma/Eventos y Stop_SQLite_Server_Alarms/Events)

Figura 51: Ejemplo de cierre del panel y servidor

3. PANELES DE VISUALIZACIÓN

3.1 Introducción

En el toolkit encontramos dos paneles de visualización, uno para alarmas y otro para eventos. En este apartado se procederá a explicar sus funcionalidades, no obstante, debido a que presenta similitudes, algunos detalles de su utilización se explicarán conjuntamente.

Los paneles de alarmas y eventos son aplicaciones independientes con interfaz gráfica integrada para explotar la información de la base de datos de alarmas y eventos.

Debido a que estos paneles pueden llegar a presentar gran cantidad de variantes sobre su interfaz, la guía se realizará sobre una variante que se considera completa y más habitualmente utilizada. A continuación, se muestra una vista completa de estas dos interfaces

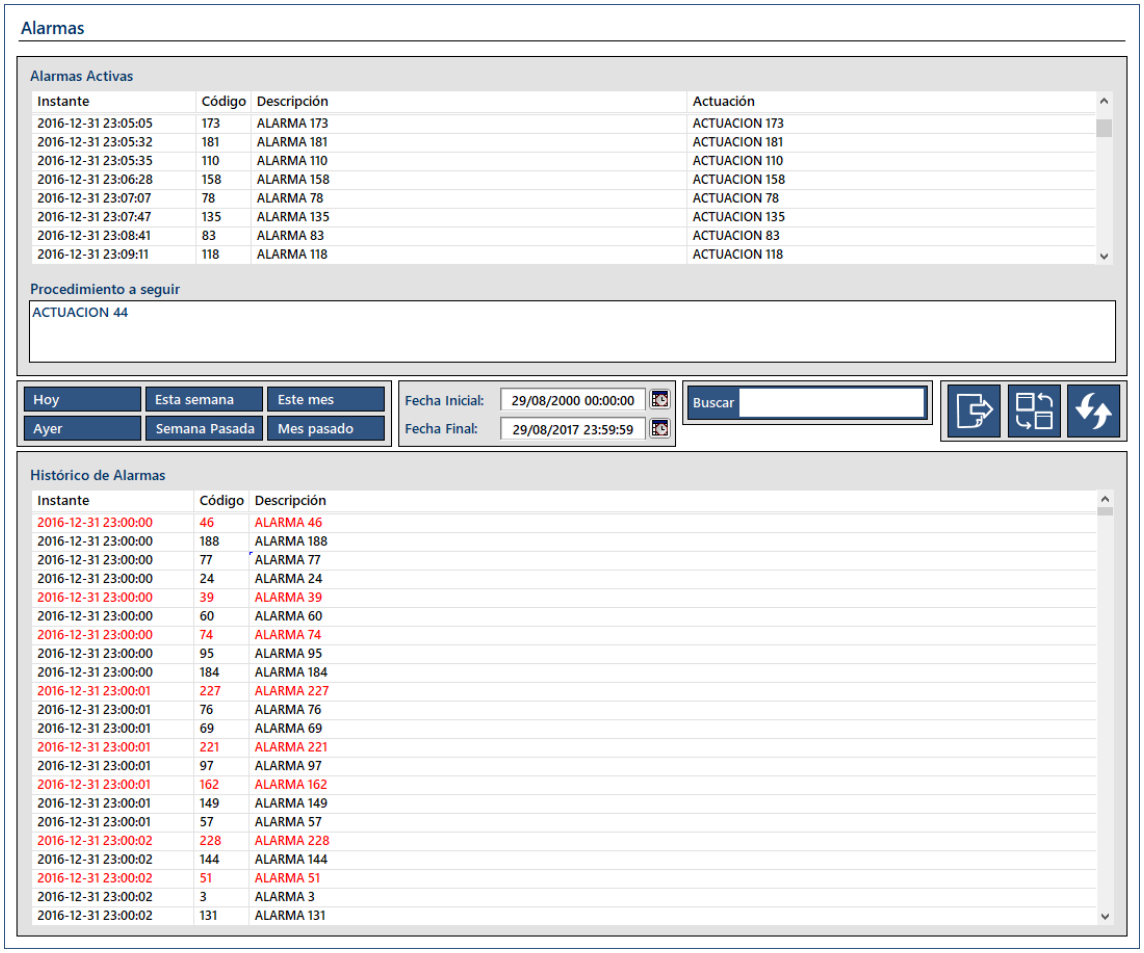

Figura 52: Panel de alarmas

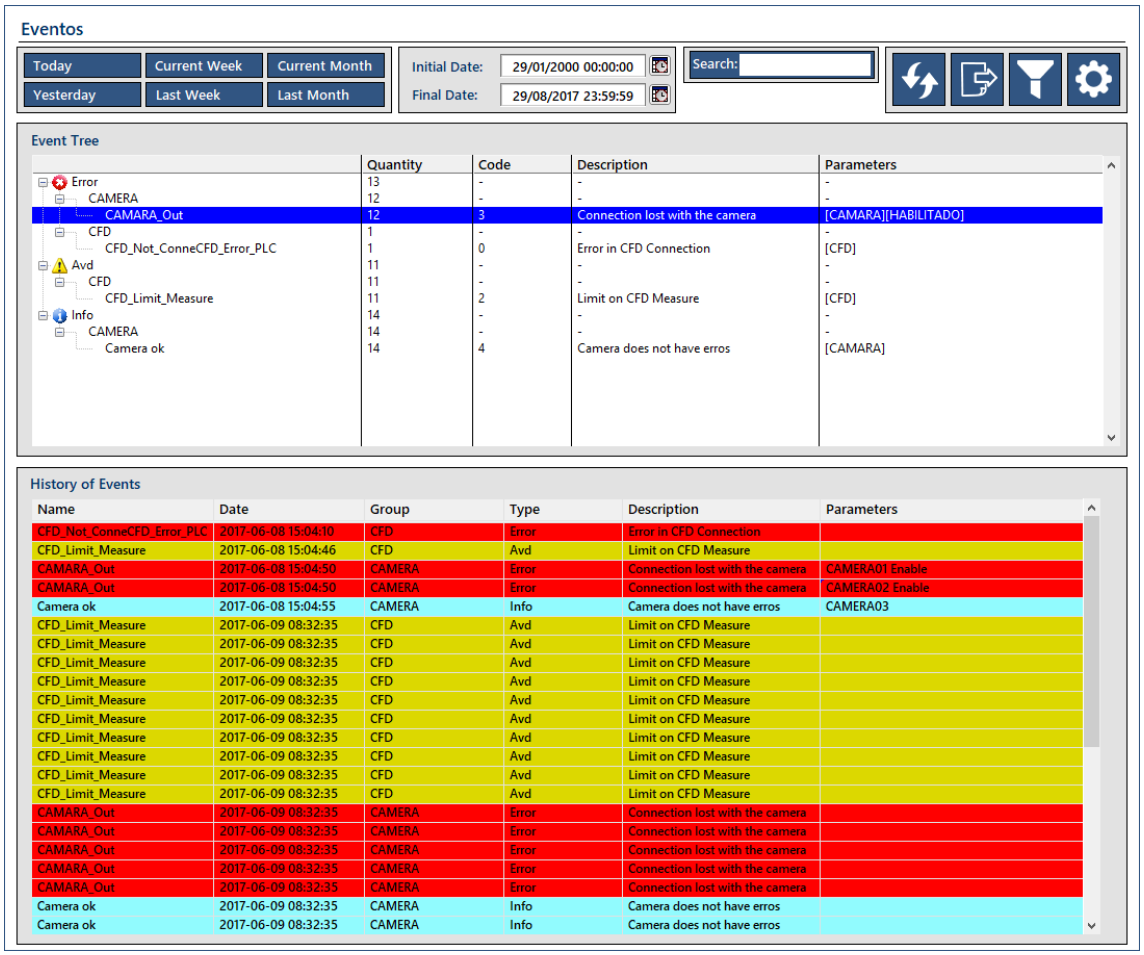

Figura 53: Panel de eventos

3.2 Funcionamiento: Elementos comunes

CONTROLES DE TIEMPO

Ambos paneles se basan en explotar la información de sus respectivas bases de datos, es decir, son paneles de consulta. Es por ello que ambos presentan los mismos controles de selección de ventana temporal:

- **Controles booleanos de tiempo:** 6 botones (hoy, ayer, esta semana, semana pasada, este mes, mes pasado) para establecer un rango de fechas siempre relativo al día actual.
- **Controles de fecha Inicial-Final**: para establecer un rango de búsqueda personalizado.

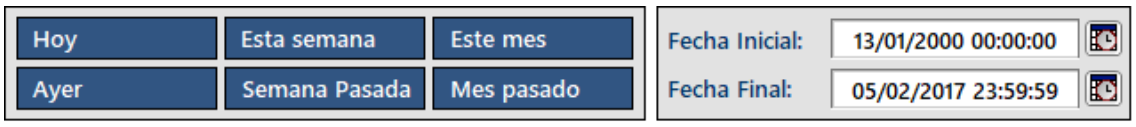

Figura 54: Controles de selección de ventana temporal

Cada vez que pulsa un botón, los controles de fecha se actualizan con el rango de fechas correspondiente al botón que se ha pulsado, además de despulsar el último botón activo (ya que solo puede haber uno pulsado a la vez). Del mismo modo, cuando se registra un cambio en los controles de fecha, se establece un rango de búsqueda personalizado y, por tanto, se despulsan todos los botones.

Este rango de fechas afecta a la cantidad de histórico de datos que se van a leer de las correspondientes bases de datos para actualizar las diversas tablas o árboles.

CONTROL DE BÚSQUEDA

Figura 55: Control de búsquedas

Este control se utiliza para realizar búsquedas en los datos que están presentes en las tablas de históricos (Histórico de eventos o Históricos de alarmas). Cuando el usuario escribe un conjunto de palabras:

- **Alarmas:** aparecerán en la tabla de Históricos (Histórico de alarmas e Histórico Duración) todas las alarmas que contengan en su descripción (tabla CONFIGURATION_ALARMS de la base de datos) las palabras escritas en el citado control.
- **Eventos:** aparecerán en la tabla de Históricos todas los eventos que contengan en su nombre (campo descriptivo de la tabla CONFIGURATION_EVENTS) las palabras escritas en el citado control.

Además, este control permite ejecutar un par de comandos adicionales en caso de que se requiera realizar una búsqueda por otros campos. Los comandos son:

- **“Code:”** seguido del código de la Alarma/Evento. Es posible poner tantos códigos como se quiera siempre que estén separados por comas. También es posible establecer rangos de números separados mediante guiones. Ejemplo: “Code: 1, 2, 8, 10-20”. En este caso las tablas de históricos se actualizarían con las Alarmas/Eventos correspondientes a los códigos 1, 2, 8 y los códigos del 10 al 20.
- **“Group:”** seguido del nombre del grupo de la Alarma/Evento. También es posible establecer varios grupos separando estos por comas. Por ejemplo: “Group:CFD,General”. En este caso las tablas de históricos se actualizarían con todas las alarmas y eventos correspondientes a los grupos CFD y General.

CONTROLES BOOLEANOS

Figura 56: Botones comunes de los paneles de alarmas/eventos

Estos tres controles están disponibles en ambos paneles de alarmas y eventos. En ambos casos desempeñan la misma función:

1. **Booleano de refresco de interfaz:** para actualizar la interfaz a demanda. Este control tiene mucho sentido cuando la actualización automática de la interfaz está desactivada (desde las opciones de configuración del panel).
2. **Booleano de exportación de datos:** este botón exporta a un fichero resumen en formato .xlsx o .txt (En función de la configuración de panel) toda la información de la tabla de histórico de Alarmas/Eventos.
3. **Booleano de acceso a la configuración:** mediante este botón accedemos a una ventana de diálogo para configurar el panel, la cual se explica más adelante. Normalmente, a esta configuración se accede desde las herramientas del framework y no desde el panel en tiempo de ejecución. Esto es debido a que muchas de las opciones de configuración no están pensadas para ser cambiadas más de una vez. No obstante, se deja a la elección del usuario que usa el toolkit el incluirlo en el panel o no. Como se puede observar en la **¡Error! No se encuentra el origen de la referencia.**Figura 52, este booleano no aparece, sin embargo en la Figura 53, sí. Esto es debido a que en el caso del panel de alarmas se parte de una plantilla donde el botón está oculto.

TABLAS: Comportamiento común

Existen un total de 4 tablas contando tanto el panel de alarmas como el de eventos. Es cierto que cada una muestra una información muy diferente, pero todas ellas poseen un comportamiento y unas características similares, las cuales se explican a continuación:

- **Columnas y cabeceras:** los campos visibles, el orden de los mismo y el ancho ocupado por cada uno de ellos es completamente configurable desde las opciones de configuración del panel (se explica más adelante). Esto es debido a que en algunos proyectos se preferirá mostrar unos campos u otros o incluso modificar la cantidad de espacio destinado a cada columna.
- **Gestión de orden ascendente y descendente:** clicando en los campos de las cabeceras, las filas de las tablas se reordenan (en orden ascendente o descendente) según el tipo de dato del campo donde se clique.
- **Foco de la fila seleccionada:** cada vez que se selecciona una fila, esta se resalta mediante un color que es modificable desde las opciones de configuración de panel. Ejemplo: **¡Error! No se encuentra el origen de la referencia.**

3.3 Alarmas: Funcionamiento

Adicionalmente a los controles que se han comentado anteriormente, el panel de alarmas posee diversos elementos:

TABLA DE ALARMAS ACTIVAS

Alarmas Activas

Instante	Código	Descripción	Actuación
2016-12-31 23:05:05	173	ALARMA 173	ACTUACION 173
2016-12-31 23:05:32	181	ALARMA 181	ACTUACION 181
2016-12-31 23:05:35	110	ALARMA 110	ACTUACION 110
2016-12-31 23:06:28	158	ALARMA 158	ACTUACION 158
2016-12-31 23:07:07	78	ALARMA 78	ACTUACION 78
2016-12-31 23:07:47	135	ALARMA 135	ACTUACION 135
2016-12-31 23:08:41	83	ALARMA 83	ACTUACION 83
2016-12-31 23:09:11	118	ALARMA 118	ACTUACION 118

Figura 57: Tabla de alarmas activas

Esta tabla muestra todas las alarmas activas de la base de datos, es decir, todas las alarmas de la vista VIEW_HISTORY_ALARMS. Debido a que son alarmas que siguen activas (no son un histórico) los datos de esta tabla no obedecen al marco de los controles temporales (**¡Error! No se encuentra el origen de la referencia.**) ya que conceptualmente suponen algo crítico y deben estar visibles en todo momento.

TABLA DE HISTÓRICO DE ALARMAS

Esta tabla de alarmas refleja el histórico de cambios, es decir, presenta una fila cada vez que una alarma ha cambiado de estado, **mostrando siempre en rojo las alarmas que cambiaron su estado a true**. A diferencia de la tabla de activas, una fila roja en esta tabla no tiene porqué representar una alarma que está actualmente activa, sino el instante en el que se dio la activación.

Histórico de Alarmas

Instante	Código	Descripción
2017-06-13 22:43:38	7	ALARMA 7
2017-06-13 22:43:38	7	ALARMA 7
2017-06-13 22:43:38	6	ALARMA 6
2017-06-13 22:43:38	6	ALARMA 6
2017-06-05 22:59:28	6	ALARMA 6
2017-06-05 22:59:28	6	ALARMA 6
2017-06-05 22:59:28	6	ALARMA 6
2017-06-05 22:58:53	6	ALARMA 6
2017-06-05 22:58:53	6	ALARMA 6
2017-01-16 12:37:24	6	ALARMA 6
2017-01-16 12:36:46	6	ALARMA 6
2017-01-16 12:36:43	6	ALARMA 6
2017-01-16 12:34:31	6	ALARMA 6
2017-01-16 12:34:03	6	ALARMA 6
2017-01-16 12:33:57	6	ALARMA 6
2017-01-16 12:31:11	6	ALARMA 6
2017-01-16 12:30:51	6	ALARMA 6
2017-01-16 12:30:13	6	ALARMA 6
2017-01-16 12:27:52	6	ALARMA 6
2017-01-16 12:27:45	6	ALARMA 6
2017-01-16 12:26:21	6	ALARMA 6
2017-01-16 12:26:02	6	ALARMA 6

Figura 58: Tabla de histórico de alarmas

TABLA DE HISTÓRICO DE DURACIÓN

La tabla de histórico de duración muestra el histórico de una forma diferente al anterior, ya que presenta una fila por cada combinación de inicio y fin de alarma. Mostrando en rojo las alarmas que han dado inicio pero no han dado fin, es decir, las activas.

Histórico Duracion

Instante Inicial	Instante Final	Duración	Estado	Descripción
2017-06-05 22:59:28	2017-06-13 22:43:38	7d 23:44:10	No activa	ALARMA 6
2016-12-31 23:00:54	2016-12-31 23:59:57	0d 00:59:03	No activa	ALARMA 45
2016-12-31 23:00:16	2016-12-31 23:59:02	0d 00:58:46	No activa	ALARMA 27
2016-12-31 23:01:11	2016-12-31 23:59:45	0d 00:58:34	No activa	ALARMA 26
2016-12-31 23:00:42	2016-12-31 23:59:09	0d 00:58:27	No activa	ALARMA 37
2016-12-31 23:00:04	2016-12-31 23:58:30	0d 00:58:26	No activa	ALARMA 42
2016-12-31 23:00:27	2016-12-31 23:58:44	0d 00:58:17	No activa	ALARMA 9
2016-12-31 23:00:37	2016-12-31 23:58:46	0d 00:58:09	No activa	ALARMA 158
2016-12-31 23:00:06	2016-12-31 23:58:07	0d 00:58:01	No activa	ALARMA 2
2016-12-31 23:01:57	2016-12-31 23:59:35	0d 00:57:38	No activa	ALARMA 88
2016-12-31 23:01:39	2016-12-31 23:59:16	0d 00:57:37	No activa	ALARMA 183
2016-12-31 23:00:13	2016-12-31 23:57:48	0d 00:57:35	No activa	ALARMA 141
2016-12-31 23:01:15	2016-12-31 23:58:50	0d 00:57:35	No activa	ALARMA 47
2016-12-31 23:00:51	2016-12-31 23:58:17	0d 00:57:26	No activa	ALARMA 111
2016-12-31 23:01:01	2016-12-31 23:58:24	0d 00:57:23	No activa	ALARMA 32
2016-12-31 23:00:56	2016-12-31 23:58:19	0d 00:57:23	No activa	ALARMA 181
2016-12-31 23:00:43	2016-12-31 23:58:05	0d 00:57:22	No activa	ALARMA 209
2016-12-31 23:00:22	2016-12-31 23:57:31	0d 00:57:09	No activa	ALARMA 202
2016-12-31 23:01:06	2016-12-31 23:58:14	0d 00:57:08	No activa	ALARMA 39
2016-12-31 23:00:16	2016-12-31 23:57:14	0d 00:56:58	No activa	ALARMA 143
2016-12-31 23:00:54	2016-12-31 23:57:40	0d 00:56:46	No activa	ALARMA 171
2016-12-31 23:01:42	2016-12-31 23:58:22	0d 00:56:40	No activa	ALARMA 216

Figura 59: Tabla de histórico Duración

BOTON DE CAMBIO DE HISTÓRICO

Como se puede apreciar en la Figura 52, la tabla de histórico de alarmas e histórico duración no aparecen simultáneamente. Son dos opciones de visualización diferentes que pueden resultarle útil al usuario en un momento concreto. Por esta razón, el panel de alarmas presenta este booleano, para conmutar entre la visibilidad de las tablas.

Figura 60: Botón de conmutación de tablas de históricos

INDICADOR DE ACTUACIÓN

La finalidad de este indicador es muy simple. Mostrar el campo “Actuación” (Correspondiente a la tabla CONFIGURATION_ALARMS) de la alarma seleccionada en la tabla de activas.

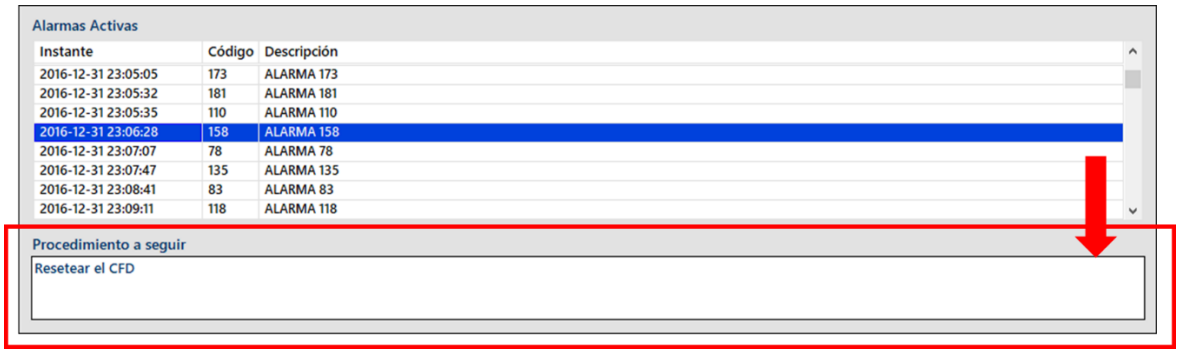

Figura 61: Indicado de actuación

3.4 Eventos: Funcionamiento

El panel de eventos también posee algunos elementos adicionales.

ÁRBOL DE EVENTOS

Muestra un resumen (Contaje) de todos los eventos presentes en la base de datos (dentro del rango de fechas establecido), además de otra información. Presenta 3 niveles:

1. **Tipo de evento:** los tipos se ordenan en función de la severidad y muestran el icono especificado en la base de datos. Dentro de cada uno encontramos los grupos con eventos que pertenecen al tipo y el contaje de todos ellos.
2. **Grupos de eventos:** subcategoría que contiene dentro el contaje de todos los eventos que pertenecen al grupo.
3. **Eventos:** por cada evento encontramos un contaje del número de veces que se ha repetido el mismo, además de información como tags de parámetros, código y descripción del evento.

Event Tree

	Quantity	Code	Description	Parameters
Error	13	-	-	-
CAMERA	12	-	-	-
CAMARA_Out	12	3	Connection lost with the camera	[CAMARA][HABILITADO]
CFD	1	-	-	-
CFD_Not_ConneCFD_Error_PLC	1	0	Error in CFD Connection	[CFD]
Avd	11	-	-	-
CFD	11	-	-	-
CFD_Limit_Measure	11	2	Limit on CFD Measure	[CFD]
Info	14	-	-	-
CAMERA	14	-	-	-
Camera ok	14	4	Camera does not have erros	[CAMARA]

Figura 62: Árbol de eventos

Cabe destacar que el árbol se comporta a modo de filtro para la tabla "Histórico de eventos" debido a que cada vez que clicamos en un elemento del árbol, la tabla se actualiza solo con esos elementos. Es decir, si se hace clic en uno o varios (CTL+ Clic) grupos/tipos/eventos, la tabla de históricos mostrará solo los elementos seleccionados. Además, para evitar malas selecciones, el árbol gestiona que los elementos seleccionados (en caso de ser múltiples) tengan que ser de la misma clase, es decir, tipos, grupos o eventos.

TABLA DE HISTÓRICO DE EVENTOS

Esta tabla muestra todos los eventos producidos dentro de los filtros establecido por el panel: filtro de fechas, parámetros, filtro de selección del árbol y filtro de búsqueda. Cada fila representa una instancia de un evento, y cada evento se colorea con el color asociado al grupo al que pertenece.

History of Events

Name	Date	Group	Type	Description	Parameters
CFD_Not_ConneCFD_Error_PLC	2017-06-08 15:04:10	CFD	Error	Error in CFD Connection	
CFD_Limit_Measure	2017-06-08 15:04:46	CFD	Avd	Limit on CFD Measure	
CAMARA_Out	2017-06-08 15:04:50	CAMERA	Error	Connection lost with the camera	CAMERA01 Enable
CAMARA_Out	2017-06-08 15:04:50	CAMERA	Error	Connection lost with the camera	CAMERA02 Enable
Camera ok	2017-06-08 15:04:55	CAMERA	Info	Camera does not have erros	CAMERA03
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CFD_Limit_Measure	2017-06-09 08:32:35	CFD	Avd	Limit on CFD Measure	
CAMARA_Out	2017-06-09 08:32:35	CAMERA	Error	Connection lost with the camera	
CAMARA_Out	2017-06-09 08:32:35	CAMERA	Error	Connection lost with the camera	
CAMARA_Out	2017-06-09 08:32:35	CAMERA	Error	Connection lost with the camera	
CAMARA_Out	2017-06-09 08:32:35	CAMERA	Error	Connection lost with the camera	
CAMARA_Out	2017-06-09 08:32:35	CAMERA	Error	Connection lost with the camera	
Camera ok	2017-06-09 08:32:35	CAMERA	Info	Camera does not have erros	
Camera ok	2017-06-09 08:32:35	CAMERA	Info	Camera does not have erros	

Figura 63: Tabla de histórico de eventos

FILTRO DE PARÁMETROS: POP UP

Este botón da acceso a una ventana de diálogo para establecer filtros de búsqueda de parámetros.

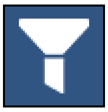

Figura 64: Botón filtro parámetros

Al clicar se accede a una ventana de diálogo que muestra una lista de todos los parámetros (Tags) diferentes encontrados en la tabla “CONFIGURATION_EVENTS” de la base de datos. Esta permite:

- Seleccionar los parámetros de la lista que se desea buscar en la tabla de históricos y en el árbol de eventos. Pulsando la tecla CTL se pueden seleccionar varios elementos a la vez.
- Cambiar la lógica de la búsqueda AND/OR, es decir, buscar eventos que tengan todos los parámetros seleccionados, o buscar los eventos que tengan alguno de los parámetros seleccionados.
- Seleccionar todos los parámetros mediante Control booleano “Select All”.

Figura 65: Ventana de filtros de parámetros

4. FRAMEWORK: HERRAMIENTAS

4.1 Introducción: Modificando el entorno de LabVIEW

El toolkit de Alarmas/Eventos pretende ofrecer un marco de trabajo cómodo para los desarrolladores que necesiten de su uso. Es por ello que el toolkit ofrece una serie de herramientas para configurar y desarrollar nuevos paneles.

Las herramientas que se describen a continuación se integran en el propio entorno de desarrollo de LabVIEW en el momento de la instalación del toolkit. De esta manera su acceso es inmediato para cualquier usuario en cualquier proyecto.

Para acceder a estas herramientas, es necesario clicar en el menú Tools: LabVIEW→ Tools→Autis Ingenieros S.L.U.

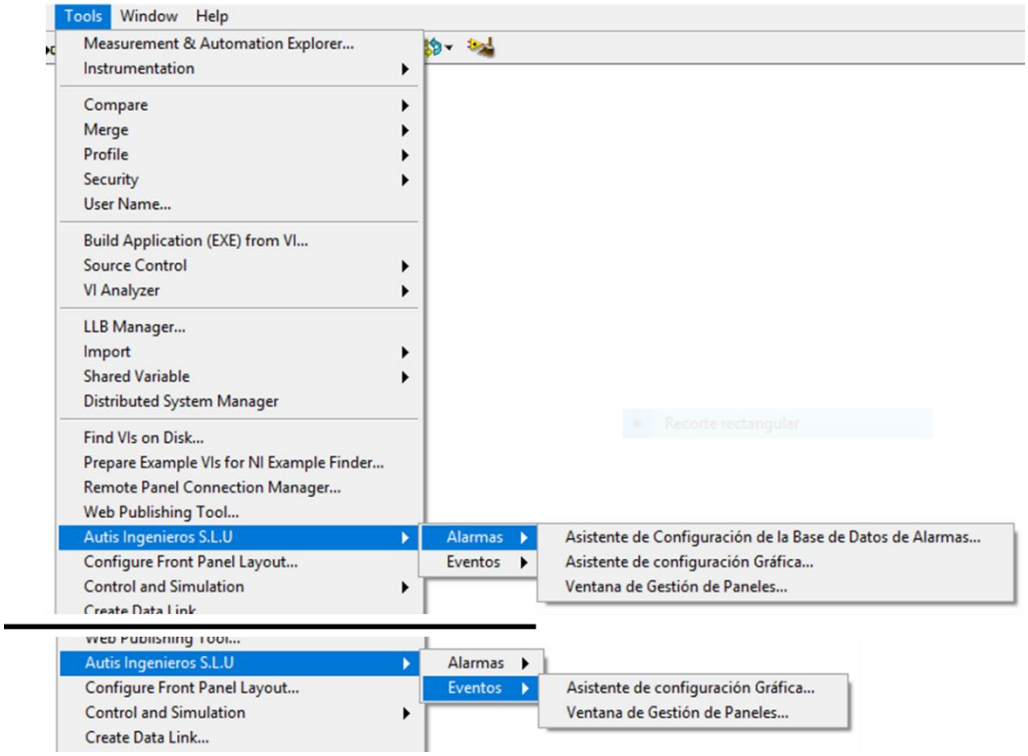

Figura 66: Acceso a las herramientas del framework

4.2 Asistente de configuración gráfica

Como bien se ha comentado anteriormente, la interfaz gráfica de estos paneles se puede modificar siempre que no se altere el comportamiento de los objetos del panel frontal. La idea principal de esta herramienta es ofrecer al usuario una manera sencilla de editar la interfaz gráfica de los paneles, conservando los diseños realizados para que estos puedan ser reciclados en otros proyectos. Para acceder a la herramienta se ha de clicar en:

- Tools→ Autis Ingenieros S.L.U. → Alarmas → Asistente de configuración gráfica
- Tools→ Autis Ingenieros S.L.U. → Eventos → Asistente de configuración gráfica

Una vez se accede a cualquiera de las opciones anteriores, se abre un VI que no está en ejecución con la interfaz gráfica (Por defecto) del panel de Alarmas/Eventos. A continuación, se muestran un par de imágenes de estos VIs.

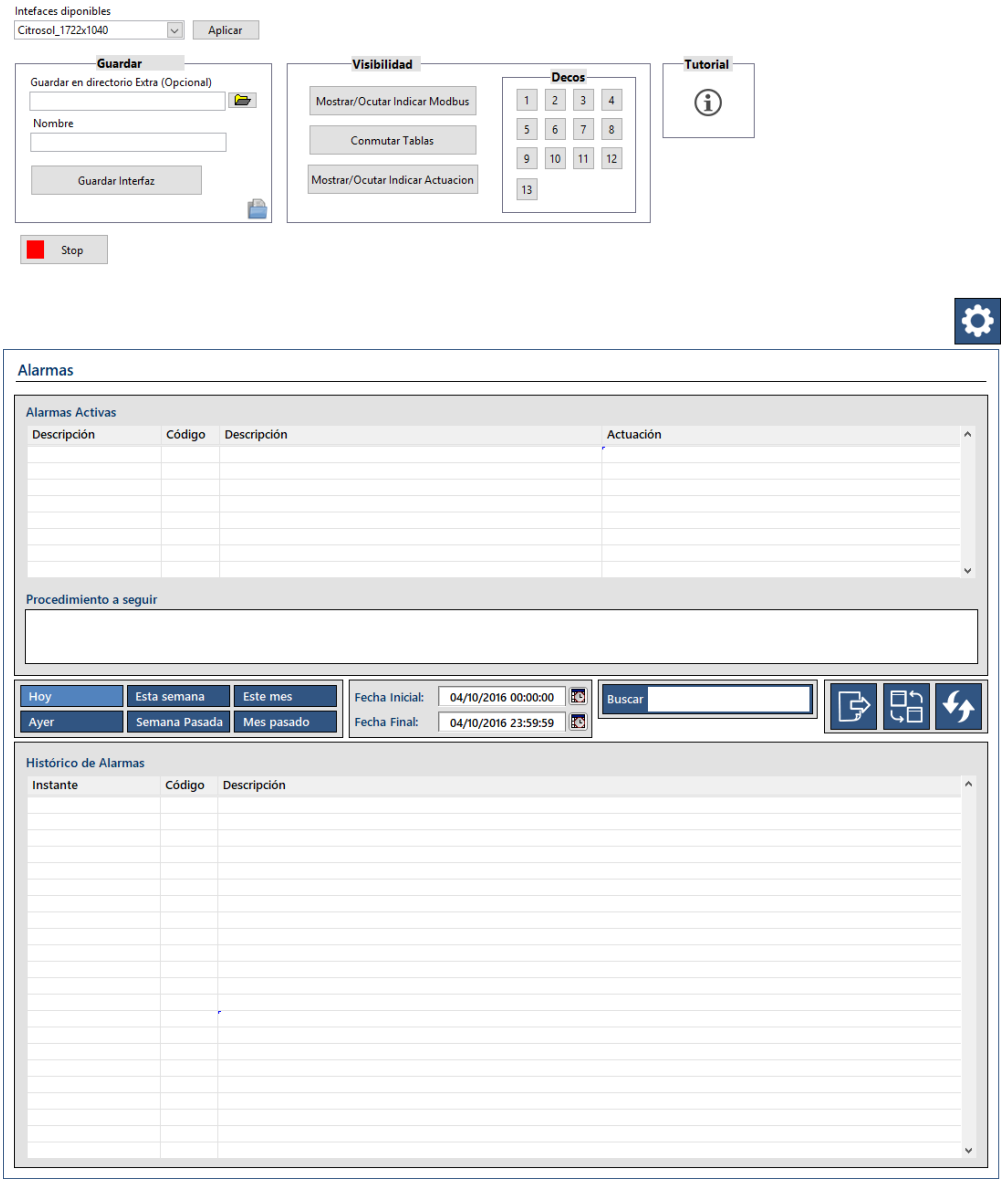

Figura 67: Asistente de configuración grafica de alarmas

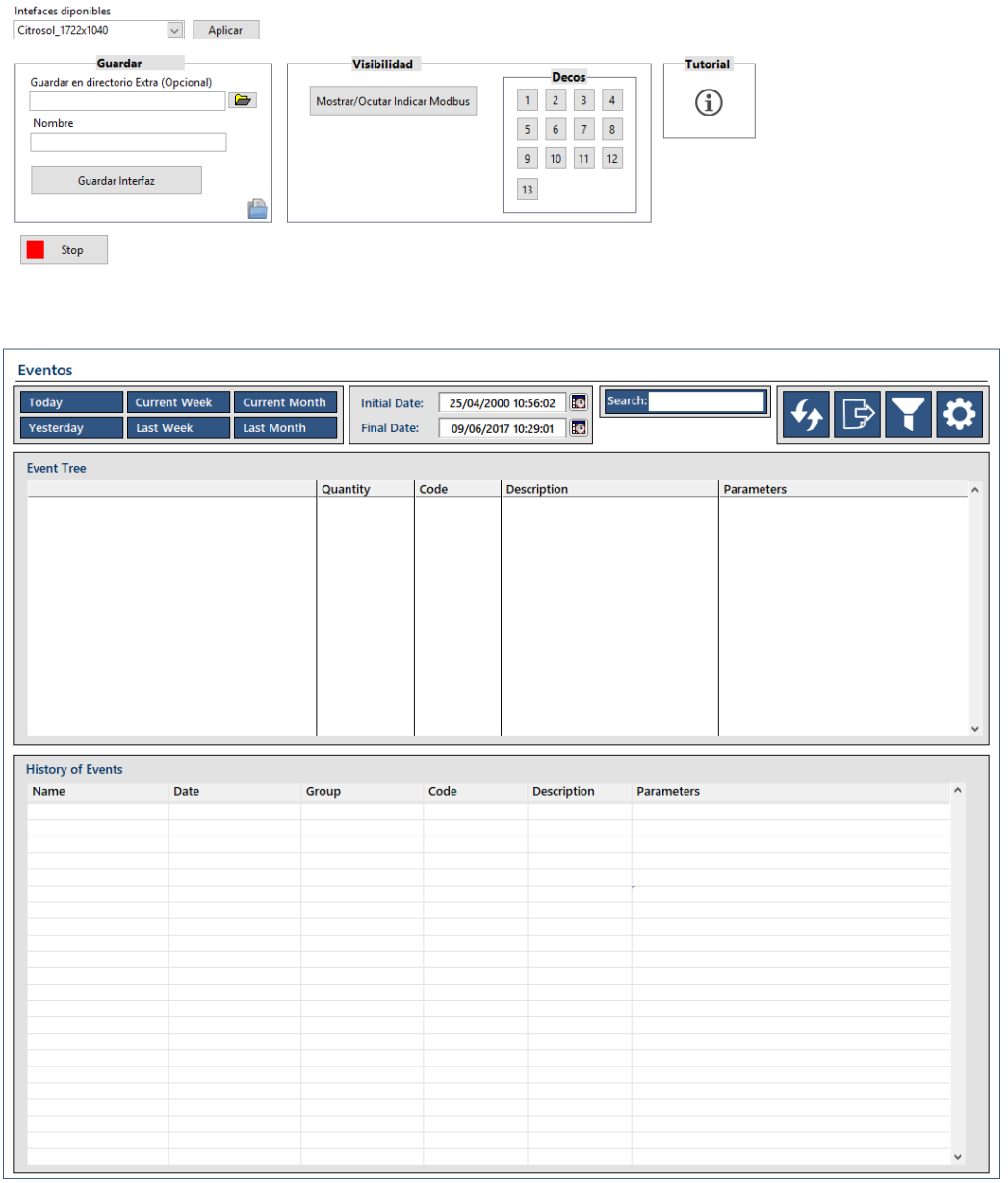

Figura 68: Asistente de configuración gráfica de Eventos

Como se puede observar en las imágenes anteriores, están formadas por dos partes:

- La parte superior del VI, muy similares en ambos casos.
- La parte inferior, con la interfaz correspondiente.

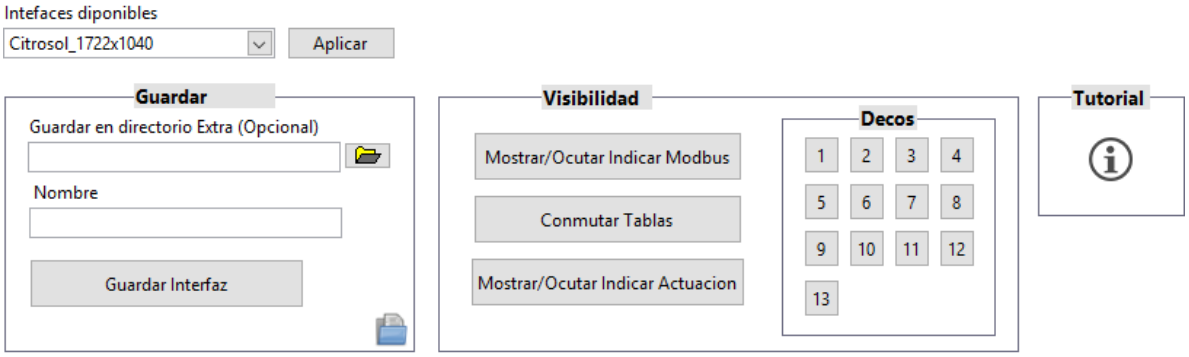

Figura 69: Parte superior del asistente de configuración gráfica

Ambas herramientas funcionan exactamente igual, por tanto, se procederá a explicarlas conjuntamente. En primer lugar, se deberá elegir la interfaz de partida, para ello, el usuario deberá:

1. Arrancar el VI.
2. Seleccionar del desplegable la interfaz deseada y clicar en el botón Aplicar. Cada vez que se clica en Aplicar, los objetos del panel frontal modifican su interfaz con la apariencia del panel aplicado. A continuación, se muestra un ejemplo del panel de alarmas con otra interfaz diferente.

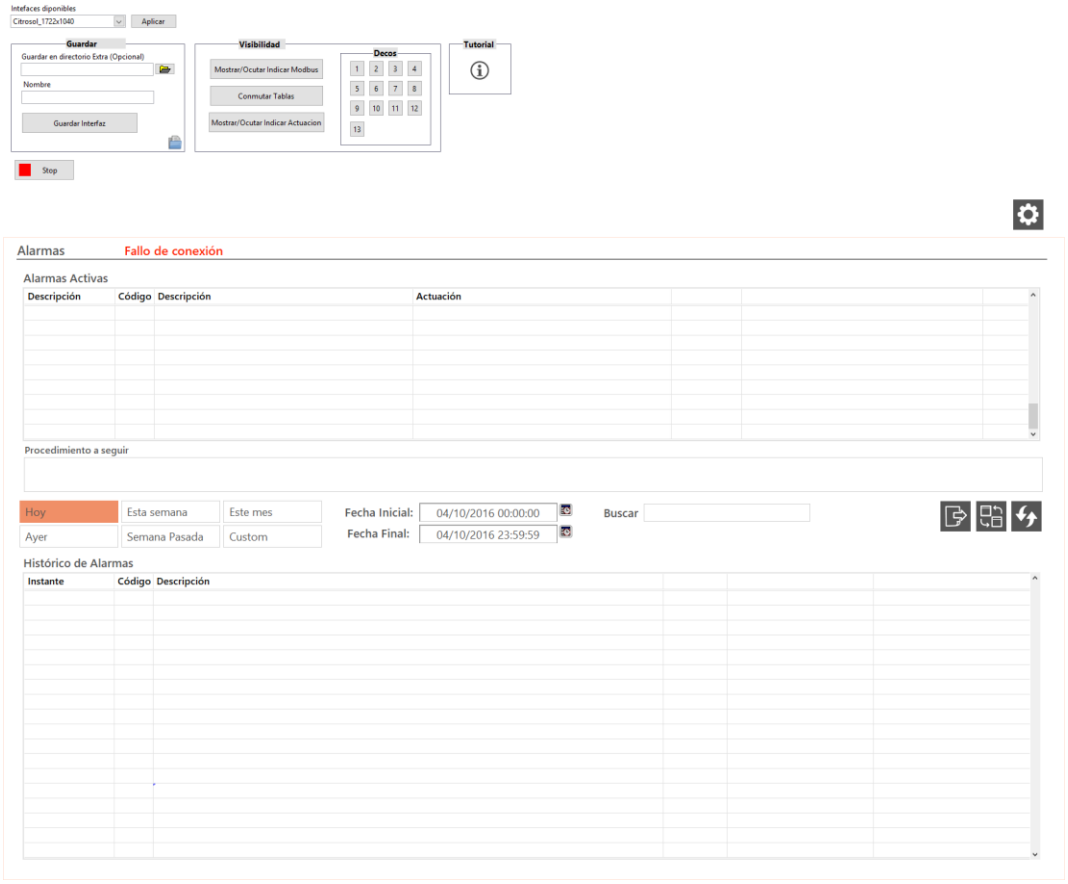

Figura 70: Ejemplo de interfaz variable

Cabe destacar que este VI es una mera copia del VI original, por tanto, en caso de eliminar cualquier elemento inintencionadamente, bastaría con cerrar la herramienta y volverla a abrir.

Una vez se elige la interfaz, el usuario deberá parar la ejecución del VI y editar la apariencia de los objetos del panel frontal mediante las herramientas del entorno de desarrollo de LabVIEW (Como haría un usuario normal desarrollando un panel independiente). Finalmente, para guardar la nueva interfaz gráfica diseñada, se deberán seguir los siguientes pasos:

1. Ejecutar de nuevo el VI.
2. Escribir un nombre valido en el control “Nombre”.
3. Clicar en el botón “Guardar Interfaz”. Esto generará un fichero de configuración con toda información gráfica de los objetos del panel frontal perteneciente al panel de Alarmas/Eventos. Por defecto, este se guarda en el directorio raíz del toolkit.
4. Parar el VI pulsando el botón “Stop”.

Adicionalmente, para facilitar la edición del panel, la herramienta cuenta con unas cuantas funcionalidades:

OPCIONES DE VISIBILDAD

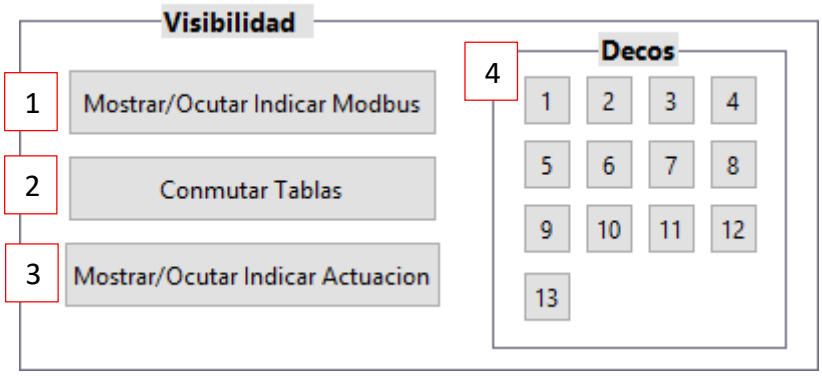

Figura 71: Controles de cambio de visibilidad (Alarmas)

Mediante estos botones los usuarios podrán conmutar la visibilidad de algunos objetos para facilitar la labor de edición:

1. Cambia la visibilidad del indicador de fallo de conexión con el servidor.
2. Conmuta la visibilidad de las tablas histórico duración e histórico de alarmas. (**En el caso de eventos este botón no aparece debido a que no existen estas tablas**)
3. Cambia la visibilidad del indicador de actuación para las alarmas. (**En el caso de eventos este botón no aparece debido a que no existen estas tablas**)
4. Conmuta la visibilidad de los 13 objetos de decoración que existen en el panel frontal.

OPCIONES DE GUARDADO

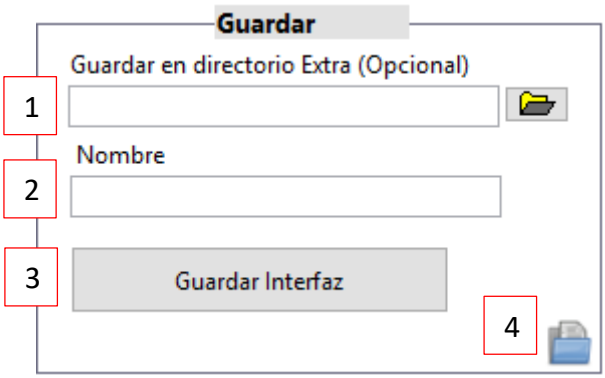

Figura 72: Opciones de guardado

1. Permite elegir un directorio extra para guardar el fichero con la información del panel cuando se hace clic en el botón “Guardar interfaz”.
2. Especifica el nombre con el que se va a guardar el fichero de interfaz gráfica.
3. Guarda en un fichero de configuración toda información gráfica de los objetos del panel frontal perteneciente al panel de Alarmas/Eventos.

4. Abre el directorio raíz del toolkit con los archivos de configuración gráfica de Alarmas/Eventos.

TUTORIAL

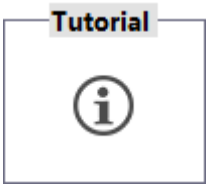

Figura 73: Botón de tutorial

Este último botón accede a un pequeño tutorial explicativo, con algunos consejos de cómo usar esta herramienta. A continuación, se muestra una imagen con la información de este tutorial en el caso de alarmas, ya que en ambos casos es prácticamente idéntico.

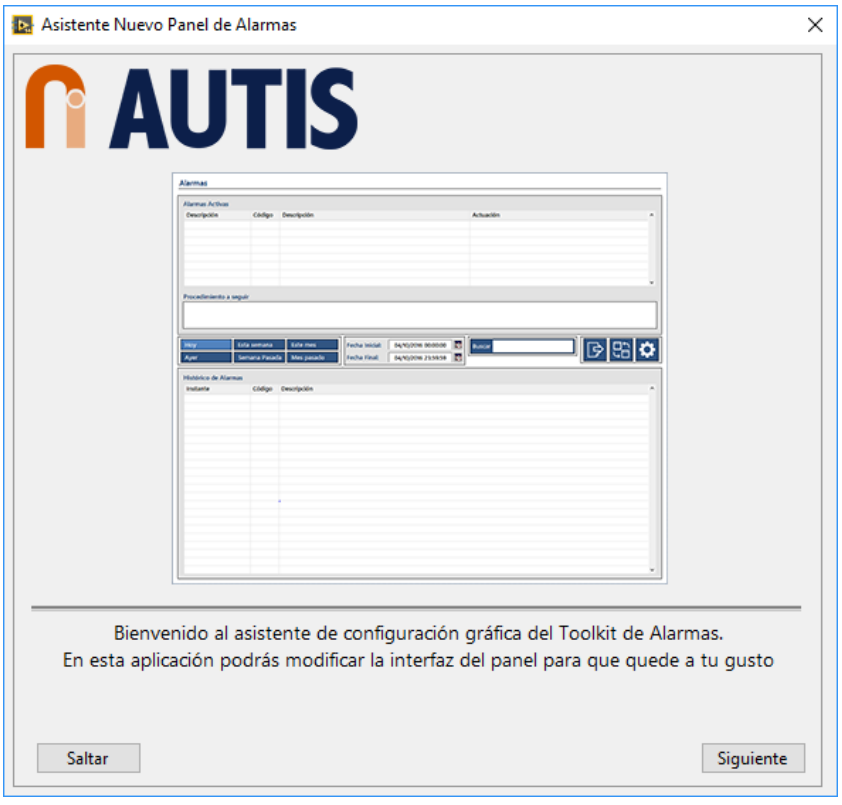

Figura 74: Tutorial, ventana nº 1

Figura 75: Tutorial, ventana nº 2

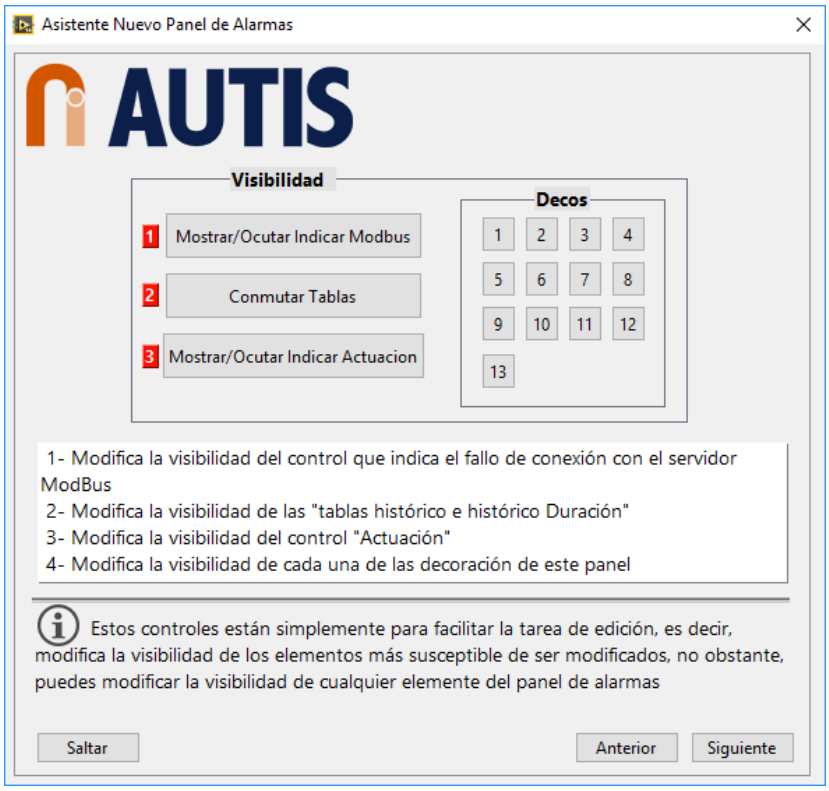

Figura 76: Tutorial, ventana nº 3

Figura 77: Tutorial, ventana nº 4

Figura 78: Tutorial, ventana nº 5

4.3 Ventana de gestión de Paneles

La finalidad de esta ventana es gestionar la creación de paneles y el mantenimiento de paneles existentes en los diferentes proyectos que pueda manejar un programador. Para acceder a ella los usuarios deberán navegar por el menú hasta:

- Tools→ Autis Ingenieros S.L.U. → Alarmas → Ventana de gestión de paneles
- Tools→ Autis Ingenieros S.L.U. → Eventos → Ventana de gestión de paneles

Una vez abierta la herramienta, aparecerá un dialogo con 3 opciones posibles:

Figura 79: Ventana de gestión de paneles

1. AÑADIR PROYECTOS

Esta herramienta está pensada para gestionar varios proyectos al mismo tiempo. Por tanto, antes de modificar o crear un panel, el usuario deberá dar de alta el proyecto sobre el que está trabajando o seleccionarlo en caso de que ya esté añadido.

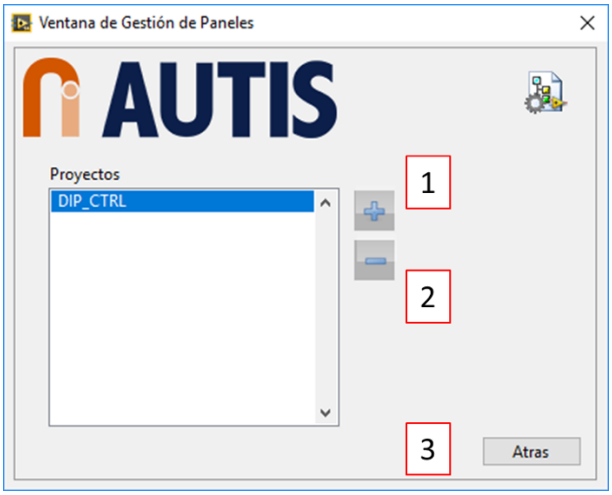

Figura 80: Gestión de proyecto (Ventana de gestión de paneles)

1. **Añadir nuevo proyecto:** al pulsar este botón "+", se abre una ventana de diálogo para añadir el fichero ".lvproj" del nuevo proyecto con el que se pretende trabajar.
2. **Eliminar un proyecto existente:** seleccionar un proyecto existente en la lista y posteriormente clicar en el botón "-".

3. **Seleccionar un proyecto existente:** seleccionamos un proyecto de la lista y posteriormente clicamos en el botón “Atrás” para volver al menú inicial.

2. NUEVO PANEL

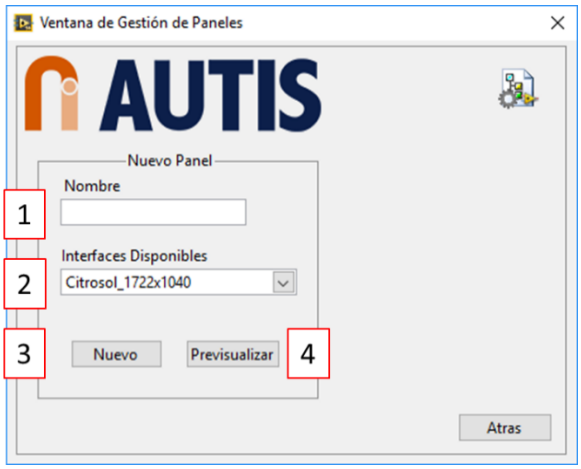

Figura 81: Nuevo panel (Ventana de gestión de paneles)

1. **Nombre:** Se ha de elegir un nombre para el nuevo panel que se va a crear. El nombre del panel es el mismo que posteriormente se va a utilizar en la función Config_SubPanel_Alarms/Events. El nombre es la manera que tiene el toolkit de identificar que archivo de configuración que le corresponde a cada panel. Por ello es importante respetar el nombre en el momento en el que se crea y se carga el panel.
2. **Seleccionar interfaz:** Se ha de elegir qué interfaz va a tener el panel de Alarmas/Eventos. Si anteriormente el usuario ha creado un fichero de “configuración grafica”, esta herramienta reconocerá el fichero (ya que se ha guardado en el directorio raíz del toolkit) y permitirá cargar la interfaz creada.
3. **Crear Panel:** Una vez realizados los pasos anteriores se crea el nuevo panel. Al pulsar el botón nuevo, se abre una ventana de diálogo que permite al usuario guardar el panel en un directorio concreto. Este nuevo panel será una instancia independiente (copia) del panel original pero con la interfaz modificada según el fichero de configuración gráfica.
4. **Pre visualizar:** Antes de crear el panel, se puede pre visualizar la interfaz que se ha seleccionado. En caso de pulsar, se abre el panel de Alarmas/Eventos con la interfaz modificada, según el fichero de configuración gráfica seleccionado.

3. PANEL EXISTENTE

Figura 82: Panel existente (Ventana de gestión de paneles)

Para usar esta ventana, el usuario deberá tener añadido un proyecto y al menos un panel creado en este.

1. **Seleccionar panel:** este desplegable se actualiza con el nombre de todos los paneles que se han creado para el proyecto anteriormente seleccionado.
2. **Configurar:** accedemos a la ventana de configuración de paneles para configurar el panel seleccionado.

4.4 Ventana de configuración de paneles

A esta ventana se puede acceder desde la venta de configuración de paneles o desde el botón de configuración del panel, como ya se ha comentado en el apartado 3.2 Funcionamiento: Elementos comunes. En general, tiene más sentido no incluir este botón de configuración (ocultándolo) ya que muchas de las opciones no están pensadas para ser modificadas mientras se ejecuta la aplicación. No obstante, se deja a decisión del programador el incluir este botón o no.

Tanto para alarmas como para eventos estas ventanas son muy semejantes. Por ello, se procederá a explicar ambas conjuntamente salvo cuando sea necesario puntualizar. A continuación, se muestra una vista general de ambas ventanas.

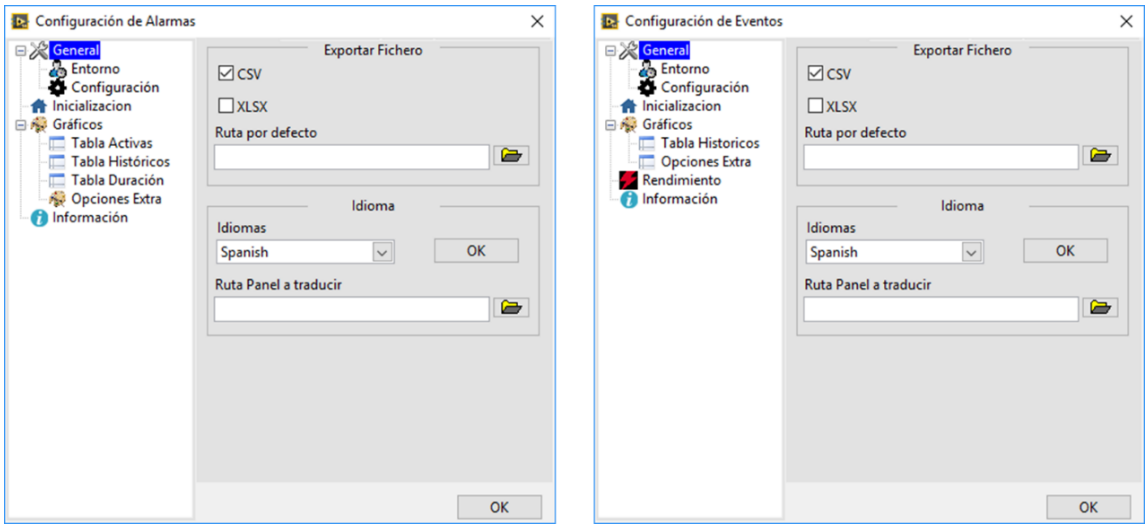

Figura 83: Ventanas de configuración de alarmas y eventos

4.4.1. General: Entorno

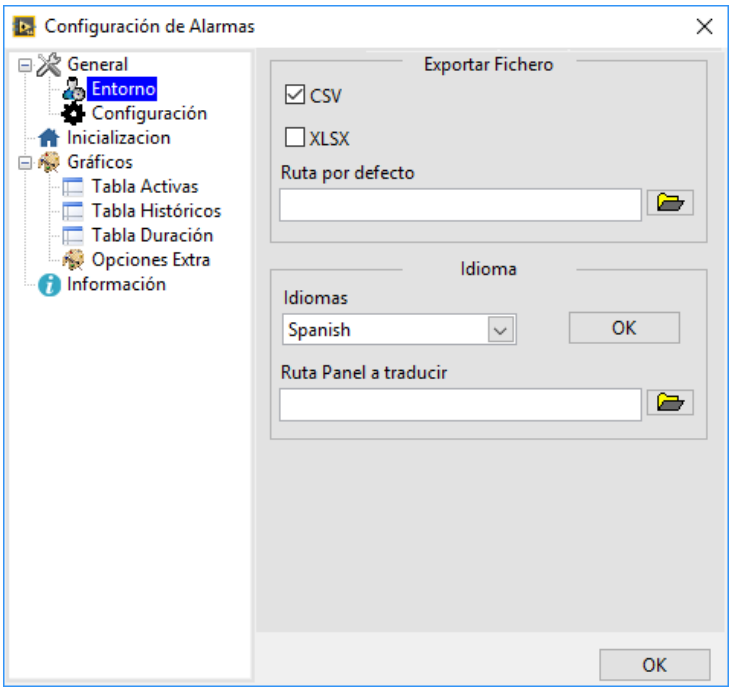

Figura 84: Entorno (Configuración alarmas y eventos)

Esta venta permite:

- **Conmutar el formato del fichero resumen de la tabla de históricos**: mediante los controles “CSV” y “XLSX”.
- Elegir la ruta por defecto que muestra el diálogo que aparece cuando se pulsa el botón exportar sobre los paneles de alarmas y eventos (Figura 84).
- **Traducir un panel:** Para ello el usuario deberá elegir el idioma (inglés o español) y especificar la ruta del VI traducir.

4.4.2 General: Configuración

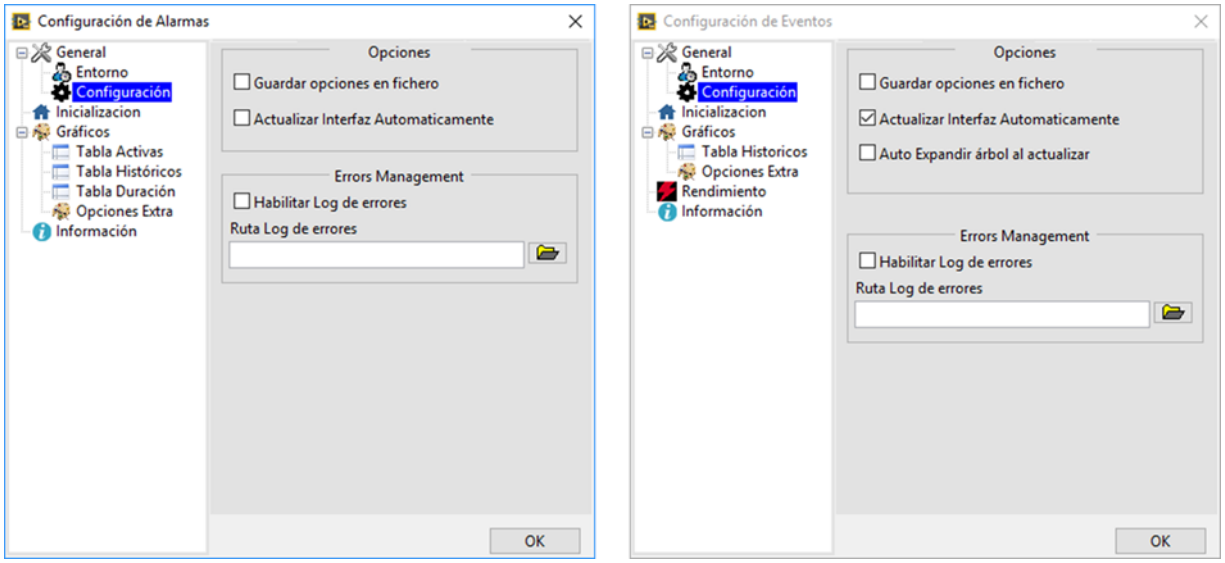

Figura 85: Configuración (Configuración alarmas y eventos)

Esta ventana permite:

- **Restablecer los valores por defecto:** desmarcando la casilla "Guarda opciones en fichero" el archivo de configuración volverá a sus valores iniciales.
- **Actualizar la interfaz automáticamente:** activa una funcionalidad del panel para que el servidor (tanto el local como en remoto) le avise de cuando se ha realizado una modificación de la base de datos y, entonces, actualizarse.
- **Habilitar log de errores:** esta funcionalidad está pensada para la depuración del panel de alarmas y eventos. Habilita un subproceso del mismo que registra todos los puntos del programa donde se han producido errores.
- **Ruta log de errores:** modifica la ruta por defecto de log de errores. Por defecto se usa una ruta relativa al proyecto donde se está utilizando el panel de alarmas y eventos.

EVENTOS

En el caso de eventos, esta ventana tiene una opción adicional:

- **Auto expandir árbol al actualizar:** esta opción, como su nombre indica, expande o no el árbol de eventos cuando se hace un refresco de la interfaz gráfica.

4.4.3 Inicialización

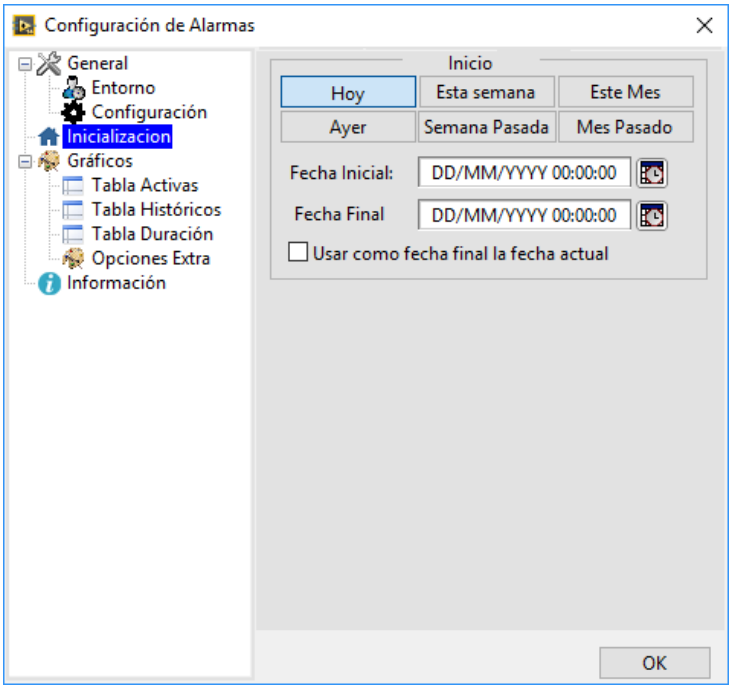

Figura 86: Inicialización (Configuración alarmas y eventos)

Esta ventana permite cambiar el rango de fechas que usar el panel por primera vez para actualizar la interfaz (en el arranque). Permite diferentes opciones:

- **Botones predefinidos**: hoy, esta semana, este mes, ayer, semana pasada, mes pasado.
- **Rango de fechas personalizado:** mediante los controles de fecha inicial y final.
- **Rango de fechas con final variable:** al usar la opción “Usar como fecha inicial la fecha actual” el panel utiliza como fecha inicial la especificada en el control “Fecha inicial” y como fecha final la del día en el que arranca.

4.4.4 Gráficos: Tablas

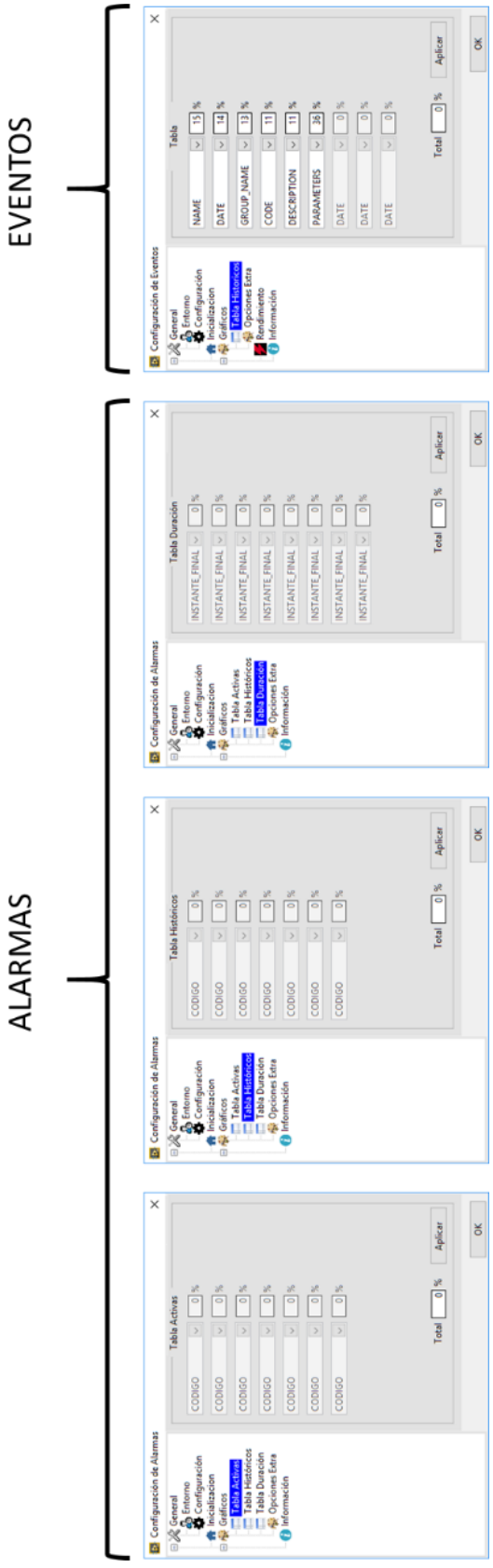

Figura 87: Tablas (Configuración alarmas y eventos)

Por cada tabla presente en los paneles existe un apartado de configuración dentro de la categoría de gráficos, por tanto, hay tres ventanas en el caso de alarmas y una en caso de eventos. Todas funcionan de la misma manera, pero hacen referencia a una tabla en concreto. La función de estas ventanas es que los usuarios puedan elegir:

- **Los campos que parecen en las tablas:** las posibilidades para cada tabla son todos los campos disponibles en las vistas de las bases de datos correspondientes a esas tablas:
    - Tabla de alarmas activas → VIEW_ACTIVE_ALARMAS la cual tiene disponibles los campos: Instante, código, descripción, estado, actuación, grupo e info.
    - Tabla de histórico de alarmas → VIEW_HISTORY_ALARMAS la cual tiene disponibles los campos: Instante, código, descripción, estado, actuación, grupo e info.
    - Tabla de histórico duración → VIEW_HISTORY_DURATION_ALARMAS la cual tiene disponibles los campos: Istante_inicial, Instante_final, duración, estado, descripción, actuación, código, grupo e info.
    - Tabla de histórico de eventos → VIEW_HISTORY_EVENTS la cual tiene disponibles los campos: Name, Date, Group_Name, Code, Color, Event_Type, Icon, Description y Parameters.
- **El orden de los mismos:** para añadir una columna a la tabla, basta con clicar en los desplegables (de arriba a abajo) y elegir el campo deseado, en el orden requerido por el usuario.
- **Porcentaje de ocupación de cada campo sobre el ancho total de la tabla:** cada elemento añadido posee a su derecha un control numérico para especificar un porcentaje de ocupación.

Una vez elegidos todos los campos y aplicados los porcentajes, se ha de pulsar el botón Aplicar. De esta manera, la aplicación comprueba si la suma de todos los porcentajes es el 100%, en caso contrario salta un diálogo indicando que la configuración es incorrecta.

4.4.5 Gráficos: Opciones extra

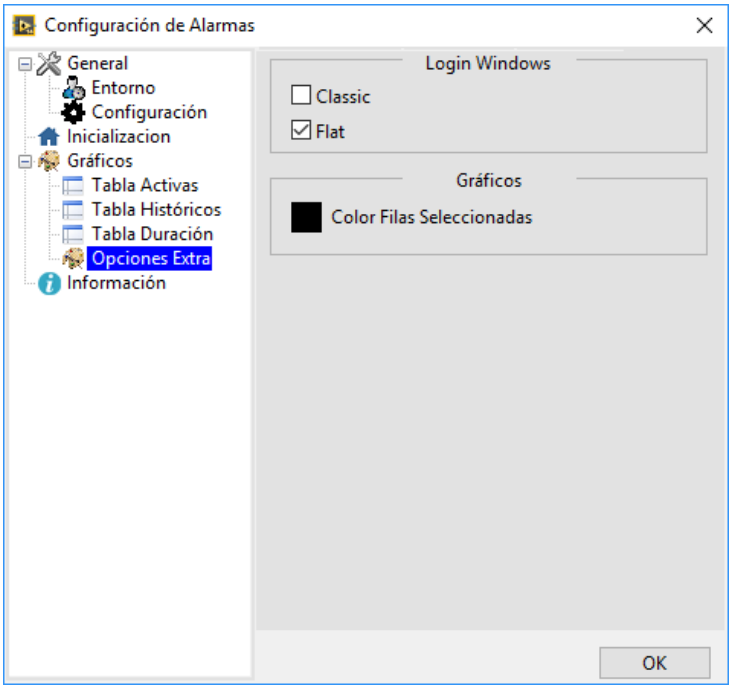

Figura 88: Opciones extra (Configuración alarmas y eventos)

Esta ventana permite:

- **Conmutar entre las "Login Windows" (botones Flat y Classic):** Esta opción solo tiene sentido en el caso de que el usuario haya incluido el botón de configuración en la interfaz del panel. No obstante, **no es lo aconsejable**. Debido a que existen opciones que no han de ser accesibles para cualquier usuario, existe una gestión de privilegios para cuando se intenta acceder a esta ventana desde el panel (no es así si se accede desde el framework), como resultado de esta gestión se muestran estas ventanas de "Login" al pulsar sobre el botón de configuración (Figura 56). Estas ventanas son parte del toolkit de control de usuarios, el cual usa el toolkit de alarmas y eventos.
- **Color Filas seleccionadas:** este control modifica el color de la filas de las tablas (y del árbol de eventos) cuando se seleccionan.

A continuación, se muestra una imagen las ventanas de "login" (Figura 89 y 90):

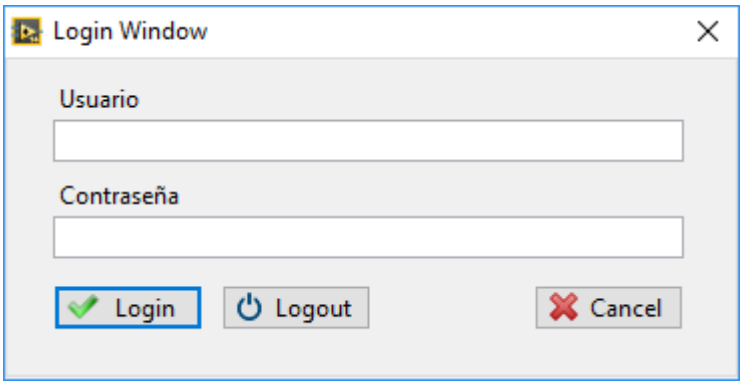

Figura 89: Ventana de identificación (Classic)

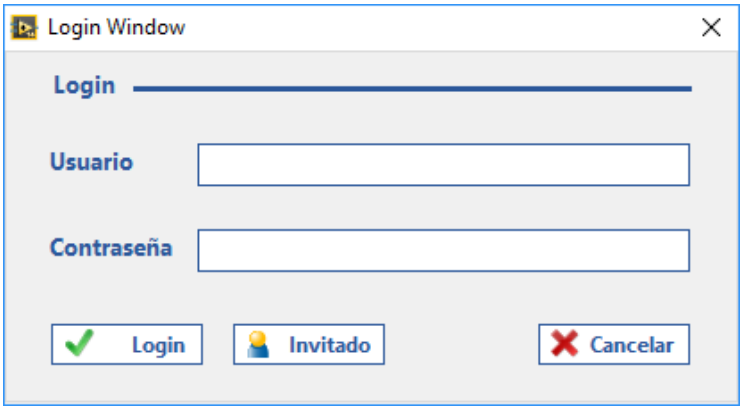

Figura 90: Ventana de identificación (Flat)

4.4.6 Rendimiento

Esta ventana es exclusiva del módulo de eventos. Y se usa para controlar el nivel de renderizado de color en la tabla de histórico de eventos.

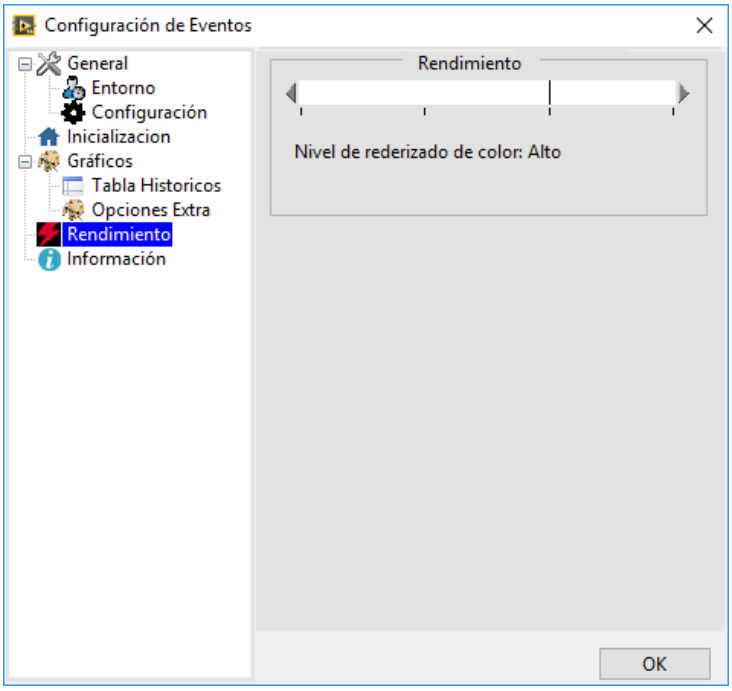

Figura 91: Rendimiento (Configuración alarmas y eventos)

Cuando las tablas se cargan con grandes cantidades de datos (50000 filas o más) el pintado de cada una de ellas suele ser una tarea computacional costosa, sobre todo cuando el programa funciona bajo equipos de bajas especificaciones (en equipos actualizados no es un problema). Es por ello que existen 4 niveles de rendimiento a la hora de pintar las filas:

- Nivel de renderizado 0: No se pinta el color.
- Nivel de renderizado 1: Se pinta el color cuando el Scroll está parado (tabla estática).
- Nivel de renderizado 2: Se pinta el color cuando el Scroll se mueve a bajas velocidades.
- Nivel de renderizado 3: El color se pinta siempre.

4.4.7 Información

Por último, la ventana información, que está reservada para almacenar información de toolkit. Actualmente, únicamente posee el logo de Autis Ingenieros S.L.U.

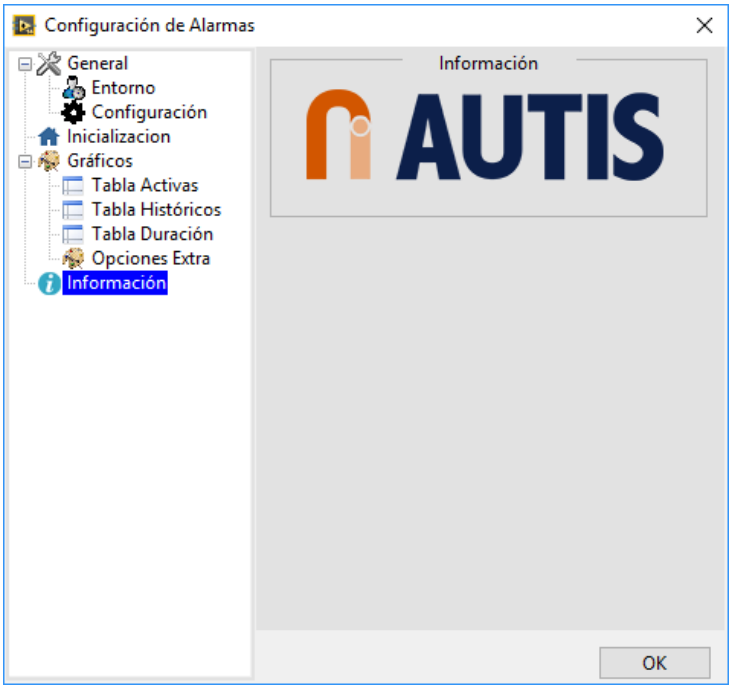

Figura 92: Información (Configuración alarmas y eventos)

4.5 Asistente de configuración de base de datos.

Como bien se ha comentado anteriormente, esta aplicación solo se ha desarrollado para la parte de alarmas.

El objetivo principal de esta es asistir al usuario en la creación de una base de datos de alarmas. Está basada en una serie consecutiva de ventanas autodocumentadas que explican al usuario los elementos de la base de datos y ayudan a introducir la información. Para acceder a esta herramienta, se ha de clicar en:

- Tools→ Autis Ingenieros S.L.U. → Alarmas → Asistente de configuración de la base de datos de alarma.

VENTANA 1:

Figura 93: Ventana 1 (A.C.B.D.A)

Ventana de introducción con algo de información.

VENTANA 2:

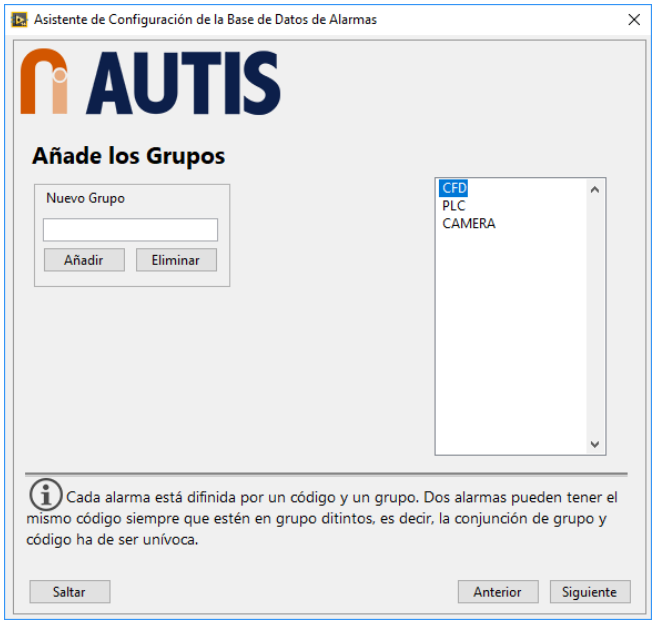

Figura 94: Ventana 2 (A.C.B.D.A)

En esta ventana se deberán añadir todos los grupos que ha de tener la base de datos de alarmas.

VENTANA 3

Figura 95: Ventana 3 (A.C.B.D.A)

En esta ventana se deberá añadir la configuración de todas las alarmas de la base de datos:

- **Código:** número identificador de la alarma.
- **Grupo:** tag identificador de la alarma. En este control deberán aparecen los grupos añadidos en la ventana anterior.
- **Descripción:** campo descriptivo asociada al tipo de alarma.
- **Actuación:** campo descriptivo asociado al procedimiento a seguir cuando salta una alarma.
- **Habilitada Activas:** campo para habilitar la visualización de la alarma en las vistas de alarmas activas.
- **Habilitada Históricos:** campo para habilitar la visualización de la alarma en las vistas de históricos.

En caso de querer eliminar alguna de las alarmas configuradas, se podrá seleccionar de la lista y pulsar el botón Eliminar.

VENTANA 4

Figura 96: Ventana 4 (A.C.B.D.A)

En este punto el usuario ha de rellenar dos campos más y pulsar sobre el botón de guardar (Save):

- **El número máximo de registros de la base de datos**: pasado ese límite, el servidor de alarmas selecciona el exceso de registros más viejos sobre el límite establecido y los copia a un fichero de texto por meses (los meses en los que se han producido esas alarmas).
- **Ruta donde desea guardar la base de datos.**

VENTANA 5

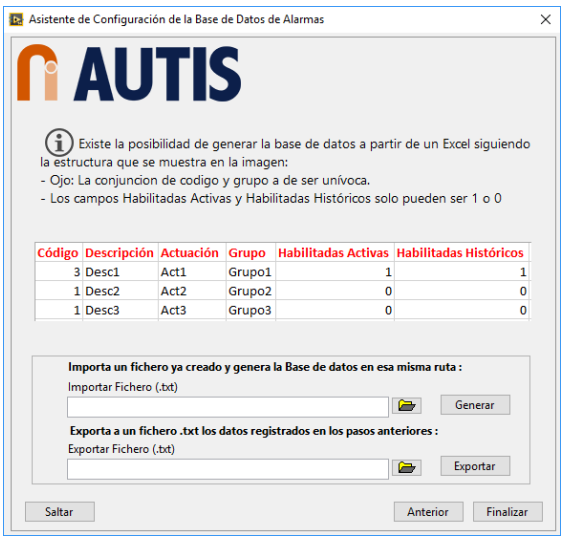

Figura 97: Ventana 5(A.C.B.D.A)

Esta ventana permite al usuario exportar toda la información anteriormente introducida a un fichero de texto delimitado por tabulaciones (mediante el botón exportar).

CODIGO	DESCRIPCION	ACTUACION	GRUPO	HABILITADAS_ACTIVAS	HABILITADAS_HISTORICOS		Limit Lines	50000
1	Connection Failure	Reset CFD	CFD	1	1			
2	Bad Position	Call AIS Support	CFD	1	1			
1	Connection Failure	Check Wires	PLC	1	1			
2	Tunel A Out of Position	Call AIS Support	PLC	1	1			
3	Tunel B Out of Position	Call AIS Support	PLC	1	1			
1	Connection Failure	Reset Camera	CAMERA	1	1			

Figura 98: Ejemplo fichero generado

Del mismo modo que permite generar ficheros, también permite cargar esa misma estructura de ficheros y regenerar la base de datos (mediante el botón generar). La finalidad de esto es poder modificar ese fichero fácilmente mediante algún programa de edición de hojas de cálculo, para posteriormente volver a cargarlo en esta misma ventana y así volver a generar la base de datos en caso de que sea necesario. Otra finalidad es tener toda la configuración de la base en ese fichero para poder generarla vacía sin tener que volver a introducir todos los campos de las ventanas anteriores.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

Desarrollo de una librería en LabVIEW para la implementación de sistemas de gestión distribuidos de alarmas y eventos. Aplicación a un proyecto de visión artificial para la detección de defectos en soldaduras de ejes destinados al sector de la automoción.

Documento Nº 3: Presupuesto

AUTOR: Dotor Alonso, Luis

TUTOR: Herrero Durá, Juan Manuel

Curso Académico: 2016-2017

ÍNDICE

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En el actual documento se procede a desglosar el presupuesto del desarrollo del toolkit que se ha expuesto en los anteriores documentos.

2. CUADRO DE PRECIOS

2.1 Cuadro de precios nº1: Mano de obra

En cuanto a la mano de obra directa, únicamente se cuenta con horas de programación debido a que el toolkit ha sido desarrollado en su totalidad por una persona. El precio se ha obtenido de los salarios del Boletín Oficial del Estado:

- Graduado en Ingeniería en Tecnologías Industriales – Categoría II

CONCEPTOS	**TOTAL (€)**
Salario base	18756,171
Gratificaciones extra (junio, Navidad y Vacaciones)	4220,138
Pluses salariales (Transporte, desgaste herramientas, prendas trabajo)	1200,843
Seguridad social (Base cotización, contingencias comunes, accidentes trabajo, desempleo, fondo garantía salarial y formación profesional)	6.025,32
TOTAL ANUAL	30202,472
A FACTURAR:	
Por Jornada	119,851
Por hora	14,98

Tabla 1: Coste de un graduado en ingeniería en tecnologías industriales

Se estima que el tiempo de programación del toolkit y su instalación en el proyecto Axle Welding ha consistido en 5 meses de trabajo (22 días laborales por mes): 880 horas de trabajo. Por tanto, el coste de la mano de obra total asciende a 13182,4€ como se puede ver en la Tabla 2.

	Precio/hora(€/h)	**Tiempo(h)**	**Total(€)**
Graduado en Ingeniería en Tecnologías Industriales	14,98	880	13182,4
Total	14,98	880	**13182,4**

Tabla 2: Coste total de la mano de obra

2.2 Cuadro de precios nº2: Materiales

En este apartado se precederá a detallar los materiales utilizados para la realización del proyecto. Debido a que se trata de un desarrollo Software, únicamente se cuenta con 3 materiales: PC, Licencia de LabVIEW, Licencia de Windows. Todos ellos representan un coste de amortización en lo que a este proyecto se refiere, lo cual se ha calculado como:

$$Coster_{Amortización}\left(\frac{€}{h}\right) = \frac{Coste_{Total}}{Tiempo_{Amortización}}$$

PRODUCTO	PRECIO (€)
PC: HP ProBook 650 G2	967,80
Licencia de Windows	43
Licencia de LabVIEW Completo	3605

Tabla 3: Coste materiales

Se considera un periodo de amortización de 2 años, que corresponden con 3872 horas laborales. Teniendo en cuenta la formula anterior, se obtiene la tabla siguiente.

Descripción del elemento	Precio (€/h)
PC: HP ProBook 650 G2	0,2499
Licencia de Windows	0,0111
Licencia de LabVIEW Base	0,931

Tabla 4: Gastos de software

A continuación, en la Tabla 4 se presenta el coste total del material.

PRODUCTO	Precio unitario	Unidades	Total (€)
PC: HP ProBook 650 G2	0,2499	880	219,912
Licencia de Windows	0,0111	880	9,768
Licencia de LabVIEW Base	0,931	880	819,31
		TOTAL	**1048,99**

Tabla 5: Coste total de material

2.3 Cuadro de precios nº3: Precios unitarios

A continuación, se presenta el importe de las unidades de obra en las que se ha dividido el proyecto. Debido a que todas ellas consumen fundamentalmente horas de programador, y las licencias asociadas, su importe se ha obtenido a partir del porcentaje de tiempo de cada unidad de obra sobre el total, el coste de la hora de un programador y el coste por hora del material:

$$Coste\ unidad\ de\ obra = \frac{\%Tiempo_{ud}}{100} * Horas_{Totales} * (CosteHora_{Programador} + CosteHora_{Materiales})$$

1. Estudio de necesidades y diseño de la arquitectura: 40% de las horas totales.
2. Programación y depuración del código: 50% de las horas totales.
3. Instalación en Axel Welding: 2% de las horas totales.
4. Redacción y documentación del proyecto: 8%

Nº de Orden	Descripción	Medición	Precio (€/ud)	Importe(€)
1	Estudio de necesidades y diseño de la arquitectura.	1	3163,776	5863,32
2	Programación y depuración del código	1	4587,47	7329,15
3	Instalación en Axel Welding	1	158.188	293,17
4	Redacción y documentación del proyecto	1	1448,79	1172,66

Tabla 6: Precios unitarios

2.4 Cuadro de precios nº4: Precios descompuestos

En este apartado se detallan los precios de las unidades de obra de manera más desglosada en sus distintos componentes. Del mismo modo que en el apartado anterior, el precio de cada componente de unidad de obra se ha calculado en base a las horas totales:

$$Coste\ Componente\ de\ u.d = \frac{\%Tiempo_{ud}}{100} * \frac{\%Tiempo_{Componente}}{100} * Horas_{Totales} * (CosteHora_{Programador} + CosteHora_{Materiales})$$

1. Estudio de necesidades y diseño de la arquitectura:
    - Estudio y diseño de bases de datos: 20% de las horas de diseño.
    - Estudio y diseño de arquitectura servidor: 40% de las horas de diseño.
    - Estudio y diseño del HMI: 20% de las horas de diseño.
    - Estudio y diseño del framework: 20% de las horas de diseño.
2. Programación y depuración del código:
    - Programación de las bases de datos: 10% de las horas de programación.
    - Programación del servidor: 20% de las horas de programación.
    - Programación del HMI: 40% de las horas de programación.
    - Programación del framework: 30% de las horas de programación.
3. Instalación en Axel Welding: 2% de las horas totales.
    - Planteamiento de la estructura de integración: 40% de las horas de instalación.
    - Programación con el toolkit y revisión de fallos: 60% de las horas de instalación.
4. Redacción y documentación del proyecto.
    - Redacción de la memoria: 50% del tiempo de documentación.
    - Redacción del manual de usuario: 40% del tiempo de documentación.
    - Redacción del presupuesto: 10% del tiempo de documentación.

1	Estudio de necesidades y diseño de la arquitectura				
1.1	**Unidad**	**Descripción**	**Medición**	**Precio (€/h)**	**Importe (€)**
	h	Estudio y diseño de bases de datos	70,4	19,98	1138,51
	h	Estudio y diseño de arquitectura servidor	140,8	19,98	2277,02
	h	Estudio y diseño del HMI	70,4	19,98	1138,51
	h	Estudio y diseño del framework	70,4	19,98	1138,51
		Costes indirectos	0,03(%)	5692,54	170,78
				TOTAL	5863,32

Tabla 7: Precio descompuesto unidad de obra 1

2	Programación y depuración del código				
2.1	**Unidad**	**Descripción**	**Medición**	**Precio (€/h)**	**Importe (€)**
	h	Programación de las bases de datos	44	19,98	711,57
	h	Programación del servidor	88	19,98	1423,14
	h	Programación del HMI	176	19,98	2846,27
	h	Programación del framework	132	19,98	2134,70
		Costes indirectos	0,03(%)	7115,68	213,47
				TOTAL	7329,15

Tabla 8: Precio descompuesto unidad de obra 2

3	Instalación en Axel Welding				
3.1	**Unidad**	**Descripción**	**Medición**	**Precio (€/h)**	**Importe (€)**
	h	Planteamiento de la estructura de integración	7,04	19,98	113,85
	h	Programación con el toolkit y revisión de fallos	10,56	19,98	170,78
		Costes indirectos	0,03(%)	284,62	8,54
				TOTAL	293,17

Tabla 9: Precio descompuesto unidad de obra 3

4	Redacción y documentación del proyecto				
4.1	**Unidad**	**Descripción**	**Medición**	**Precio (€/h)**	**Importe (€)**
	h	Redacción de la memoria	35,2	19,98	569,25
	h	Redacción del manual de usuario	28,16	19,98	455,40
	h	Redacción del presupuesto	7,04	19,98	113,85
		Costes indirectos	0,03(%)	1138,5088	34,16
				TOTAL	1172,66

Tabla 10: Precio descompuesto unidad de obra 4

3. PRESUPUESTO DE EJECUCIÓN MATERIAL

Tras el desglose anteriormente realizado, se procede a calcular el presupuesto total de ejecución material, el cual refleja el coste directo del trabajo.

Nº de Orden	Descripción	Importe (€)
1	Estudio de necesidades y diseño de la arquitectura	5863,32
2	Programación y depuración del código	7329,15
3	Instalación en Axel Welding	293,17
	Redacción y documentación del proyecto	1172,66
Presupuesto total de ejecución de material		**13485,64**

Tabla 11: Presupuesto total de ejecución material

4. PRESUPUESTO TOTAL DE EJECUCIÓN POR CONTRATA

El presupuesto de ejecución por contrata se ha calculado añadiendo al presupuesto de ejecución material los gastos generales, que incluyen otros aspectos como la tramitación licencias, documentación, estudios, etc. y el beneficio industrial.

Descripción	Importe (€)
Presupuesto total de ejecución de material	13485,64
Gastos Generales (13%)	1753,13
Beneficio industrial (6%)	809,13
Presupuesto de Ejecución por Contrata	**16047,91**

Tabla 12: Presupuesto total de ejecución por contrata

5. PRESUPUESTO BASE DE LICITACIÓN

Por último, el presupuesto base de licitación el cual se obtiene añadiendo el I.V.A al presupuesto de ejecución por contrata.

Descripción	Importe (€)
Presupuesto de Ejecución por Contrata	16047,91
I.V.A. (21%)	3370,06062
PRESUPUESTO BASE DE LICITACIÓN	**19417,97**

Tabla 13: Presupuesto base de licitación