Universidad de las Ciencias Informáticas

Facultad 4

Título: *Paleta de componentes gráficos para la representación de estaciones de vaporización en un central azucarero desde el sistema AREX*

Trabajo de Diploma presentado en opción por el título de Ingeniero en Ciencias Informáticas

Autor: Karla Martínez Carrera

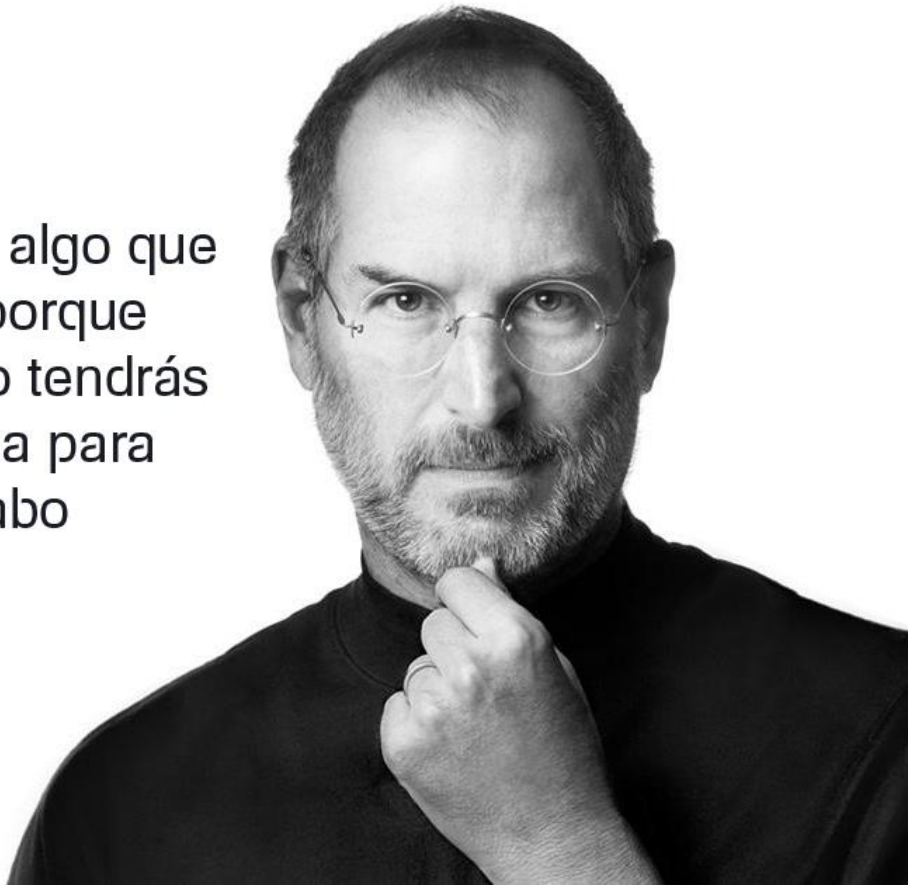
Tutores:

MSc. Orlando Grabiel Toledano López

Ing. Diana Tahirí Galí Ramírez

Ing. Raúl Enrique Alamo Gihón

Tienes que hacer algo que
te apasiones, porque
de lo contrario, no tendrás
la perseverancia para
llevarlo a cabo



Declaración de autoría

Declaro ser autor del presente trabajo y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas de hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

Karla Martínez Carrera

_____

Firma del autor

MSc. Orlando Grabiel Toledano López

_____

Firma del tutor

Ing. Diana Tahirí Galí Ramírez

_____

Firma del tutor

Ing. Raúl Enrique Alamo Gihón

_____

Firma del tutor

Dedicatoria

Dedico este trabajo de diploma a la mujer más fuerte que conozco, a mi madre por enseñarme que, aunque te encuentres en las circunstancias más difíciles, al final todo va a estar bien. Y a mi padre por luchar tanto para que hoy, yo pudiera estar aquí.

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia por apoyarme en todo momento, pero en especial a mi hermano Ariel por ser mi gran ejemplo y por enseñarme que el camino del estudio y la preparación al final nos llevara al éxito. A mi novio Elias por estar a mi lado siempre y por quererme tanto. Agradecer a mis tutores Orlando, Diana y Raúl por todo su apoyo en estos ocho meses de continuo trabajo, pero para Orlando mi agradecimiento es doble por aceptar ser mi tutor y por ser un gran amigo. Al tribunal por todas sus sugerencias, recomendaciones y tiempo invertido para lograr una investigación con calidad. A todos los profesores, de los cuales recibí las mejores clases y los mejores consejos, en especial a Yanelis Benites profesora de cálculo de la FICI. A los ingenieros del Centro de Informática Industrial por sus aportes a la investigación, de igual forma al grupo empresarial AZCUBA y grupo EROS. Hoy no puedo dejar de agradecer a esos amigos que me acompañaron desde la infancia Jennifer Aput Guerra muchas gracias por estar hoy aquí. A mis compañeros de aula y a los grandes amigos que he encontrado en la universidad. Y por último me encantaría agradecer a los mejores ingenieros que graduara la Universidad de las Ciencias Informáticas Melissa, María Elena, Daniel Miguel, Lysbet, Jorge y Marcos Antonio por acompañarme desde el primer día, por su amistad incondicional, porque a pesar de que el camino ha sido largo y lleno de dificultades, me llevo los mejores recuerdos de estos cinco años de carrera, les deseo éxitos porque el camino más difícil comienza ahora. Muchas gracias.

Resumen

Resumen:

El desarrollo creciente de las tecnologías ha permitido el reemplazo de sistemas analógicos o manuales por sistemas automatizados digitalmente, trayendo consigo disímiles ventajas, como el aumento de los niveles de producción. Por su parte, el Centro de Informática Industrial perteneciente a la Facultad 4 de la Universidad de las Ciencias Informáticas, desarrolla el Sistema de Medición y Adquisición de Datos AREX con el objetivo de ser desplegado en las industrias. Este cuenta con una paleta de componentes dirigida a algunos sectores industriales. El sistema en su estado actual no cuenta con los componentes gráficos que representan la estructura/edificación de un central de la industria azucarera. No permite la visualización de datos obtenidos de los procesos que ocurren en dicha industria por lo que en este momento no podría ser aplicado en este tipo de entornos. Para dar solución a esta problemática se ha desarrollado una paleta de componentes que representa una estación de evaporación de un central azucarero. La propuesta de solución fue desarrollada mediante la metodología AUP-UCI en su escenario cuatro: historia de usuario. Se emplea además el *framework* de desarrollo *Qt* y como lenguaje de etiquetado QML. Se realizaron los diagramas necesarios para el modelado, implementación y prueba, obteniéndose una paleta de componentes que cumple con los requerimientos funcionales y no funcionales. Con esta se permite la representación gráfica de los componentes que representan una estación de evaporación de un central de la industria azucarera.

Palabras clave: estación de evaporación, componente gráfico, industria, paleta de componentes.

Índice

Índice.

Introducción 1
Capítulo 1. Fundamentación teórica 5
1.1 Estudio del arte de la industria azucarera en Cuba 5
1.2 Proceso de producción de azúcar 6
1.3 Estación de vaporización 8
1.5 Descripción de las tecnologías, herramientas y metodologías a utilizar 14
1.5.1 Metodología de desarrollo 14
1.5.2 Herramientas de modelado 17
1.5.3 Lenguaje de programación 17
1.5.4 Framework de programación 18
1.5.5 Entorno de desarrollo 18
1.5.6 Herramienta para dibujo vectorial 18
1.5.7 Patrón de arquitectura 19
1.5.8 Arquitectura modular 19
1.5.9 Conclusiones parciales 20
Capítulo 2. Propuesta de solución 22
2.1 Modelo de Dominio 22
2.2 Captura de requisitos 23
2.2.1 Requisitos funcionales 24
2.2.2 Requisitos no funcionales 25
2.3 Historia de usuario 26
2.4 Diagrama de paquetes 29
2.1 Patrones de diseño 29
2.1.1 Patrones GRASP 30
2.6 Conclusiones parciales 31
Capítulo 3. Implementación y prueba 31
3.1 Modelo de implementación 31
3.1.1 Diagrama de componente 31
3.1.1 Diagrama de despliegue 32

Índice

3.3 Estándar de codificación 33
3.3 Solución del problema 34
3.4 Pruebas 35
3.4.1 Aplicación de las pruebas de caja negra 37
3.5 Conclusiones parciales 41
Conclusiones generales 42
Glosario de términos 48

Introducción

Desde la antigüedad el hombre ha buscado la forma de facilitar los trabajos que implican grandes esfuerzos físicos. A partir de la revolución industrial se han creado máquinas que remplazan el trabajo forzado por operaciones más sencillas en aras de aumentar la calidad y eficiencia de las industrias. Con la revolución tecnológica se han logrado automatizar gran parte de los procesos industriales, trayendo consigo disímiles ventajas, como el aumento de los niveles de producción y mejora en la calidad de los productos (Mónica Mulet-Hing, 2016). Cuba no ha quedado exenta de estos adelantos, ejecutando grandes esfuerzos para incrementar la eficiencia y la capacidad del sector industrial. Esto se puede constatar en industrias y fábricas pertenecientes a las distintas ramas de la economía, entre ellas: los centrales azucareros que juegan un papel importante en la misma.

La automatización de procesos industriales se logra a través de los sistemas de supervisión, control y adquisición de datos SCADA (por sus siglas en inglés: *Supervisory Control And Data Adquisition*). Un SCADA es un *software* especialmente diseñado para funcionar sobre operadores de producción proporcionando comunicación con los dispositivos de campo (autómatas programables, sensores y actuadores) (Zhang, Wang, Xiang, & Chee-Wooi, 2016). Este sistema al ser desplegado en la industria, es capaz de tener control sobre las variables que intervienen en los distintos procesos de producción desde la pantalla de un ordenador (Pérez-López, 2015). Proporciona además, información del proceso a los operadores, supervisores de control de calidad, operadores de mantenimiento, entre otros (Lozano & Romero, 2013).

La utilización de SCADA en la industria aumenta la calidad del producto fabricado, mejora los sistemas de seguridad del proceso industrial y ejecuta operaciones cuya realización sería más compleja con la única participación del hombre (Pérez-López, 2015), de ahí la importancia de desplegar este tipo de *software* en el sector azucarero. En la actualidad los centrales azucareros cubanos cuentan con un sistema de supervisión y control de procesos también conocido con el nombre comercial EROS, este se encuentra desplegado en todos los centrales del país. Constituye el pilar fundamental del trabajo en los centrales, por lo cual es de gran importancia para esta industria. Además, es el encargado de la supervisión y control de los procesos de producción de azúcar y derivados de la caña. Debido al paso del tiempo y continuos procesos de mantenimientos, en estos momentos presenta algunas deficiencias, tales como: obsolescencia

tecnológica, problemas de visualización de componentes que intervienen en el proceso productivo y no es multiplataforma (Grupo Eros, 2014).

En la Universidad de las Ciencias Informáticas (UCI) desde su fundación, se han creado diversas soluciones de gran utilidad para la industria cubana. En la actualidad, el Centro de Informática Industrial (CEDIN) tiene como una de sus líneas de investigación: Los Sistemas Embebidos, perteneciente al departamento de desarrollo de aplicaciones. En este se desarrolla el proyecto Sistema de Medición y Adquisición de Datos AREX, el cual permite medir y adquirir datos de procesos de baja y mediana complejidad. Este *software* puede ser ejecutado en *hardware* económico y de bajas prestaciones, tales como: tarjetas *Raspberry Pi* (Duverger, Vargas, & Naranjo, 2017). Además, es un sistema multiplataforma, que cuenta con un conjunto de componentes gráficos divididos en categorías, que permiten representar en la pantalla de un ordenador procesos a distancia. Es de interés del centro extender estas categorías hacía otros sectores industriales, en particular al sector azucarero con el objetivo de incrementar su eficiencia y radio de acción (Tahiri, Ramirez, Alberto, & Durán, 2019).

De acuerdo con la visión que tiene el centro, AREX en un futuro evolucionará completamente hacia un sistema SCADA en cuanto a la supervisión y control, pues en la actualidad solo realiza la adquisición de datos.

El sistema en su estado actual, no permite la representación gráfica de componentes que conforman la instrumentación de un central de la industria azucarera. Específicamente en una estación de vaporización, la cual es la encargada de concentrar el jugo de caña para la producción de meladura (Pérez, 2016), por lo que AREX no podría aplicarse a este tipo de entorno.

Lo que conduce a enunciar el siguiente **problema de investigación:** ¿Cómo representar el estado de la instrumentación en una estación de vaporización de un central azucarero desde el sistema AREX?

El diseño de la investigación permite declarar como **objeto de estudio** el proceso de representación de la instrumentación en una estación de vaporización de un central azucarero. Enmarcando el **campo de acción** la representación de la instrumentación de una estación de vaporización de un central azucarero desde el sistema AREX.

Introducción

A partir del problema de investigación se define como **objetivo general** desarrollar una paleta de componentes gráficos para representar el estado de la instrumentación en una estación de vaporización de un central azucarero, desde el sistema AREX.

Para el cumplimiento del objetivo general se plantean los siguientes objetivos específicos:

- Analizar los principales fundamentos teóricos y tecnológicos relacionados con los procesos existentes en la industria azucarera.
- Analizar las principales tendencias actuales en el desarrollo de componentes gráficos para la representación de procesos.
- Seleccionar las tecnologías, herramientas y técnicas para el desarrollo de la propuesta de solución.
- Implementar la paleta de componentes para la representación de la instrumentación existente en una estación de vaporización de un central azucarero.

Para el desarrollo de la investigación se emplearon los siguientes métodos de investigación científica.

Métodos teóricos:

Analítico-sintético: El uso del mismo permitió analizar el proceso de vaporación de caña para identificar los elementos que lo componen.

Modelación: Fue utilizado para realizar los diagramas necesarios que modelan la solución. Como por ejemplo en la definición del modelo de dominio, diagrama de paquetes entre otros artefactos generados durante la investigación.

Métodos empíricos:

Observación: Se utilizó para obtener información de las necesidades existentes a lo largo del desarrollo del proyecto.

Entrevistas: Se utilizó para obtener opiniones y criterios de los especialistas que intervienen en el desarrollo del sistema AREX y los usuarios que utilizan el sistema EROS.

La siguiente investigación está conformada por tres capítulos:

Introducción

Capítulo 1. Fundamentación teórica: En este capítulo se exponen los elementos teóricos utilizados en la investigación. Se analizan las características fundamentales de los componentes gráficos que formarán parte de la propuesta de solución. Se describen, las herramientas, tecnologías, lenguaje de programación y metodologías utilizadas en el desarrollo de la propuesta de solución.

Capítulo 2. Propuesta de solución: En este capítulo se exponen los elementos que describen la propuesta de solución tales como: Modelo de dominio, diagrama de paquete, requisitos funcionales y no funcionales y las historias de usuario para una mejor comprensión de los requisitos.

Capítulo 3. Implementación y prueba: En este capítulo se realiza el modelado de las funcionalidades y la implementación de la propuesta de solución donde se obtiene una paleta de componentes que permite la representación gráfica de una estación de vaporización. Se describen además las pruebas realizadas al sistema, para asegurarse que este cumpla con las especificaciones requeridas.

Capítulo 1. Fundamentación teórica

En este capítulo se presentan los elementos teóricos y conceptuales referentes al problema a resolver, para ello se explican los procesos que ocurren en una industria azucarera, haciendo énfasis en el proceso de producción de azúcar. Se expone la instrumentación y control de una estación de vaporización en la etapa perteneciente al proceso de producción de azúcar. Se realiza una investigación de sistemas similares utilizados para la medición y adquisición de datos en procesos de producción de azúcar. También se describen las diferentes herramientas de *software*, metodologías y lenguaje de programación que se emplean en el proceso de diseño y desarrollo de la solución planteada.

1.1 Estudio del arte de la industria azucarera en Cuba

Desde que se introdujo la caña de azúcar en Cuba, el sector agroindustrial azucarero es sin lugar a dudas, uno de los más importantes de la producción cubana de todos los tiempos (García, 2011). Fue la fuente principal de ingresos en divisas durante muchísimos años, y la garantía para la adquisición de créditos externos (González, 2017).

AZCUBA es el grupo empresarial encargado de rectorar las entidades que integran la agroindustria azucarera para el desarrollo de la producción de azúcares, energía eléctrica, derivados de la caña de azúcar y alimentos para consumo humano y agropecuario. Dentro del grupo empresarial AZCUBA existen diversas empresas subordinadas, tales como TECNOAZUCAR, TRANSMEC, ZERUS, ZETI, ARCAZ, AZUIMPORT, AZUMAT, AZUTECNIA, ESAZUCAR, ICIDCA, IPROYAZ, ATAC, INICA, CNCA, las cuales atienden distintas áreas para la producción y comercialización del azúcar. AZCUBA administra 56 centrales azucareros y casi 100 unidades para la producción de azúcar, electricidad, alcohol, ron, levadura, tableros, alimento animal, sorbitol y CO2 fundamentalmente; así como varios talleres mecánicos para producir y reparar equipos industriales y agrícolas (Azcuba, 2018).

Como fue mencionado anteriormente, en la industria azucarera se utiliza el sistema EROS para la supervisión, control y adquisición de datos de procesos industriales y la automatización del proceso de producción de azúcar. La primera versión fue desarrollada en el año 1993, con tecnología ya obsoleta, y en estos momentos con más de 20 años de explotación. Este presenta problemas con la visualización de los procesos de producción, porque los componentes que brinda el sistema no son visualmente atractivos

desde el punto de vista del diseño para los usuarios. Cuando es necesario realizar un cambio en un estado de alarma el sistema necesita consumir una imagen externa en formato JPG realizada por otra aplicación lo cual implica un aumento en la duración de las tareas. Además, solo puede ser utilizado en el sistema operativo Windows (Grupo Eros, 2014) y teniendo en cuenta la nueva política de gobernabilidad tecnológica, su uso comenzaría a verse limitado en nuestro país.

Luego de analizar las principales características del sistema EROS se pudo determinar que en su estado actual no funciona de manera eficiente.

1.2 Proceso de producción de azúcar

Desde la era industrial la automatización se ha convertido en una herramienta indispensable para aumentar la calidad de sus productos, reducir los tiempos de producción, ayudar a la protección del medio ambiente, realizar tareas complejas, reducir los desperdicios y especialmente aumentar la rentabilidad (Pérez, 2016).

En un central de la industria azucarera no solo se produce azúcar parda y azúcar refino sino también, azúcar ecológica (Orgánica), alcohol, ron, aguardiente, licores, energía eléctrica entre otros productos de extrema calidad. Para la fabricación de estos productos es necesario llevar a cabo largos procesos, entre los que se encuentra el proceso de producción de azúcar el cual está formado por distintas etapas, explicadas a continuación (Azcuba, 2018).

El proceso de producción de azúcar está compuesto por etapas, la primera de estas es la cosecha, donde se corta la caña y es enviada al ingenio. Luego comienza el triturado de caña, se lleva a cabo en las picadoras que están compuestas por cuchillas giratorias que cortan los tallos y los convierten en astilla dando un tamaño uniforme. La caña previamente preparada en las picadoras llega a los molinos donde se extrae el máximo jugo y recolectado en los tanques de almacenamiento. En la etapa de filtración se calienta el jugo a una temperatura cercana a los 102º C a 105º C y se almacena en tanques de clarificación. El jugo clarificado obtenido se filtra para eliminar partículas y se envía a los evaporadores (Enrique Baeyens Lázaro, 2011). La etapa de vaporación es realizada en una estación de vaporización, de la cual se explicará su funcionamiento con mayor profundidad en el próximo acápite. La etapa siguiente es clarificación de la meladura, en la cual la meladura obtenida de la estación de evaporación es sometida a una segunda clarificación por flotación con ácido fosfórico, floculante, cal y aire; para de esta forma separarle la espuma

que contiene los sólidos no azúcares que no fue posible extraer en la primera clarificación de jugo alcalizado (Enrique Baeyens Lázaro, 2011). Luego en la etapa de cristalización, la sacarosa contenida en la meladura es cristalizada por medio de evaporadores de simple efecto o tachos, de tal forma que la meladura es llevada hasta la zona meta de sobresaturación. La última etapa es la de secado que tiene el objetivo de secar la azúcar húmeda obtenida en la etapa anterior para ser embazada (Guzman, 2006). La siguiente imagen muestra un resumen del proceso de producción de azúcar:

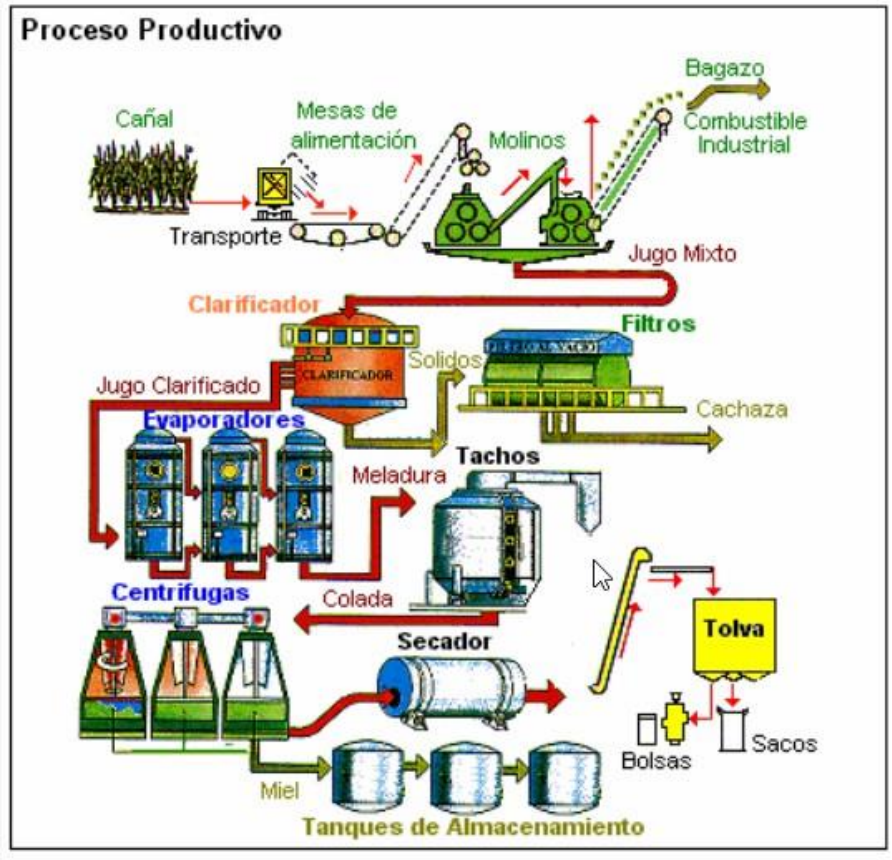


Figura 1. Proceso de producción de azúcar (Guzman, 2006)

Después de caracterizar el funcionamiento del proceso de producción de azúcar fueron identificadas otras etapas, en las que, para futuras versiones, sería posible incorporar componentes para la medición y adquisición de variables involucradas en cada proceso desde el sistema AREX, tales como: temperatura, presión, PH, densidad y concentración de ciertas sustancias.

1.3 Estación de vaporización

Las estaciones de vaporización están presentes en diferentes industrias, como la química, alimentaria, metalúrgica entre otras. Estos equipos representan el subproceso tecnológico de mayor importancia, determinando en gran medida la calidad de la producción. Su funcionamiento exige grandes gastos de energía (Pérez, 2016).

Los evaporadores se clasifican en Evaporadores Simple Efecto y Evaporadores Múltiple Efecto. El Evaporador Simple Efecto está formado por un solo vaso, el vapor producido como resultado de la ebullición de la solución que se concentra, se envía a un condensador y no se aprovecha para continuar concentrando la solución en otros vasos (Enrique Baeyens Lázaro, 2011). Los Evaporadores Múltiple Efecto están formados por varios vasos acoplados en cascada como se muestra en la figura 1; el vapor producido como resultado de la ebullición de la solución que se concentra se aprovecha para continuar concentrando la solución en otros vasos (Enrique Baeyens Lázaro, 2011) (Sharan & Bandyopadhyay, 2017).

La estación de vaporización forma parte del proceso de producción de azúcar, esta etapa se realiza para la concentración del jugo de caña y producir meladura. Su principal objetivo consiste en aumentar la concentración de la solución, alimentada mediante la evaporación del solvente de dicha solución. En la industria azucarera los dos tipos de evaporadores en un primer momento el evaporador múltiple efecto porque es necesario pasar el jugo de caña por varios evaporadores para obtener un producto altamente concentrado y luego el simple efecto en caso de que la concentración no sea la idónea (Pérez, 2016). Durante el proceso de producción de azúcar se supervisan diferentes variables, en el caso de la etapa de vaporización se involucran las variables de nivel, temperatura, presión y concentración.

Esta etapa comienza al verter la solución en tanques de almacenamiento, para ser impulsados al primer vaso por una bomba centrífuga. La solución pasa por una válvula (*on-off*) conectada a un controlador lógico programable (PLC[1], por sus siglas en inglés) (Hudedmani, Kabberalli, & Hittalamani, 2017) (Bolton, 2015), del que están conectados los sensores de nivel (LT) que se encuentran en los vasos. Por lo tanto, si el nivel

[1] PLC: Unidad central para el control de procesos industriales que puede ser programada de forma efectiva y segura asegurando el cumplimiento de objetivos requeridos.

del primer vaso marca el máximo se cierra la válvula; en cambio sí el nivel marca el mínimo, la válvula se abre. El vapor producido en el primer vaso se conduce al próximo vaso por un conducto en el que se encuentran los sensores de presión (PT) conectados al PLC, pasando por una válvula conectada también al PLC (Marino, 2018). Si los sensores marcan la presión máxima y la concentración es la correcta entonces la válvula se cierra, en caso contrario la válvula se abre. Al salir la solución del primer vaso antes de pasar al segundo, pasa por un conducto que contiene una válvula; esta se abre en caso de que el nivel del próximo vaso no marque el máximo, en caso contrario la válvula se cierra. Este proceso se repite en los vasos siguientes. El vapor final se conduce hacia un condensador en el conducto de salida del ultimo vaso. En este se encuentra un sensor de concentración que mide la concentración del jugo. (Pérez, 2016).

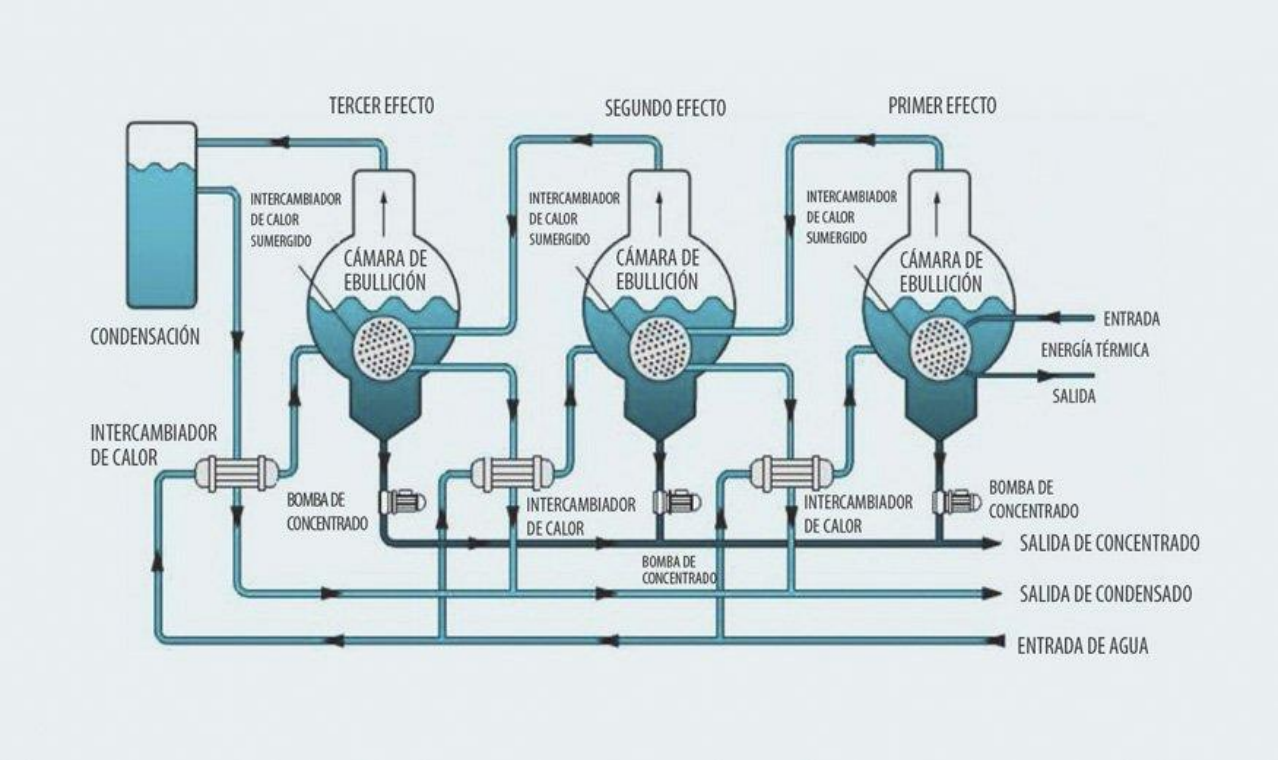


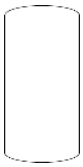
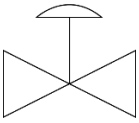
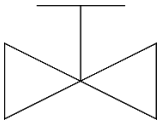
Figura 2. Diagrama de estación de vaporización (Evitech, 2018)

Con el estudio del funcionamiento de las estaciones de vaporización se pudieron identificar las variables y componentes que intervienen en el proceso de concentración del jugo de caña. Esto resulta de vital

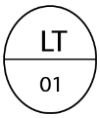
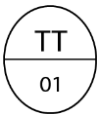
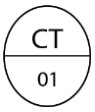
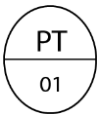
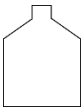
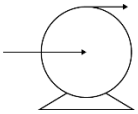
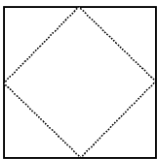
importancia para identificar posibles requerimientos o funcionalidades que brindará la paleta de componentes en aras de desarrollar la propuesta de solución.

La siguiente tabla muestra los componentes que fueron identificados, su representación en norma ISA[2] y su descripción técnica (Oscar Páez Rivera, 2015).

Tabla 1: Descripción de componentes

Nombre	Figura en norma ISA	Descripción
Vaso		Sistema complejo multi-variable diseñado para aumentar la concentración del jugo de caña.
Condensador		Medio técnico de automatización que convierte el vapor excedente en agua para ser utilizada posteriormente.
Electroválvula		Actuador eléctrico utilizado para manipular el paso de flujo.
Válvula manual		Actuador manual utilizado para manipular el paso de flujo.

[2] ISA: Norma internacional para la representación de instrumentación industrial.

Sensor de nivel		Sensor utilizado para medir el nivel del vaso.
Sensor de temperatura		Sensor utilizado para medir la temperatura de la calandria localizada en el vaso evaporador.
Sensor de concentración		Sensor utilizado para medir la concentración del jugo de caña.
Sensor de presión		Sensor utilizado para medir la presión dentro de los vasos evaporadores.
Tanque		Recipiente utilizado para almacenar el jugo de caña.
Bomba centrífuga		Dispositivo utilizado para impulsar el jugo de caña hacia los evaporadores.
PLC		Instrumento utilizado para controlar las variables que intervienen en el proceso.

1.4 Estudio de sistemas similares

Los sistemas de supervisión, control y adquisición de datos entran en funcionamiento a raíz de la revolución tecnológica, por lo que han surgido multitud de productos en el mercado, que supervisan la producción de azúcar. Los sistemas SCADA se pueden dividir en dos grupos específicos: de cada fabricante y genéricos. Los primeros solo funcionan con su equipamiento y los sistemas genéricos son válidos para productos de varios fabricantes, estos sistemas necesitan de *software* adicional para establecer las comunicaciones (Penin, 2017).

WinCC

SIMATIC WinCC se encuentra en el *Totally Integrated Automation Portal* (*TIA Portal*) y forma parte de un nuevo concepto de ingeniería integrada que ofrece el fabricante SIEMENS en un entorno conjunto de usuarios para configurar las soluciones de control, visualización y accionamiento. Este marco de ingeniería constituye un hito dentro del desarrollo de *software* y supone un perfeccionamiento consecuente de la filosofía de TIA. El SCADA *WinCC* es un sistemas multiusuario basados en PC (SIEMENS, 2018).

A continuación, se definen algunas ventajas:

- Interfaz de configuración innovadora y basada en los últimos avances en tecnología de *software*.
- Completas librerías para objetos definibles y *facePlates*.
- Herramientas inteligentes para la configuración gráfica y de datos masivos (Technology, 2017).
- Los componentes que utiliza son imágenes en Vector Gráfico Escalable (SVG[3], por sus siglas en inglés) lo que permite que la imagen no pierda calidad al aumentar su tamaño debido a que es dibujada dinámicamente por el CPU/GPU mediante fórmulas matemáticas que representan objetos geométricos y admite la modificación de elementos como color desde su codificación (Almutairi, 2018).

Los editores de *software* del *TIA Portal* tienen un diseño unificado y un tipo de navegación común. La configuración de un *hardware*, la programación de la lógica, la parametrización de un convertidor de

[3] SVG: *Scalable Vector Graphic* (SVG, por sus siglas en inglés), los archivos SVG contienen imágenes vectoriales de dos dimensiones que responden a un XML, constituyen un estándar de código abierto desarrollado por el *Consorcio World Wide Web* (W3C). Los archivos SVG permiten una compresión de datos sin pérdida y admiten mapas de bits, gráficos vectoriales y texto.

frecuencia o el diseño de una pantalla HMI dispone de editores con un mismo diseño orientado a un uso intuitivo, con el consiguiente ahorro de tiempo y costo. Las funciones, propiedades y librerías se muestran automáticamente de forma más intuitiva según la actividad que se desee realizar (SIEMENS, 2018).

Ecostruxure

El portafolio de *software* de *Schneider Electric* proporciona soluciones que le permiten concentrarse en aumentar la eficiencia del capital y la efectividad del operador, maximizar el rendimiento y la confiabilidad de los activos, administrar la producción y la seguridad. En particular *Ecostruxure* asegura que los sistemas eléctricos estén actualizados, lo cual es de vital importancia para mantener el hogar seguro. Para disfrutar de un entorno más seguro y una fuente de alimentación sin interrupciones, es esencial realizar actualizaciones de instalación periódicas. Con la tecnología líder en la industria de *Schneider Electric*, este sistema solo funciona con tecnología de *Schneider* (Electric, 2018).

EROS

EROS es un Sistema de Supervisión y Control de Procesos Industriales, facilita a los operadores, ingenieros, supervisores y directivos dirigir cualquier proceso con más eficiencia y productividad. EROS puede trabajar acoplado con diversos sistemas de colección de datos y control, como elemento único o formando parte de una red industrial. Tiene en cuenta todas las características de las variables medidas y realiza un potente tratamiento estadístico y determinístico de las mismas con sólo configurarlo. Está soportado en ambiente Windows NT/2000/XP/Vista/W7/W8 lo cual permite utilizar todas las posibilidades de esta plataforma. Facilita el mando a distancia y el control desde la aplicación, que son herramientas que potencian el automatismo del proceso tecnológico. EROS es un sistema distribuido en el cual sus diferentes componentes se interconectan a través de la Intranet Empresarial. Los componentes que cooperan entre sí son: estaciones de medición, estaciones de visualización, servidores de reportes, páginas web y otras aplicaciones que se desarrollen usando *ErosNet* y *ActiveX*. Estos componentes pueden estar en ordenadores separados vinculados a través de una red *Ethernet* o en un mismo ordenador (Grupo Eros, 2014).

A partir del estudio de las soluciones similares se pudo constatar que los sistemas *WinCC* y *Ecostruxure* no pueden ser aplicados a la solución debido a que son privativos, y por tal incumplen con las políticas de

independencia tecnológica que sigue el país. EROS como fue explicado anteriormente, aunque es un *software* desarrollado en Cuba no es multiplataforma y presenta deficiencia que impiden el correcto funcionamiento de las tareas, por lo que es necesario desarrollar un sistema capaz de satisfacer todas las necesidades del sector azucarero.

Se identificó que la utilización de imágenes en formato SVG garantizan la escalabilidad de los componentes manteniendo su calidad sin importar las dimensiones, también permiten modificar valores como color, texto entre otros, en el propio componente con facilidad.

1.5 Descripción de las tecnologías, herramientas y metodologías a utilizar

El constante desarrollo de las tecnologías ha permitido que las industrias sustituyan la mano de obra humana por maquinarias capaces de supervisar y controlar la producción las 24 horas (Machado, 2009). Esto se logra con la unión de *hardware* y *software* diseñados para la automatización de procesos industriales. El *software* utilizado en estos casos son los sistemas SCADA estos sistema deben cumplir con características específicas como monitoreo, control, supervisión y diagnóstico (Lozano & Romero, 2013). El sistema AREX en su estado actual cumple con la característica de medición y adquisición de datos (Duverger et al., 2017), por lo tanto no debe considerarse aún un sistema SCADA.

La propuesta de solución que se persigue es una paleta de componentes gráficos para la representación de estaciones de vaporización desde del sistema AREX. Por lo que, en aras de facilitar futuros mantenimientos perfectivos al sistema por parte de otros desarrolladores, se escoge la misma metodología de desarrollo, herramientas y arquitectura definidas ya en el sistema.

1.5.1 Metodología de desarrollo

Las metodologías de desarrollo definen un conjunto de procedimientos, técnicas, herramientas, etapas y un soporte documental que ayuda a los desarrolladores durante el proceso de desarrollo de *software* (Pressman, 2012).

Existen dos enfoques: el prescriptivo o tradicional y el ágil. En la actualidad, existe un ambiente de negocio muy variable, donde las peticiones del cliente varían con facilidad y se hace necesario minimizar el tiempo en las entregas de los productos, por lo que el enfoque ágil prevalece sobre el tradicional, al ser este más

dinámico (Britto Montoya & Jaime Andrés, 2016). Existen varias metodologías con enfoque ágil, se han analizado las más representativas y utilizadas en el mundo para el proceso de desarrollo de *software* (Mirta et al., 2016), a continuación, se exponen alguna de estas.

Programación extrema (XP): Una de las características distintivas de XP que comparte con otras metodologías ágiles es que de alguna manera representa la antítesis de lo que es el tradicional proceso de desarrollar *software*. XP es deliberadamente una metodología "liviana" que pasa por alto la utilización de elaborados casos de uso, la exhaustiva definición de requerimientos y la producción de una extensa documentación. La tendencia de entregar *software* en lapsos cada vez menores de tiempo y con exigencias de costos reducidos y altos estándares de calidad, hace que XP sea una opción a considerar. XP es una metodología adecuada para proyectos medianos y pequeños, donde los equipos de desarrollo no pasan de 10 programadores y donde la constante es que los requerimientos cambien, a veces radicalmente durante la etapa de desarrollo. Otra característica que distingue a XP en la práctica es que típicamente, los que se encargan de introducirla en los ambientes de trabajo son los propios encargados del desarrollo y sus equipos de programación. Con esto difiere a lo que sucede con otras metodologías, las cuales normalmente se introducen a un nivel corporativo y gradualmente se van bajando hasta alcanzar a los equipos de desarrollo (Britto Montoya & Jaime Andrés, 2016).

SCRUM: Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Sus principales características se pueden resumir en dos. El desarrollo de *software* se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. Otra de las características importantes son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Además, permite la autonomía de los equipos de trabajo. Utiliza reglas para crear un entorno ágil de administración de proyectos y no prescribe prácticas específicas de ingeniería (Madariaga, Rivero, & Leyva, 2016).

***Crystal Methodologies*:** Se trata de un conjunto de metodologías para el desarrollo de *software* caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de *software* se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se

deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo: *Crystal Clear* (3 a 8 miembros) y *Crystal Orange* (25 a 50 miembros).

El equipo ajusta continuamente sus convenciones de trabajo para adaptarse a:

- Las personalidades particulares del equipo.
- El ambiente local de trabajo actual.
- Las particularidades de las asignaciones específicas (Madariaga et al., 2016).

La cantidad de detalle necesario en la documentación de los requerimientos, diseño y planeamiento varía según las circunstancias del proyecto. Quizás no sea posible eliminar todos los productos del trabajo y las notas promisorias inmediatas tales como los documentos de requerimientos, diseño y los planes para el proyecto, pero pueden reducirse a una comunicación corta, enriquecedora, e informal entre el equipo (Madariaga et al., 2016).

Proceso Unificado Ágil (AUP)

Teniendo en cuenta las características de los proyectos de desarrollo en la UCI se implementó una variante de la metodología AUP (Proceso Unificado Ágil, por sus siglas en inglés), la cual propone aumentar la calidad del *software* que se produce apoyándose en el Modelo CMMI-DEV v1.3. Constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (Rodríguez, 2014).

AUP propone 4 fases (Inicio, Elaboración, Construcción, Transición), pero la versión UCI agrupa las faces Elaboración, Construcción, Transición en solo una, llamada ejecución y agrega la fase de cierre. AUP-UCI propone 7 disciplinas: Modelado de negocio, requisitos, análisis, diseño, pruebas internas, pruebas externas, pruebas de liberación (Rodríguez, 2014).

Después de realizar un estudio de metodologías ágiles que podrían ser utilizadas para el desarrollo de la propuesta de solución se decidió utilizar la metodología AUP en su versión UCI, en su escenario 4: Historia de usuario. Con el objetivo de estar en correspondencia con la metodología empleada en el desarrollo del sistema AREX.

1.5.2 Herramientas de modelado

Visual Paradigm 8.0 es una herramienta que ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de todo tipo de diagramas. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de *software* de forma fiable mediante la utilización de un enfoque Orientado a Objetos. Esta herramienta se caracteriza por: disponibilidad en múltiples plataformas (Windows, Linux), diseño centrado en casos de uso y enfocado al negocio para obtener un *software* de mayor calidad. Usa un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación. Tiene las capacidades de ingeniería directa e inversa. El modelo y el código permanecen sincronizados en todo el ciclo de desarrollo. Permite diseñar diagramas de procesos de negocios, decisión, actor de negocio, documento, diagramas de flujo de datos. Generación de bases de datos mediante la transformación de Diagramas de Entidad-Relación (DER) en tablas de base de datos. Distribución automática de diagramas, reorganización de las figuras y conectores de los diagramas de modelado de lenguaje unificado (Pressman, 2012).

1.5.3 Lenguaje de programación

El sistema de medición y adquisición de datos AREX fue desarrollado en los lenguajes C++, QML y *QtScript* (Basado en el estándar ECMAScript-262), bajo el paradigma de la programación orientada a objetos. Para el desarrollo de la solución se utilizará el lenguaje QML.

QML 2.0: Es un lenguaje que permite a los desarrolladores y diseñadores crear aplicaciones de gran rendimiento, animadas de forma fluida, visualmente atractivas y ofrece una sintaxis muy legible. QML es un lenguaje declarativo diseñado para describir la interfaz de usuario de un programa: cómo se ve y cómo se comporta (Qt, 2018).

1.5.4 Framework de programación

Qt *5.10*: Marco de trabajo multiplataforma que permite al equipo implementar una base de código única, proporcionando API[4] (Celine Wan Shi Ann & Iqbal, 2017) comunes en todas las plataformas compatibles. Contiene elementos de UI listos para usar, bibliotecas de C ++ y un completo entorno de desarrollo (Qt, 2018).

1.5.5 Entorno de desarrollo

***Qt Creator 4.6.1*:** es un entorno de desarrollo integrado y multiplataforma para crear aplicaciones en C ++ y QML para múltiples plataformas de escritorio, integradas y móviles. Cuenta con un editor de código y está integrado con herramientas para diseñar, codificar, probar e implementar. Brinda una interfaz simple e intuitiva (Qt, 2018).

1.5.6 Herramienta para dibujo vectorial

Actualmente, el mayor obstáculo para el aprendizaje y aprovechamiento pleno de las tecnologías de la información y las comunicaciones es el alto costo del *software* especializado. *Inkscape* está realizado bajo licencia *General Public License* (GNU, por sus siglas en inglés), diseñado para que todos los usuarios puedan usarlo de manera gratuita.

Inkscape* 0.92.4*: Inicia en el año 2003, con el objetivo de diseñar una aplicación que sea capaz de aprovechar al máximo las ventajas del estándar SVG (Almutairi, 2018). *Inkscape* es un editor gráfico en formato vectorial, de código abierto y multiplataforma. Las características soportadas incluyen: formas, trazos, marcadores, clones, mezclas de canales alfa, transformaciones, gradientes, patrones, y agrupamientos. También soporta metadatos, edición de nodo, capas, operaciones complejas con trazos, vectorización de archivos gráficos, textos en trazos, alineación de textos, edición de XML, entre otras operaciones de utilidad para el dibujo vectorial (Paul, 2007).

[4] API: (*Application Programming Interface* por sus siglas en inglés), es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

Actualmente, *Inkscape* es un de *software* que proporciona la libertad creativa que permiten materializar idea de forma rápida y eficaz. Es además, una aplicación multiplataforma, esto quiere decir que existen paquetes de instalación para distintos sistemas operativos como Linux, Mac OS X, *Windows* (Zepeda, 2015).

1.5.7 Patrón de arquitectura

Antes de hablar de patrones se debe analizar el concepto de arquitectura de *software*. A partir de este se tendrá una visión más clara, para llegar a una definición de patrón de arquitectura y como emplearla en el desarrollo de la solución. La arquitectura es la estructura de organización de los componentes de un programa, la forma en la que estos interactúan y la estructura de datos que utilizan. Los componentes se generalizan para que representen los elementos de un sistema grande y sus interacciones. Una meta del diseño del *software* es obtener una aproximación arquitectónica de un sistema. Ésta sirve como estructura a partir de la cual se realizan las actividades de diseño más detalladas (Pressman, 2012).

Un conjunto de patrones arquitectónicos permite que el ingeniero de *software* resuelva problemas de diseño comunes. Describe una estructura de diseño que resuelve un problema particular del diseño dentro de un contexto específico y entre fuerzas que afectan la manera en la que se aplica y en la que se utiliza dicho patrón (Pressman, 2012).

1.5.8 Arquitectura modular

La modularidad es la manifestación más común de la división de problemas. El *software* se divide en componentes con nombres distintos y abordables por separado, en ocasiones llamados módulos, que se integran para satisfacer los requerimientos del problema (Pressman, 2012).

El sistema AREX cuenta con una arquitectura modular separada en cinco módulos que serán explicados a continuación para una mejor comprensión de la arquitectura presente.

El HMI Editor es el encargado de la configuración de todos los componentes, generando un archivo global de configuración en formato XML que es accedido únicamente por el Servidor de Seguridad. El HMI Ejecución en comunicación temporal con el Servidor de Seguridad, permite la autenticación de los usuarios y recibir la configuración en formato XML además de las imágenes necesarias. A partir de la configuración recibida y en comunicación permanente con el Servidor de Recolección, permite tener acceso de forma

gráfica a la información del proceso que se monitorea y ejercer control manual sobre él. El Servidor de Recolección en comunicación temporal con el Servidor de Seguridad recibe la configuración en formato XML, realiza la comunicación directa con los dispositivos de campo y brinda dicha información a quien la solicite. Los Servidores de Históricos en comunicación temporal con el Servidor de Seguridad, reciben la configuración en formato XML. Además, en comunicación permanente con el Servidor de Recolección, permiten almacenar puntos de variables y alarmas en sus correspondientes bases de datos, para inmediata o posterior consulta. El HMI Cliente de BDH, en comunicación directa con el Servidor de Históricos con *SQLite*, permite generar gráficas en tiempo real y la consulta de los datos históricos almacenados. EL HMI *Grafana*, accede a la información almacenada en la base de datos *Influxdb* del Servidor de Históricos correspondiente. El Servidor Multimedia en comunicación temporal con el Servidor de Seguridad recibe la configuración en formato XML y accede a las cámaras configuradas para obtener su *streaming* de video o ejecutar instrucciones sobre las mismas. De igual forma, este módulo brinda *streaming* de video procesado y ejecución de comandos a quien lo solicite (Tahiri et al., 2019).

El módulo Editor

El módulo HMI editor presenta el patrón arquitectónico Modelo Vista Controlador (MVC, por sus siglas en inglés). La vista del módulo Editor de AREX se compone de un explorador de proyectos en forma arbórea, una paleta de componentes gráficos organizados por categorías, un inspector de propiedades para visualizar propiedades de un ítem seleccionado de la estructura de un proyecto. Cuenta además con un área central de visualización donde se muestra la vista de un ítem seleccionado, una barra de herramientas donde aparecen algunas acciones generales de la aplicación y otras que se habilitan según el ítem seleccionado. Por último, un menú donde aparecen las acciones generales de la aplicación y otras que se habilitan según el ítem seleccionado. La propuesta de solución queda enmarcada como una nueva categoría en la paleta de componentes gráficos del HMI Editor, siendo necesario implementar los Ítem que representan los componentes en la vista (Tahiri et al., 2019).

1.5.9 Conclusiones parciales

- Con el análisis del funcionamiento del proceso de producción de azúcar fueron identificadas otras áreas en las que para futuras versiones podría ser aplicado el sistema AREX.

- Después de caracterizar el funcionamiento de una estación de vaporización se identificaron 11 componentes a desarrollar como parte de la propuesta de solución.
- A partir del estudio de los sistemas similares se pudo determinar que los sistemas *WinCC* y *Ecostruxure* no se pueden aplicar a la solución del problema debido a que son privativos. En el caso del sistema EROS presenta deficiencias con el tratamiento de los componentes gráficos, además no es un sistema multiplataforma. Por lo tanto, es necesario desarrollar un sistema capaz de satisfacer las necesidades el sector azucarero.
- Después de analizar la metodología de desarrollo, herramientas y arquitectura que utiliza AREX se determinó utilizarlas en el desarrollo de la propuesta de solución, en aras de facilitar futuros mantenimientos perfectivos al sistema por parte de otros desarrolladores.

Capítulo 2. Propuesta de solución

En este capítulo se modela la propuesta de solución, se exponen los conceptos del dominio, la especificación de los requisitos funcionales y no funcionales de la paleta de componentes gráficos para el sector azucarero, se presentan las historias de usuario y la interacción de los actores con el sistema, así como una descripción detallada de los mismos.

2.1 Modelo de Dominio

Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos *software*. El modelo del dominio muestra clases conceptuales significativas en un dominio de problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes *software* (Larman, 2003).

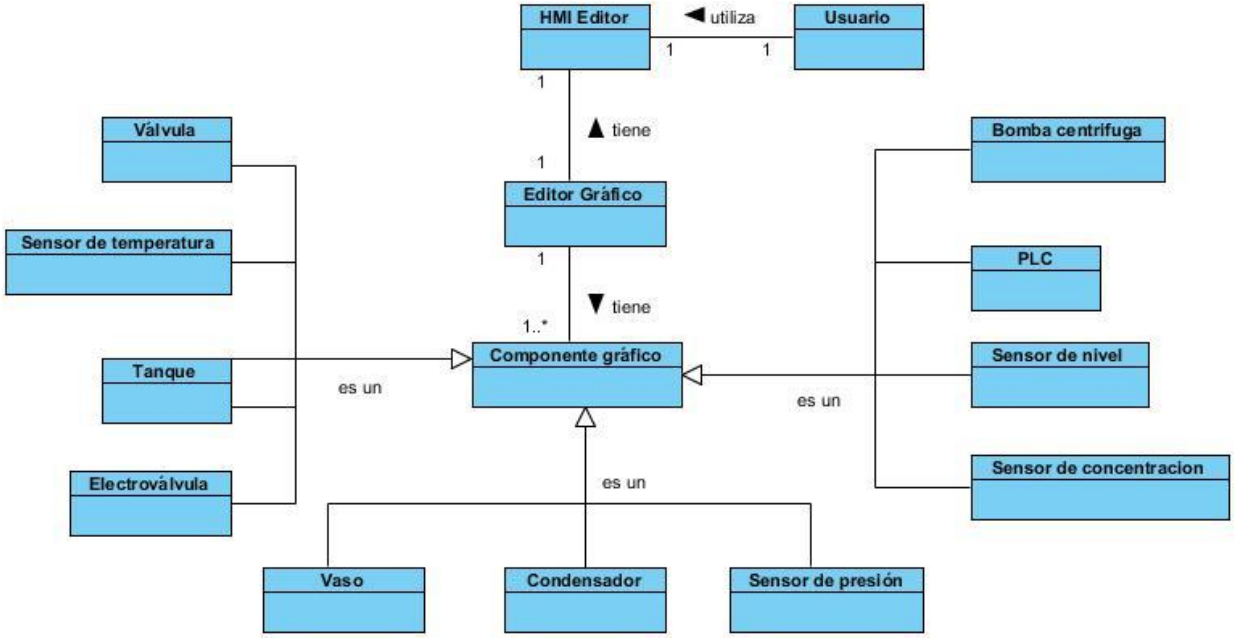


Figura 3. Modelo de dominio

Descripción del modelo de dominio:

HMI Editor: Interfaz gráfica que se encarga de mostrar los procesos que ocurren en el campo.

Componente gráfico: Representa de manera visual un conjunto de datos a través de una imagen.

Editor gráfico: Interfaz donde se muestra la paleta de componentes gráficos disponibles.

Bomba centrífuga: Ítem de tipo bomba centrífuga.

PLC: Ítem de tipo PLC.

Sensor de nivel: Ítem de tipo sensor de nivel.

Sensor de concentración: Ítem de tipo sensor de concentración.

Sensor de temperatura: Ítem de tipo sensor de temperatura.

Sensor de presión: Ítem de tipo sensor de presión.

Tanque: Ítem de tipo tanque.

Electroválvula: Ítem de tipo electroválvula.

Válvula: Ítem de tipo válvula.

Vaso: Ítem de tipo vaso.

Condensador: Ítem de tipo condensador.

2.2 Captura de requisitos

Según *Sommerville* (2011): "*Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información*" (Sommerville, 2011). Los requisitos se pueden clasificar en funcionales y no funcionales. Existen varios métodos para la descripción de requisitos, se utilizará la historia de usuario el cual pertenece al escenario cuatro de la metodología AUP-UCI.

2.2.1 Requisitos funcionales

Según *Sommerville* (2011) se define como requisito funcional (RF): *"(…) las declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a las entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer"* (Sommerville, 2011).

A continuación, se enumeran los requisitos funcionales que fueron definidos para dar solución al problema.

RF1: Representar los componentes en la paleta de componentes del HMI Editor del sistema AREX con su correspondiente ícono y nombre.

RF2: Crear una o más instancias de los componentes en un despliegue del HMI Editor del sistema AREX.

RF3: Expandir o contraer las dimensiones del componente en caso de poderse redimensionar en el HMI Editor del Sistema AREX.

RF4: Asignar una variable a una instancia de componente en el HMI Editor del sistema AREX.

RF5: Modificar el valor de la variable asignada al ejecutar eventos del mouse sobre la instancia del componente en el HMI Editor y HMI Ejecución del sistema AREX.

RF6: Visualizar el componente en el HMI Ejecución del sistema AREX conforme a la configuración de propiedades realizadas en el editor.

RF7: Modificar el estado de la instancia del componente (valor mostrado) al modificar el valor de la variable asignada en el HMI Editor y HMI Ejecución del sistema AREX.

RF8: Editar las instancias de componentes en el inspector de propiedades del HMI Editor del sistema AREX.

RF9: Definir estado en el inspector de propiedades para el componente Válvula.

RF10: Definir porciento de apertura en el inspector de propiedades para el componente Electroválvula

RF11: Definir la velocidad angular en el inspector de propiedades para el componente Bomba centrífuga.

RF12: Definir el nivel máximo en el inspector de propiedades para el componente Tanque.

RF13: Definir el nivel mínimo en el inspector de propiedades para el componente Tanque.

RF14: Visualizar aumento de nivel en el componente Tanque.

RF15: Visualizar en los componentes Sensor de presión, Sensor de temperatura, Sensor de nivel y Sensor de concentración, del valor de la variable asignada en el inspector.

RF16: Visualizar movimiento de la aguja en componente Sensor de Presión.

2.2.2 Requisitos no funcionales

Según *Sommerville* (2011) se define como requisito no funcional: *"(…) restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema"* (Sommerville, 2011).

***Software*:**

- El sistema debe mostrarse en forma de aplicación de escritorio.
- Los íconos de los componentes deben estar en norma ISA.
- Se deben utilizar imágenes vectoriales (SVG) en la implementación de los componentes.
- El sistema debe funcionar sobre los sistemas operativos Windows y sistema Linux.

Usabilidad:

- El sistema debe contar con una paleta de componentes gráficos estructurados por categoría en la cual queda enmarcada la categoría de estación de vaporización y ésta a su vez debe desplegar los 10 componentes gráficos que la componen.
- Se debe visualizar el nombre y el ícono de cada componente para una fácil identificación.

Apariencia o interfaz:

- Diseño de los componentes debe ser representativo, permitiendo al usuario identificarlos rápidamente.
- El producto final debe tener una interfaz usable.
- El ícono de cada componente debe corresponder con la norma ISA (Oscar Páez Rivera, 2015).

***Hardware*:**

- La PC de trabajo debe tener como mínimo 2 GB de RAM, 3 GB de espacio libre en el medio de almacenamiento y 2.16 GHz de velocidad del procesador.

2.3 Historia de usuario

Las historias de usuario pertenecen al escenario número 4 de la metodología AUP-UCI. Este escenario se aplica a los proyectos que hayan evaluado el negocio y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos (Rodríguez, 2014).

A continuación, se muestran las historias de usuario de los RF1 y RF2, el resto de las historias de usuario se encuentran disponibles en el Anexo 1:

Tabla 2. Historia de usuario del RF1

Historia de usuario	
Número: 1	Nombre del requisito: Representar los componentes en la paleta de componentes del Editor del sistema AREX con su correspondiente ícono y nombre.
Programador: Karla Martínez Carrera	Iteración asignada: 2
Prioridad: Alta	Tiempo estimado: 1 semana

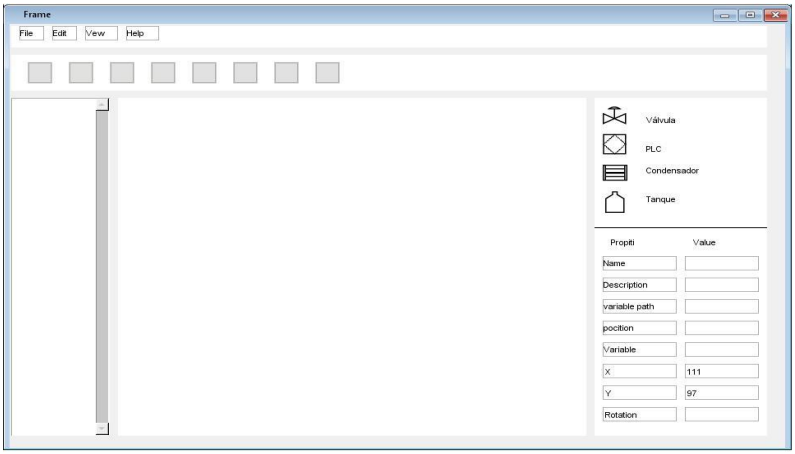
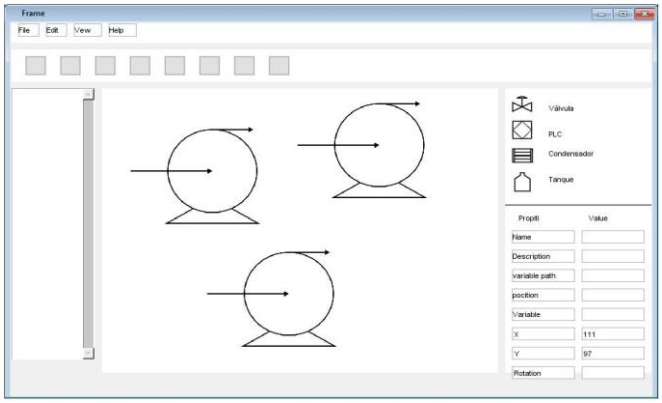
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá visualizar los componentes de una estación de vaporización en la paleta de componentes del HMI editor.	
Prototipo de interfaz: 	

Figura 4. Prototipo de interfaz para el requisito 1

Tabla 3. Historia de usuario del RF2

Historia de usuario	
Número: 2	Nombre del requisito: Crear una o más instancias de los componentes en un despliegue del Editor del sistema AREX.
Programador: Karla Martínez Carrera	Iteración asignada: 1

Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá crear una o más réplicas de un mismo componente.	
Prototipo de interfaz:  **Figura 5. Prototipo de interfaz para el requisito 2**	

Con la descripción de cada requisito en historia de usuario, permite comprender mejor la respuesta que debe dar el sistema. Con la realización de los prototipos de interfaz queda diseñado la forma en que será visualizado cada requisito.

2.4 Diagrama de paquetes

El diagrama de paquetes es el encargado de representar las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre ellas (Cardozzo, 2014).

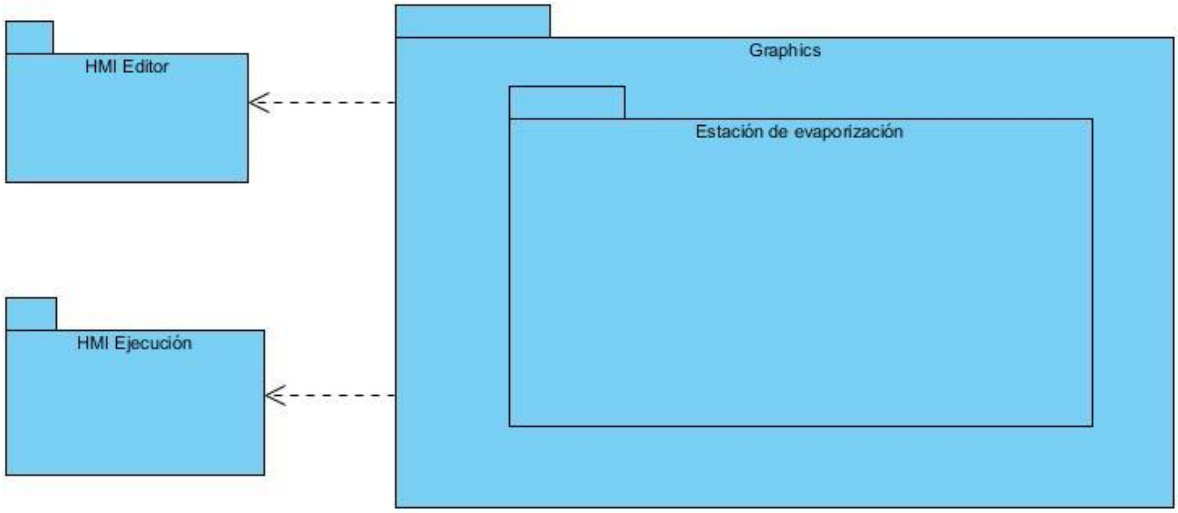


Figura 6. Diagrama de paquetes

El diagrama representa los paquetes HMI Editor y HMI Ejecución que son los módulos que consumen del paquete *Graphics* que incluye los componentes gráficos entre los que se encuentra el paquete Estación de vaporización el cual contiene los componentes que conforman esta categoría como: Válvula, Bomba, Vaso entre otros.

2.1 Patrones de diseño

Cada patrón describe un problema que ocurre una y otra vez en el entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de una vez. Los patrones de diseño además describen soluciones simples y elegantes a problemas específicos en el diseño de *software* orientado a objetos (Guerrero, Suárez, & Gutiérrez, 2018).

2.1.1 Patrones GRASP

Los patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades, expresados en forma de patrones, define un grupo de principios fundamentales para diseñar eficazmente un software (Larman, 2003).

En la propuesta de solución se aplican los siguientes:

Patrón creador: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase? El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento (Astudillo & Visconti, 2019). Este patrón está evidenciado en la clase ***GraphicPack***, la cual se encarga de crear todos los componentes definidos como ***SvgItem*** dentro de la paleta.

Experto: Consiste en la asignación de responsabilidades a la clase que contiene la información necesaria para el cumplimiento de las mismas. Se puede aprovechar la oportunidad de reutilizar componentes en futuras aplicaciones. Este patrón esta evidenciado en la clase ***SvgItem***.

Bajo acoplamiento: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización? El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa, que una clase no depende de muchas clases (Astudillo & Visconti, 2019). Este patrón se evidencia en las clases que modelan los componentes (***Svgitem***), los cuales carecen de total relación.

Alta cohesión: Mantiene la asignación de responsabilidades de manera que la información almacenada en un **Ítem** sea coherente. El uso de este patrón se ve reflejado en cada componente, ya que este almacena la información del mismo de manera coherente.

2.6 Conclusiones parciales

- La realización del modelo de dominio permitió obtener una representación de los conceptos utilizados en el desarrollo de la propuesta de solución, evidenciando las relaciones entre ellos.
- El levantamiento de requisitos permitió identificar las funcionalidades que se deben implementar para dar solución al problema planteado.
- El levantamiento de requisitos no funcionales permitió definir los requerimientos de hardware, software, apariencia y usabilidad que debe cumplir la paleta de componentes.
- Las descripciones de las historias de usuario permitieron una mejor comprensión de los requisitos funcionales.

Capítulo 3. Implementación y prueba

Este capítulo describe la implementación de la propuesta de solución y las pruebas realizadas a la aplicación. Para ello se realiza un modelado de las funcionalidades y la implementación de la propuesta de solución. Se muestra el estándar de codificación empleado para el desarrollo de la propuesta de solución, así como el diagrama de componente y despliegue. Con el objetivo de comprobar la calidad del *software* se realizan pruebas funcionales a la paleta de componentes. Finalmente, se obtiene una paleta de componentes que permite la representación gráfica de una estación de vaporización para la industria azucarera desde el sistema AREX.

3.1 Modelo de implementación

Un diagrama de componentes representa la separación de un sistema de *software* en componentes físicos, por ejemplo: archivos, módulos, paquetes, código fuente, ejecutable entre otros. Muestra las dependencias y organización existente entre estos componentes. Los diagramas de componentes prevalecen en el campo de la arquitectura de *software*, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema (Sommerville, 2011).

3.1.1 Diagrama de componente

El diagrama de componente tiene en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión de *software*, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo (Sparks, 2015) (Giraldo, Leyva, & Moreno, 2011).

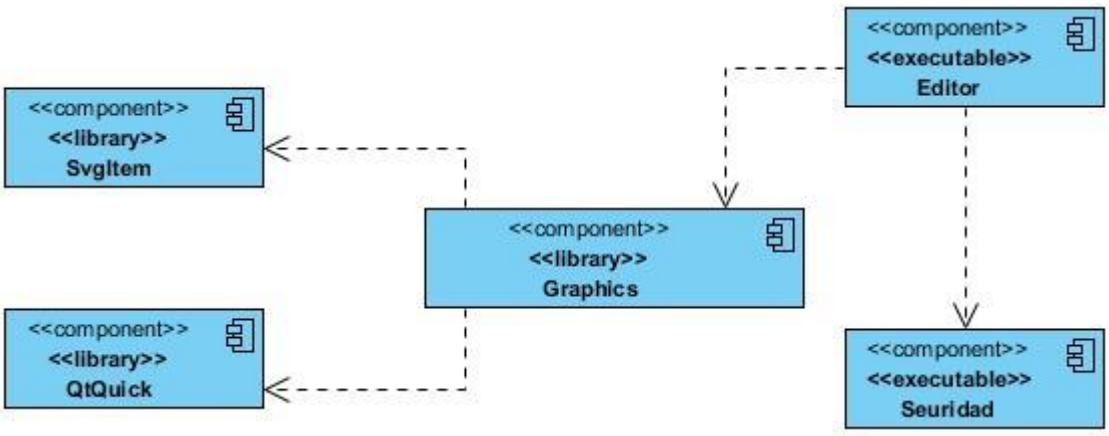


Figura 7. Diagrama de componentes

Editor: Módulo de editor es la aplicación donde se trabaja el entorno de configuración para representar los procesos.

Seguridad: Es el módulo encargado de consumir la configuración del editor.

Graphics: Librería que contiene los componentes gráficos que son cargados por el editor.

SvgItem: Librería que permite crear los componentes.

QtQuick: Librería que permite construir aplicaciones QML

3.1.1 Diagrama de despliegue

Con el objetivo de proveer una descripción de la distribución del sistema se realiza el modelo de despliegue, mediante el cual se muestran las relaciones físicas entre los componentes *hardware* y *software* en el sistema, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes *software*. Los diagramas de despliegue son capaces de describir la arquitectura física del sistema durante la ejecución en términos de procesadores, dispositivos y componentes de *software*. Permiten una mejor comprensión entre la correspondencia de la arquitectura de *software* y la arquitectura de *hardware* (Pressman, 2012) (Huallpara, Michael, Limachi, & Susana, 2016).

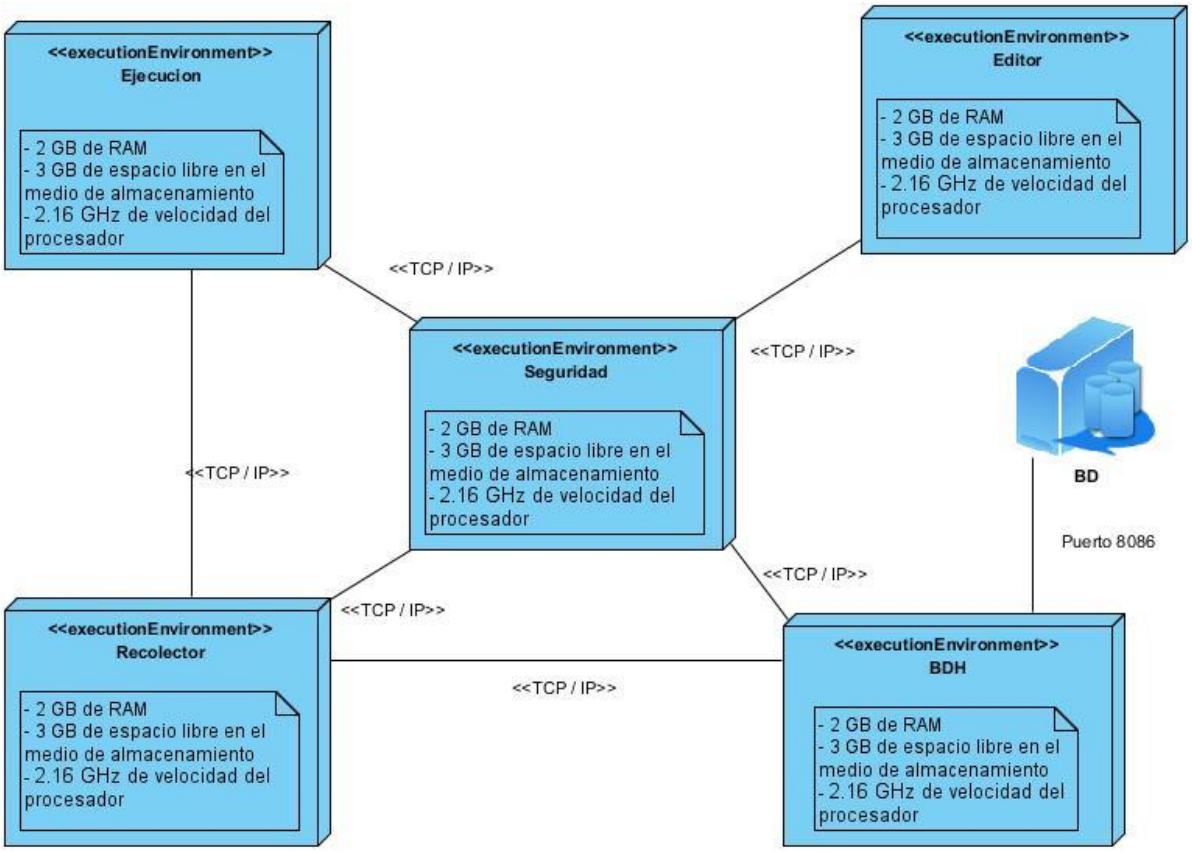


Figura 8. Diagrama de despliegue

3.3 Estándar de codificación

Los estándares de codificación, también llamados estilos de programación, son convenios para escribir código fuente en ciertos lenguajes de programación. Estos estándares facilitan el mantenimiento del código, sirven como punto de referencia para los programadores, mantienen un estilo de programación y ayudan a mejorar el proceso de codificación, haciéndolo más eficiente y en muchos casos reutilizables (Lorenzo, 2015). Los estándares de codificación permiten entender de manera rápida, el código empleado en el desarrollo de una aplicación. El uso de las técnicas de codificación es de gran importancia para realizar buenas prácticas de programación con vista a generar un código de alta calidad (Pressman, 2012).

La solución propuesta en este trabajo forma parte del sistema AREX, por lo tanto, el estándar de codificación utilizado fue definido por el proyecto. Algunas de estas pautas son:

- El código será escrito en inglés y la documentación en español.
- Las variables y funciones comienzan con letra minúscula. Cada palabra consecutiva en el nombre comienza con letra mayúscula.
- Los atributos que aparecen en el inspector de propiedades deben empezar con m_ seguido del nombre del atributo.
  Ejemplo: *m_level_max*.
- Las funciones utilizan la nomenclatura camello.
- Los valores de los numerativos deben ser con letras mayúsculas.
- Las secciones *public*, *protected* y *private* son declaradas en el orden expuesto.

3.3 Solución del problema

En este epígrafe se muestra la interfaz del sistema AREX donde se encuentra la clasificación de estación de vaporización dentro de la paleta de componentes en el entorno de configuración (Editor) y entorno de visualización (Ejecución).

En la siguiente imagen se muestra los componentes para la representación gráfica de una estación de vaporización al añadir un total de 11 componentes con funcionalidades individuales. De esta manera se da cumplimiento al problema planteado sobre la representación de una estación de vaporización desde el sistema AREX.

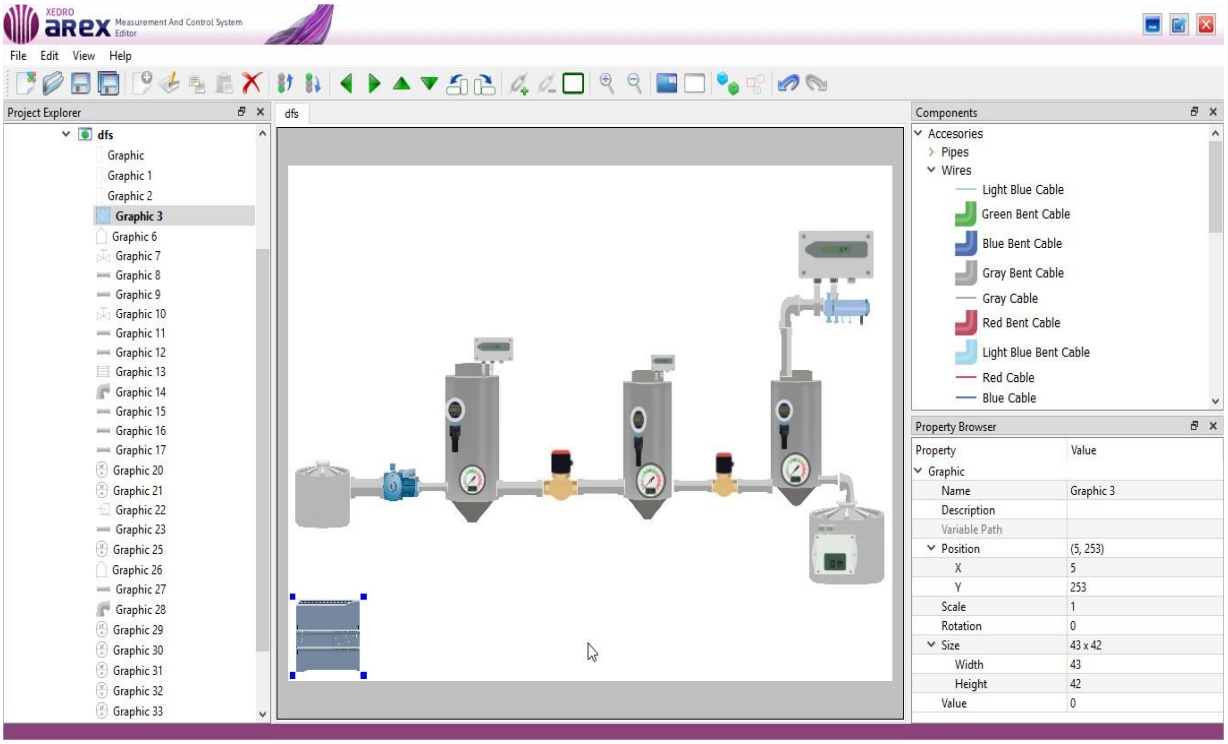


Figura 9. Interfaz del Editor del sistema AREX

3.4 Pruebas

Las pruebas de *software* son muy importantes en la obtención de un producto de alta calidad y buen funcionamiento, por lo que su objetivo fundamental es verificar y validar que realmente el *software* realice lo que está planificado que haga. Forman un punto indispensable para cerciorarse de que un sistema es realizado con calidad y para reducir el número de errores que no fueron detectados en la etapa de implementación. Cuando se aplican pruebas a un *software* es necesario tener en cuenta el objetivo que se persigue, debido a que las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo. Dentro de las estrategias de pruebas que existen pueden ser mencionadas las pruebas de unidades, las pruebas de integración, las pruebas de sistema, las pruebas de aceptación y pruebas de regresión (Pressman, 2012).

Pruebas unidad: Pretenden probar que los fragmentos individuales (unidades) que forman el sistema cumplen las especificaciones y tienen el comportamiento esperado. Cumplimiento del objetivo de un

componente de *software* (Método-Clase) (GAVIRIA, 2017). El primer nivel de las pruebas es el de las pruebas unitarias o de componente, consiste en la verificación de unidades de *software* de forma aislada, es decir, probar el correcto funcionamiento de una unidad de código. Este tipo de prueba suelen ser realizadas por los desarrolladores, ya que es muy recomendable conocer el código fuente del programa y generalmente se realizan pruebas de caja blanca o se analiza el código para comprobar que cumple con las especificaciones del componente (Peño, 2015).

Pruebas integración: Los módulos o componentes funcionan integrados. Los datos enviados de un módulo o componente se reciben de manera consistente en el otro (GAVIRIA, 2017). Tienen el objetivo de verificar que las interfaces entre las entidades externas (usuarios) y las aplicaciones funcionan correctamente. Son utilizadas para determinar cómo la base de datos de prueba será cargada. Verifican que las especificaciones de diseño sean alcanzadas.

- Describe cómo verificar que las interfaces entre las componentes de *software* funcionan correctamente.
- Determina cómo la base de datos de prueba será cargada.
- Determina el enfoque para avanzar desde un nivel de integración de las componentes al siguiente.
- Decide qué acciones tomar cuando se descubren problemas (Mera, 2016).

Pruebas de sistemas: Los procesos soportados por la aplicación se cumplen completamente, es decir, los procesos fluyen desde su inicio hasta el final (GAVIRIA, 2017). Se utilizan para validar que todas las funciones y componentes del sistema trabajen correctamente. Estas pruebas se ejecutan una vez concluidas las pruebas de componentes y las de integración. A diferencia de estas 2 pruebas, las pruebas de sistema se realizan desde el punto de vista del usuario y validan en su totalidad los requerimientos del usuario especificados e incluso aquellos no especificados u olvidados que afectan el funcionamiento del sistema (Mera, 2016).

Método de caja negra: La prueba de caja negra se refiere a las pruebas que se llevan a cabo en la interfaz del *software*. Una prueba de caja negra examina algunos aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del *software* (Pressman, 2012). La técnica de parición equivalente pertenece a las técnicas de caja negra. Se basa en dividir en subconjuntos equivalentes respecto a una relación específica el dominio de las entradas. La prueba de un valor representativo de una

clase permite suponer que el resultado obtenido será el mismo que para otro valor de la clase. Se realiza un conjunto representativo de casos de prueba para cada clase de equivalencia (Blanco, 2015).

Pruebas de Aceptación: Realizadas con los clientes y define su aceptación del *software* (GAVIRIA, 2017).

3.4.1 Aplicación de las pruebas de caja negra

Para comprobar el correcto funcionamiento de la paleta de componentes se aplicó el método de caja negra empleando la técnica de partición equivalente el cual permitió evaluar los valores de entrada al sistema. Los casos de prueba diseñados pretenden demostrar que el sistema funciona de manera correcta, las entradas válidas y no válidas se aceptan de forma adecuada y los resultados obtenidos son correctos.

Tabla 4 Casos de prueba

Descripción general			
Mostrar e interactuar con los componentes de la categoría estación de vaporización.			
SC1 Mostrar los componentes en la paleta de componentes del HMI Editor del sistema AREX con su correspondiente ícono y nombre.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Mostrar los componentes en la paleta de componentes	En la derecha del HMI Editor se encuentra la paleta de componentes con la categoría	El sistema muestra los componentes con su ícono en la paleta de componentes.	Siguiendo la siguiente ruta: En la paleta de componentes situada en la derecha del editor se muestra la clasificación de estación de estación de vaporización.

	estación de vaporización.		
SC2 Crear una o más instancias de los componentes en un despliegue del HMI Editor del sistema AREX.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 2.1 Crear una o más instancias de los componentes	1 Siguiendo la ruta del flujo central se selecciona el componente con un clic. 2 Se da clic en el despliegue. 3 se repite le proceso.	1 Permite dar clic en el componente. 2 Permite dar clic en el despliegue. 3 Aparece el componente en el despliegue.	Siguiendo la siguiente ruta: Se selecciona el componente de la paleta con un clic. Se da clic en el despliegue. Se repite este proceso cuantas instancias se desee crear.

Con las pruebas realizadas se detectaron 12 no conformidades (NC) para un total de 16 historias de usuario. Las cuales se recogen en la siguiente tabla.

Tabla 5 Relación de no conformidades detectadas por requisito e iteración

No. NC	Requisito Funcional	Tipo de NC	Descripción	Iteración	Estado

1	RF15	Error funcional	En el componente Sensor de nivel no se muestra el valor asignado en el inspector de propiedades.	1	Resuelta
2	RF15	Error funcional	En el componente Sensor de concentración no se muestra el valor asignado en el inspector de propiedades.	1	Resuelta
3	RF10	Error funcional	El componente Electroválvula no muestra el porciento de apertura.	1	Resuelta
4	RF10	Error funcional	El componente Electroválvula no cambia de color rojo cuando pasa de 0% a 1% en adelante.	1	Resuelta
5	RF10	Error de idioma	El parámetro porciento de apertura que se encuentra en el inspector de propiedades no aparece en letra inicial mayúscula.	1	Resuelta
6	RF11	Error de idioma	El parámetro velocidad angular que se encuentra en el inspector de propiedades no aparece en letra inicial mayúscula.	1	Resuelta
7	RF14	Error de validación	Si se define como nivel máximo 100 y se asigna un valor de 50 el componente Tanque muestra el nivel por en sima del máximo.	2	Resuelta
8	RF14	Error de validación	Si se define como nivel máximo 1 y se asigna un valor de 1 el componente Tanque no muestra correctamente el nivel.	2	Resuelta
9	RF16	Error funcional	La aguja en el componente Sensor de presión no marca el valor asignado en el inspector de propiedades.	2	Resuelta

10	RF16	Error funcional	Si se define como valor máximo 100 y se asigna un valor de 10 la aguja del componente Sensor de presión no marca el valor 10.	2	Resuelta
11	RF5	Error no funcional	El componente Válvula no modifica visualmente su estado al ser modificado en el inspector de propiedades.	3	Resuelta
12	RF5	Error no funcional	El componente Válvula no modifica visualmente su estado al ejecutar eventos de *mouse*.	3	Resuelta

En la primera iteración se encontraron 6, de las cuales 4 fueron de tipo funcional, asociadas a los requisitos: visualizar en los componentes Sensor de presión, Sensor de temperatura, Sensor de nivel y Sensor de concentración el valor de la variable asignada en el inspector y definir porciento de apertura en el inspector de propiedades para el componente Electroválvula y 2 no conformidades de idioma asociadas a los requisitos: definir porciento de apertura en el inspector de propiedades para el componente Electroválvula y definir la velocidad angular en el inspector de propiedades para el componente Bomba centrífuga. En la segunda iteración se encontraron 4 de las cuales 2 de validación asociada al requisito: visualizar aumento de nivel en el componente Tanque y 2 de tipo funcional asociadas al requisito: visualizar movimiento de aguja en componente Sensor de presión. Y en la tercera se encontraron 2 de tipo no funcional asociadas al requisito: modificar el valor de la variable asignada al ejecutar eventos del *mouse* sobre la instancia del componente Válvula. Todas las no conformidades encontradas fueron resueltas después de cada una de las iteraciones, para una cuarta iteración no fueron encontradas no conformidades.

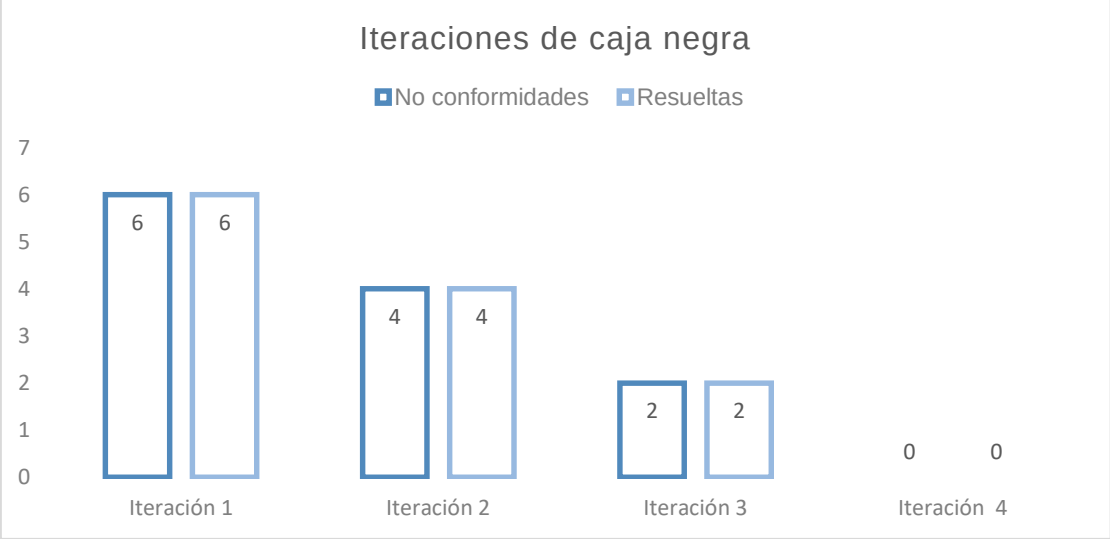


Figura10. Iteraciones de caja negra.

El resultado de las pruebas realizadas al sistema comprobó que funciona satisfactoriamente y cumple con los requisitos funcionales y no funcionales planteados anteriormente.

3.5 Conclusiones parciales

- La realización del diagrama de componentes permitió comprender las restricciones impuestas por el lenguaje de programación.
- El diseño del diagrama de despliegue permitió comprender las relaciones físicas entre los componentes de *hardware* y *software* del sistema.
- Establecer un estándar de codificación permitió obtener una codificación homogénea y clara para futuros mantenimientos al sistema por otros desarrolladores.
- Para la validación del correcto funcionamiento del *software* se realizaron las pruebas al sistema mediante el método de caja negra utilizado la técnica de partición equivalente, la cual permitió detectar a tiempo un total de 12 no conformidades en 4 iteración, las cuales fueron resueltas al final de cada una.

Conclusiones generales

Con la realización de la presente investigación se arriba a las siguientes conclusiones:

- El estudio del sistema AREX evidencia la necesidad de implementar una paleta de componentes que represente una estación de vaporización para poder ser aplicado en la industria azucarera.
- El estudio del funcionamiento de una estación de vaporización en un central azucarero permitió identificar los componentes necesarios para su representación desde el sistema AREX.
- Con la implementación de la paleta de componentes que representa una estación de vaporación se da cumplimiento a los requisitos funcionales y no funcionales establecidos con el objetivo de que el sistema AREX pueda ser aplicado a este tipo de entorno.
- La realización de las pruebas al sistema mediante el método de caja negra, permitieron detectar un total de 12 no conformidades en tres iteraciones, las cuales fueron resueltas al finalizar cada iteración, mejorando así la calidad de la paleta de componentes.

Recomendaciones

Teniendo en cuenta los resultados obtenidos durante la investigación y basados en la experiencia adquirida, se recomienda:

- Adicionar a la solución nuevos componentes, que representen otras etapas del proceso de producción del azúcar.
- Implementar funcionalidades en el módulo HMI Editor que permitan la utilización de conectores lógicos.

Referencias bibliográficas

Almutairi, A. (2018). *A Comparative Study on Steganography Digital Images : A Case Study of Scalable Vector Graphics (SVG) and Portable Network Graphics (PNG) Images Formats*. *9*(1), 170–175.

Astudillo, H., & Visconti, M. (2019). *Fundamentos de Ingeniería de Software*.

Azcuba, G. E. (2018). *Azucareso_es. Habana : AZCUBA*.

Blanco, B. C. (2015). *Ingeniería de Software - Construcción y pruebas*. Retrieved from http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/materiales/tema1 - %0ApruebasSistemasSoftware.pdf

Bolton, W. (2015). *Programmable Logic Controllers*. BOSTON: Elsevier Newnes.

Britto Montoya, & Jaime Andrés. (2016). *Comparación de metodologías ágiles y procesos de desarrollo de software mediante un instrumento basado en CMMI Mapping agile methodologies and software development processes using a CMMI based*. *21*(ISSN 0122-1701), 7.

Cardozzo, D. R. (2014). *Desarrollo de Software. Campus Academy*. IT Campus Academy.

Celine Wan Shi Ann, & Iqbal, N. M. (2017). *Open Application Programming Interface (API): A Financial Revolution*. 57.

Duverger, F. G. V., Vargas, A. G., & Naranjo, L. Z. R. (2017). *Sistema de medición AREX*. Habana.

Electric, S. (2018). Nueva oferta de software industrial y de ingeniería sin igual. Retrieved from https://www.se.com/eg/en/work/solutions/software/.

Enrique Baeyens Lázaro. (2011). *Libro blanco del control automático en la industria de la caña de azúcar*. (December 2011), 396. https://doi.org/10.13140/RG.2.1.4229.2648

Evitech, C. (2018). Evaporadores al vacío de multiple efecto. Retrieved from https://condorchem.com/es/contacto/

Referencias bibliográficas

García, A. S. (2011). Sin azucar no hay país: la industria azucarera y la economía cubana (1919-1939). In *Editorial CSIC - CSIC Press*. Sevilla.

GAVIRIA, B. F. (2017). *Introducción a las pruebas y calidad de software*. *50*.

Giraldo, G., Leyva, A., & Moreno, D. (2011). *Una ontología para la reprecentación de conceptos de diseño de software. 8*.

González, M. M. (2017). La caña es más que azúcar. *Juventud Rebelde*.

Grupo Eros. (2014). *Sistema de supervisión y control de procesos*.

Guerrero, C. A., Suárez, J. M., & Gutiérrez, L. E. (2018). *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web*. https://doi.org/10.4067/S0718-07642013000300012

Guzman, O. (2006). *Implementación de un sistema de control electrónico para mejorar la eficiencia del área de evaporadores.* 134. Retrieved from https://repositoriotec.tec.ac.cr/bitstream/handle/2238/467/Informe Final.pdf?sequence=1&isAllowed=y

Huallpara, M., Michael, H., Limachi, Q., & Susana, N. (2016). Diagrama de Despliegue ANALISIS Y DISEÑO DE SISTEMAS II. Retrieved from virtual.usalesiana.edu.bo/web/practica/archivo/despliegue.doc

Hudedmani, M., Kabberalli, S. K., & Hittalamani, R. (2017). Programmable Logic Controller (PLC) in Automation. *Advanced Journal of Graduate Research*, *2*(1), 37–45. https://doi.org/https://doi.org/10.21467/ajgr.2.1.37-45

Larman, C. (2003). *UML y Patrones* (2nd ed.). Retrieved from www.librosite.net/larman

Lorenzo, A. C. (2015). *Estándares de codificación para C++*.

Lozano, C., & Romero, C. (2013). *Introducción a SCADA*. 30.

Machado, J. (2009). *Automatización de los procesos Productivos en la planta II División Partes y Piezas para la Empresa Indurama S.A. Cuencia*.

Referencias bibliográficas

Madariaga, C., Rivero, Y., & Leyva, A. (2016). Propuesta metodológica para desarrollo de software educativo en la. *Revista Trimestral*, *22*(4), octubre-diciembre. Retrieved from https://www.redalyc.org/pdf/1815/181548029003.pdf

Marino, P. S. A. (2018). Diseño y análisis termodinámico de un sistema de evaporación de triple efecto para el proceso de producción de panela. *Universidad de Piura*.

Mera, J. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería Solidaria*, *12*(20), 163–176. https://doi.org/10.16925/in.v12i20.1482

Mirta, N., Marcelo, M., Juan, A., Lorena, P., José, R., & Cruz, P. J. (2016). *Selección de Metodologías Ágiles e Integración de Arquitecturas de Software en el Desarrollo de Sistemas de Información*. 632–636.

Mónica Mulet-Hing, R. E. F.-S. (2016). *Automatización del tacho cristalizador de azúcar crudo*. Santiago de Cuba.

Oscar Páez Rivera. (2015). *Norma ISA*. 1–14.

Paul, M. (2007). *Importing Vector Graphics : The grImport Package for R*. Retrieved from https://cran.r-project.org/web/packages/grImport/vignettes/import.pdf

Penin, A. R. (2017). *Sistemas SCADA, Guía práctica* (MARCOMBO E).

Peño, S. J. M. (2015). *Pruebas de Software. Fundamentos y Técnicas.*

Pérez-López, E. (2015). SCADA systems in the industrial automation. *Tecnología En Marcha. Tecnología En Marcha*, *28*(4), 3–14.

Pérez, R. R. (2016). *Los evaporadores como objetos de control.* Habana.

Pressman, R. (2012). Ingeniería Del Software I Un Enfoque Práctico. In *Ingeniería Del Software I* (7th ed.). NUEVA YORK.

Qt, C. (2018). Qt APIs & Libraries, Tools and IDE. Qt | Cross-platform software development for embedded & desktop. Retrieved from https://www.qt.io/qt-for-application-development/

Referencias bibliográficas

Rodríguez, T. (2014). *Metodología de desarrollo para la Actividad productiva de la UCI.* 1–16.

Sharan, P., & Bandyopadhyay, S. (2017). Optimal Temperature Selection for Energy Integrated Multiple-Effect Evaporator System. *Process Integration and Optimization for Sustainability*, *1*(3), 189–202. https://doi.org/10.1007/s41660-017-0012-3

SIEMENS. (2018). SIEMENS. SIEMENS. SIMATIC WinCC (TIA Portal) Engineering Software. Retrieved from https://w3.siemens.com/mcms/automation-software/en/tia-portal-software/wincc-tia-portal/wincc-tia-portal-es/Pages/default.aspx

Sommerville, I. (2011). *Software Engineering*. Addison-Wesley.

Sparks, G. (2015). Una Introducción al UML El Modelo de Componentes. *Craftware Consultores Ltda*.

Tahiri, D., Ramirez, G., Alberto, J., & Durán, L. (2019). *Sistema para la adquisición de datos, monitoreo y control en procesos de pequeña y mediana complejidad.*

Technology, P. (2017). *Application of WINCC redundant system in production line Wang Na*. *130*(Fmsmt), 1554–1560.

Zepeda, C. (2015). *Diseño Vectorial Inkscape*.

Zhang, T. Y., Wang, L., Xiang, Y., & Chee-Wooi. (2016). Power System Reliability Evaluation With SCADA Cybersecurity Considerations. *IEEE*, *6*(4), 1707–1721. https://doi.org/10.1109/TSG.2015.2396994

Glosario de términos

API: Interfaz de Programación de Aplicaciones. Representa la capacidad de comunicación entre componentes de software, poseen un conjunto de subrutinas, procedimientos, funciones o métodos ofrecidos por una biblioteca para ser utilizados por otro software.

Azúcar: Nombre común de la sacarosa obtenida de la caña de azúcar o de la remolacha azucarera. Hay distintos tipos dependiendo del grado polarización (contenido en sacarosa). El crudo elaborado por los centrales cubanos contiene 96º, el refinado, refino o azúcar de consumo final, 98, 9º.

Bagazo: Despojo de la caña después de molerla en el central. Se usa como abono y combustible.

Caña de azúcar: Nombre común de la *Saccharum*. Gramínea compuesta de agua, fibras y sacarosa de la que se obtiene el azúcar. Todas las variedades comerciales son del género *Officinarum*.

Central: Fábrica de azúcar. En el siglo XIX designaba a las instalaciones más grandes y modernas frente a las más antiguas, llamadas ingenios. En el XX, una vez se mecanizó toda la industria, ambos términos ingenio y central son sinónimos.

Evaporación: Proceso mediante el cual se depura el guarapo, obteniendo la meladura.

Jugo de caña: Jugo obtenido de la caña tras eliminar el bagazo. Entonces se le añade el calificativo de crudo. Si se ha retirado también la cachaza se habla de guarapo claro.

Meladura: Sustancia obtenida del guarapo claro tras someterlo a un proceso de evaporación.

Tacho: Aparato donde se evapora al vacío la meladura hasta obtener una masa cristalizada.

Anexo 1. Historias de usuario

Tabla 6. Historia de usuario del RF3

Historia de usuario	
Número: 3	Nombre del requisito: Modificar el valor de la variable asignada la ejecutar eventos del mouse sobre la instancia del componente en el HMI Edito y HMI Ejecución del sistema AREX. (si el componente lo admite)
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá modificar el valor de la variable asignada ejecutando eventos de *mouse* sobre la instancia del componente.	
Prototipo de interfaz:	

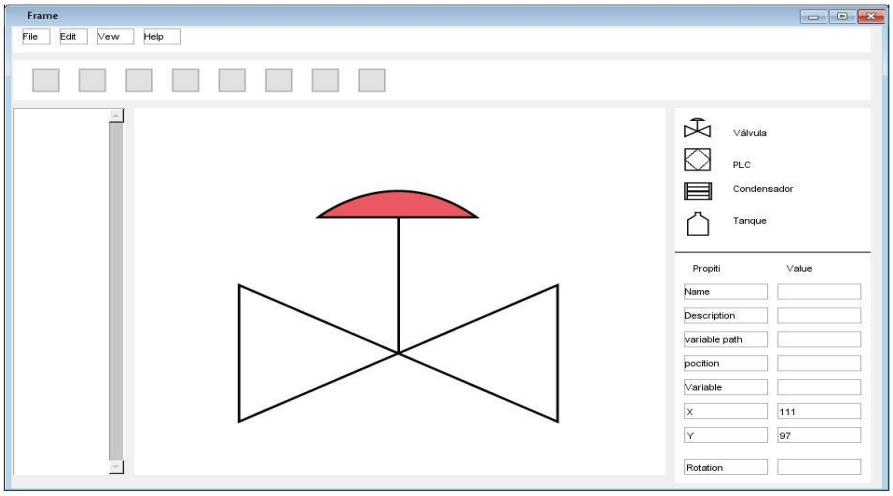


Figura 11 Prototipo de interfaz para el requisito 3

Tabla 7. Historia de usuario del RF4

Historia de usuario	
Número: 4	Nombre del requisito: Expandir o contraer las dimensiones del componente en caso de poderse redimensionar en el Editor del sistema AREX.
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá redimensionar cada componente.	

Prototipo de interfaz:

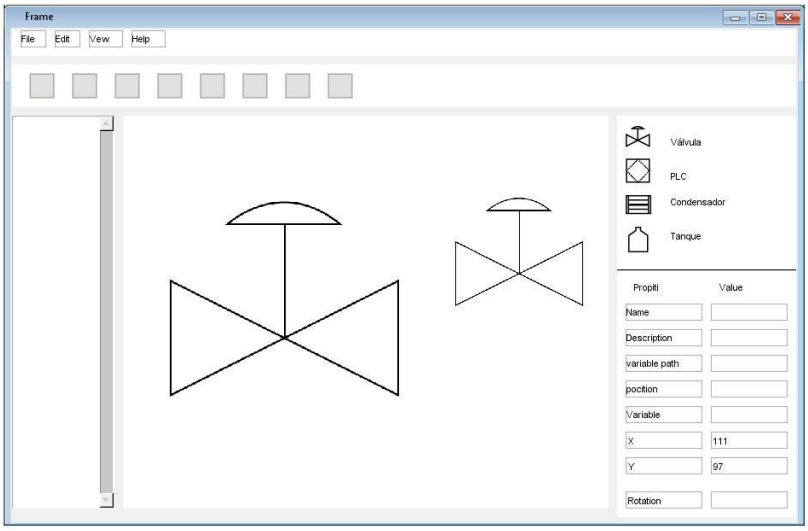


Figura 12. Prototipo de interfaz para el requisito 4

Tabla 8. Historia de usuario del RF5

Historia de usuario	
Número: 5	Nombre del requisito: Asignar una variable a una instancia de componente en el Editor del sistema AREX.
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana

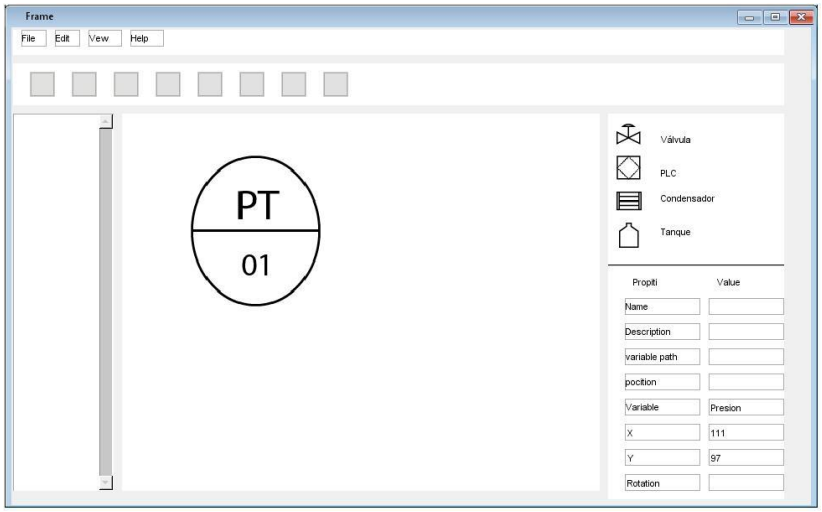
Descripción de la operación: El sistema permitirá asignar una variable a cada componente.
Prototipo de interfaz:  **Figura 13. Prototipo de interfaz para el requisito 5**

Tabla 9. Historia de usuario del RF6

Historia de usuario	
Número: 6	Nombre del requisito: Visualizar el componente en el HMI Ejecución del sistema AREX conforme a la configuración de propiedades realizadas en el Editor.
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana

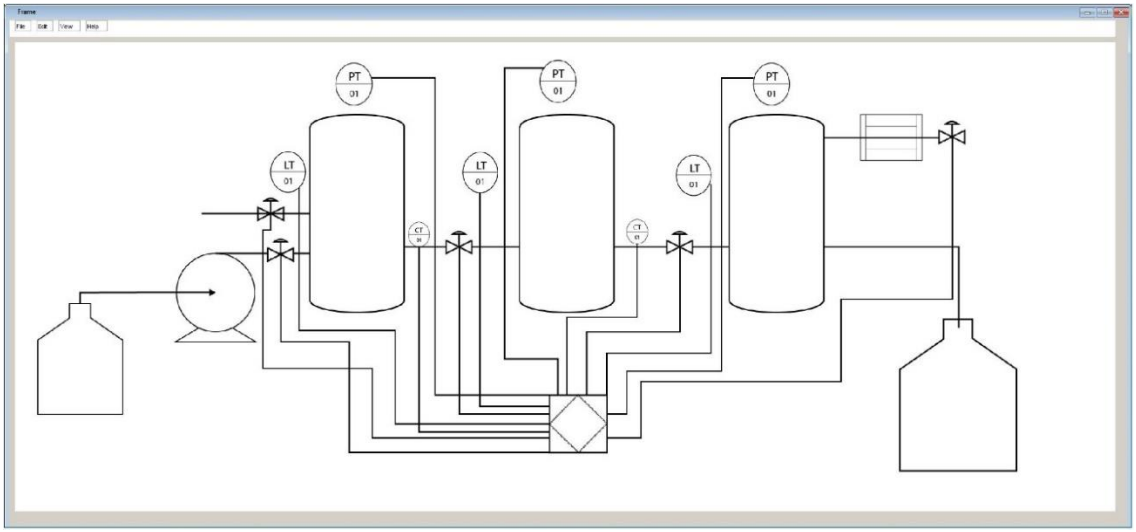
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá visualizar los componentes en el HMI Ejecución con la configuración realizada en el editor.	
Prototipo de interfaz: 	

Figura14. Prototipo de interfaz para el requisito 6

Tabla 10. Historia de usuario del RF7

Historia de usuario	
Número: 7	Nombre del requisito: Modificar el estado de la instancia del componente al modificar el valor de la variable asignada en el editor y HMI Ejecución del sistema AREX.

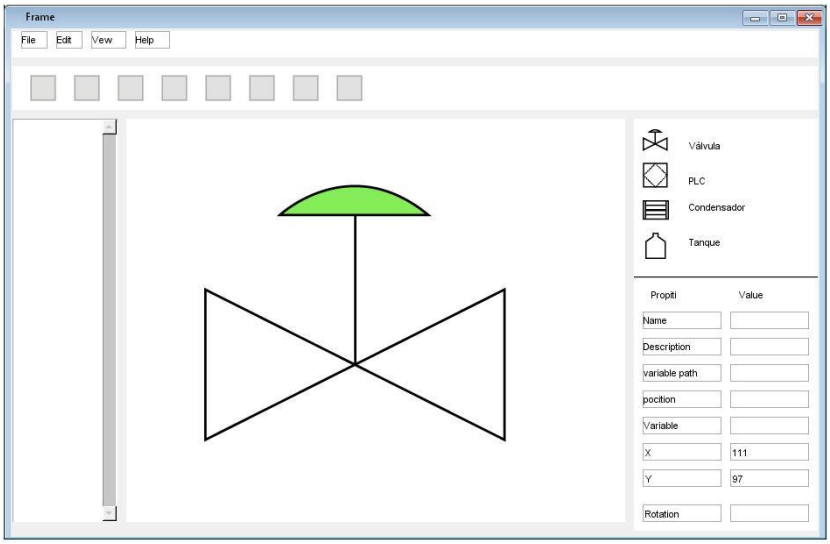
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá modificar el estado del componente asignándole un valor (si lo admite).	
Prototipo de interfaz: 	

Figura15. Prototipo de interfaz para el requisito 7

Tabla 11. Historia de usuario del RF8

Historia de usuario

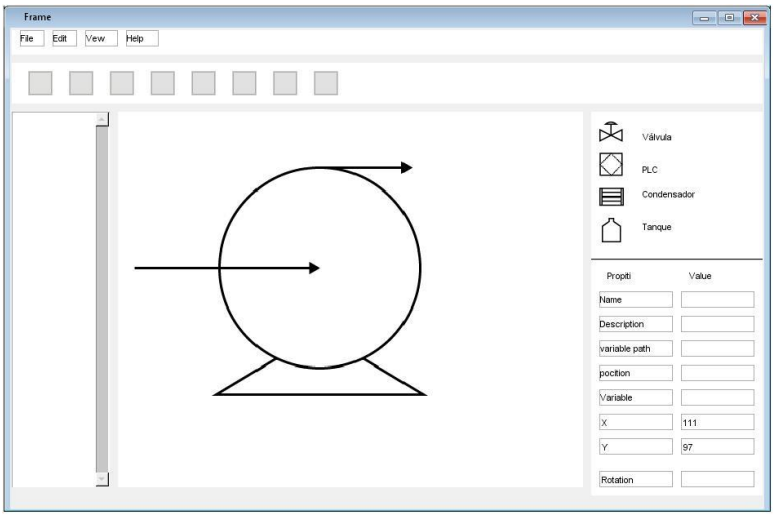
<table>
<tr><td>Número: 8</td><td colspan="2">Nombre del requisito: Editar las instancias de componentes en el inspector de propiedades del Editor del sistema AREX.</td></tr>
<tr><td colspan="2">Programador: Karla Martínez Carrera</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo estimado: 1 semana</td></tr>
<tr><td colspan="2">Riesgo en desarrollo: Problemas eléctricos y técnicos.</td><td>Tiempo real: 1 semana</td></tr>
<tr><td colspan="3">Descripción de la operación: El sistema permitirá modificar las propiedades del componente en el inspector de propiedades que se encuentra en el menú lateral.</td></tr>
<tr><td colspan="3">Prototipo de interfaz:
 

Figura16. Prototipo de interfaz para el requisito 8</td></tr>
</table>

Tabla 12. Historia de usuario del RF9

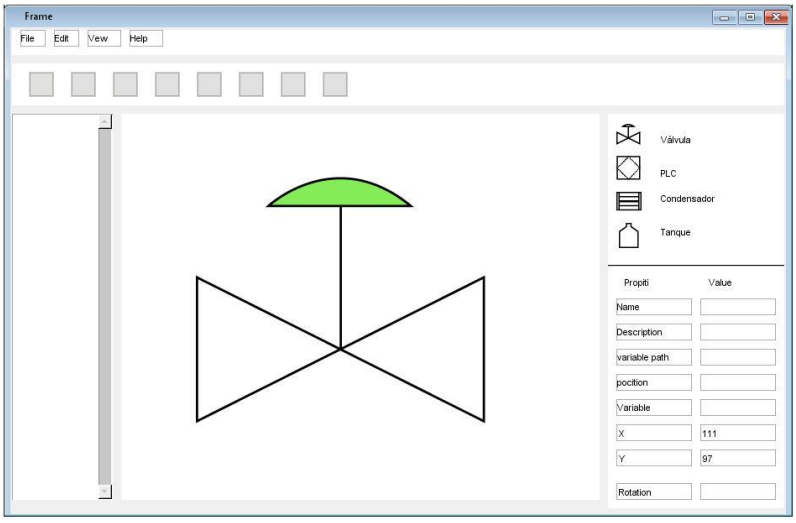
<table>
<tr><td colspan="3">Historia de usuario</td></tr>
<tr><td>Número: 9</td><td colspan="2">Nombre del requisito: Definir estado en el inspector de propiedades para el componente válvula.</td></tr>
<tr><td colspan="2">Programador: Karla Martínez Carrera</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo estimado: 1 semana</td></tr>
<tr><td colspan="2">Riesgo en desarrollo: Problemas eléctricos y técnicos.</td><td>Tiempo real: 1 semana</td></tr>
<tr><td colspan="3">Descripción de la operación: El sistema permitirá definir en el inspector de propiedades el estado del componente válvula.</td></tr>
<tr><td colspan="3">Prototipo de interfaz:
 

Figura 17. Prototipo de interfaz para el requisito 9</td></tr>
</table>

Tabla 13. Historia de usuario del RF10

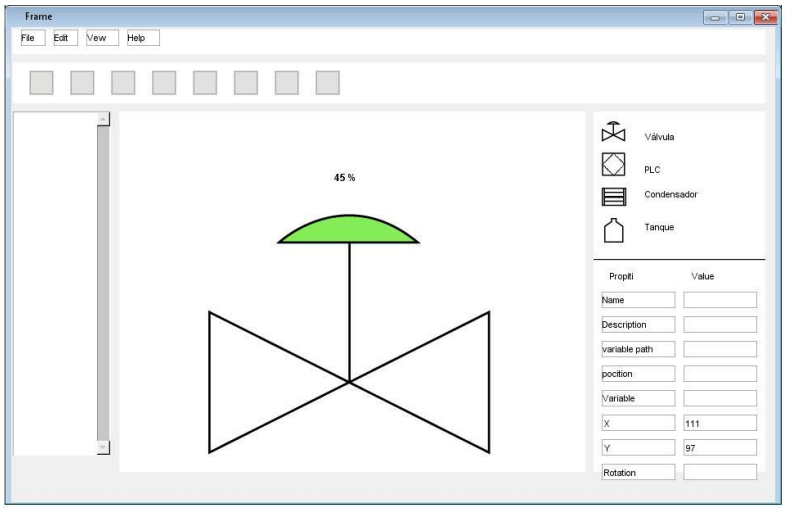
<table>
<tr><td colspan="3">Historia de usuario</td></tr>
<tr><td>Número: 10</td><td colspan="2">Nombre del requisito: Definir el porciento de apertura en el inspector de propiedades para el componente electroválvula.</td></tr>
<tr><td colspan="2">Programador: Karla Martínez Carrera</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo estimado: 1 semana</td></tr>
<tr><td colspan="2">Riesgo en desarrollo: Problemas eléctricos y técnicos.</td><td>Tiempo real: 1 semana</td></tr>
<tr><td colspan="3">Descripción de la operación: El sistema permitirá definir en el inspector de propiedades el porciento de apertura para el componente válvula.</td></tr>
<tr><td colspan="3">Prototipo de interfaz:
 
</td></tr>
</table>

Figura 18. Prototipo de interfaz para el requisito 10

Tabla 14. Historia de usuario del RF11

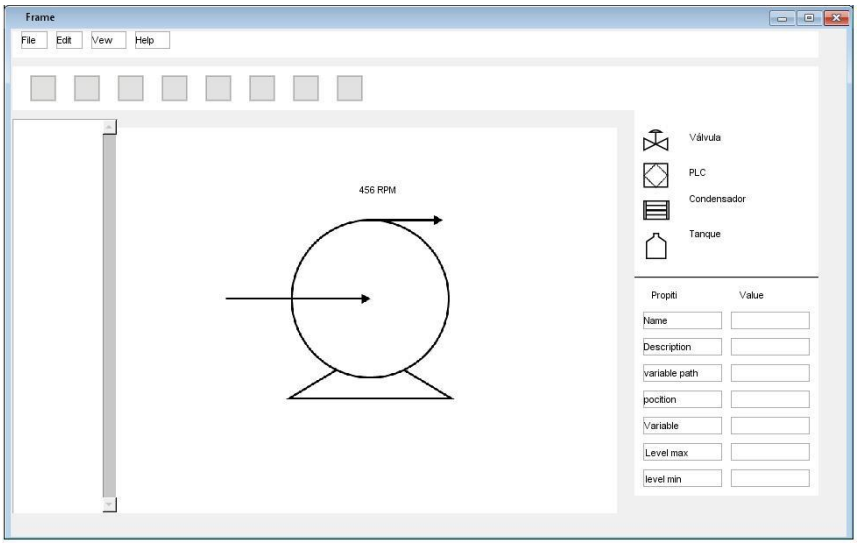
<table>
<tr><td colspan="3">Historia de usuario</td></tr>
<tr><td>Número: 11</td><td colspan="2">Nombre del requisito: Definir la velocidad angular en el inspector de propiedades para el componente Bomba centrífuga.</td></tr>
<tr><td colspan="2">Programador: Karla Martínez Carrera</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo estimado: 1 semana</td></tr>
<tr><td colspan="2">Riesgo en desarrollo: Problemas eléctricos y técnicos.</td><td>Tiempo real: 1 semana</td></tr>
<tr><td colspan="3">Descripción de la operación: El sistema permitirá definir velocidad angular en el inspector de propiedades para el componente Bomba centrífuga</td></tr>
<tr><td colspan="3">Prototipo de interfaz:
 
</td></tr>
</table>

Figura 19. Prototipo de interfaz para el requisito 11

Tabla 15. Historia de usuario del RF12

Historia de usuario	
Número: 12	Nombre del requisito: Definir el nivel máximo en el inspector de propiedades para el componente tanque.
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá definir el nivel máximo en el inspector de propiedades para el componente tanque.	
Prototipo de interfaz:	

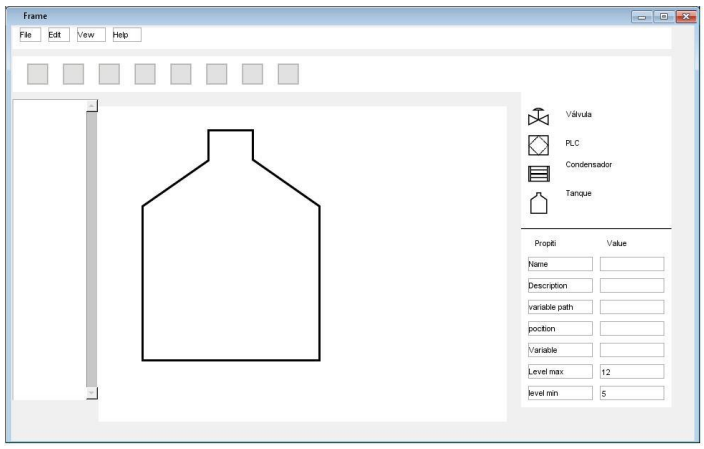


Figura 20. Prototipo de interfaz para el requisito 12

Tabla 16. Historia de usuario del RF13

Historia de usuario	
Número: 13	Nombre del requisito: Definir el nivel mínimo en el inspector de propiedades para el componente tanque.
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá definir el nivel mínimo en el inspector de propiedades para el componente tanque.	

Prototipo de interfaz:

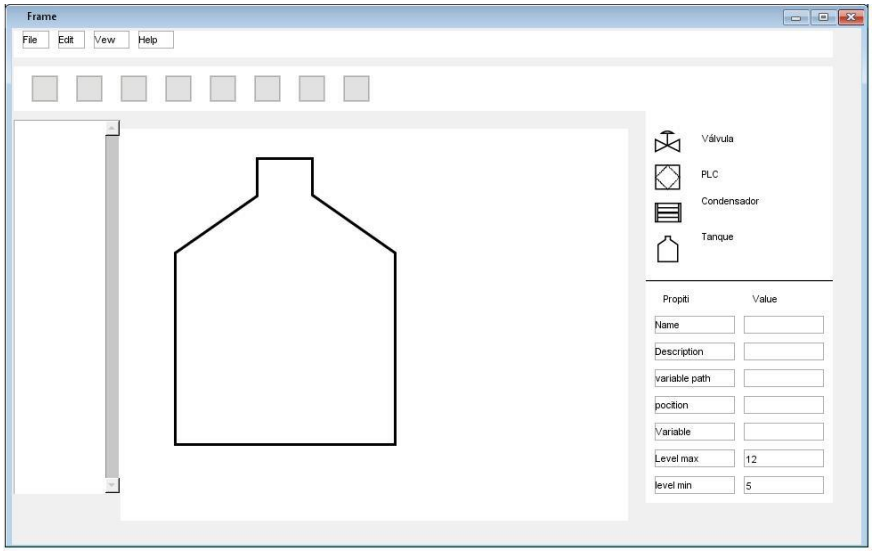


Figura 21. Prototipo de interfaz para el requisito 13

Tabla 17. Historia de usuario del RF14

Historia de usuario	
Número: 14	Nombre del requisito: Visualizar aumento de nivel en el componente tanque.
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana

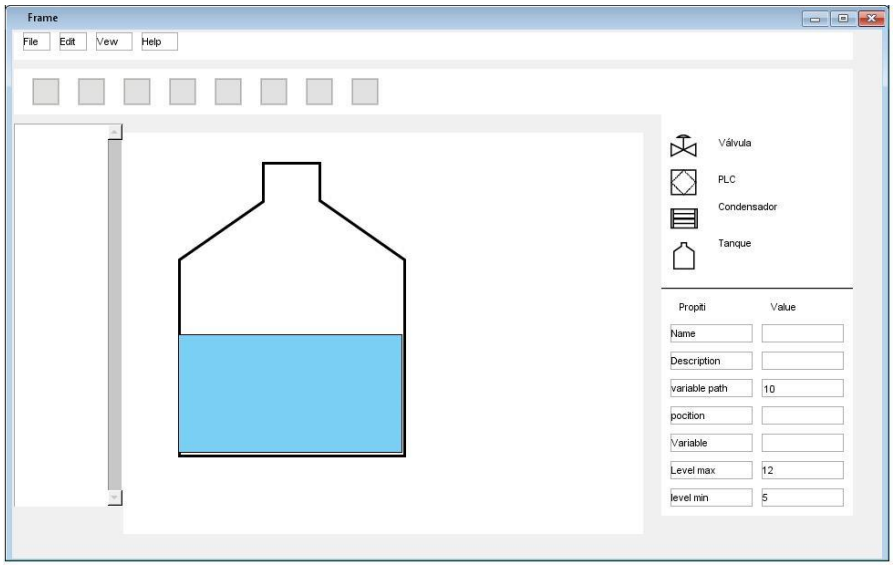
Descripción de la operación: El sistema permitirá visualizar el aumento de nivel en el componente tanque.
Prototipo de interfaz:  **Figura 22. Prototipo de interfaz para el requisito 14**

Tabla 18. Historia de usuario del RF15

Historia de usuario	
Número: 15	Nombre del requisito: Visualizar en los componentes Sensor de Presión, Sensor de temperatura, Sensor de Nivel y Sensor de concentración el valor de la variable asignada en el inspector.
Programador: Karla Martínez Carrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 1 semana

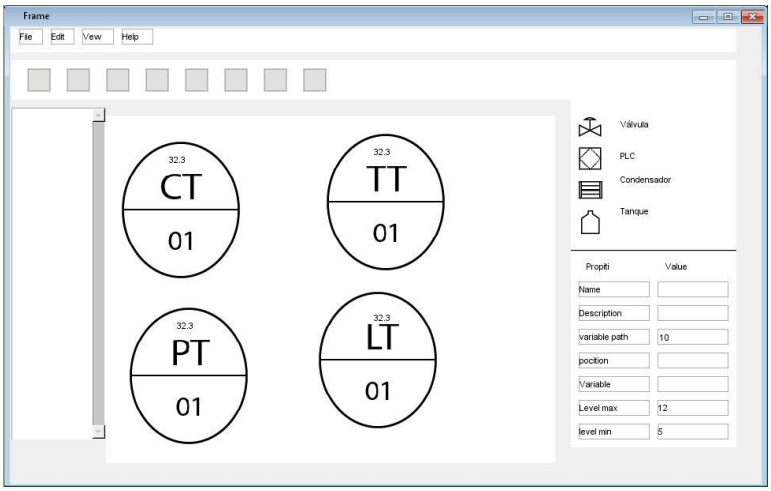
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá visualizar el valor de la variable asignada en el inspector.	
Prototipo de interfaz: 	

Figura 23. Prototipo de interfaz para el requisito 15

Tabla 19. Historia de usuario del RF16

Historia de usuario	
Número: 16	Nombre del requisito: Visualizar movimiento de aguja en componente Sensor de Presión.
Programador: Karla Martínez Carrera	Iteración asignada: 1

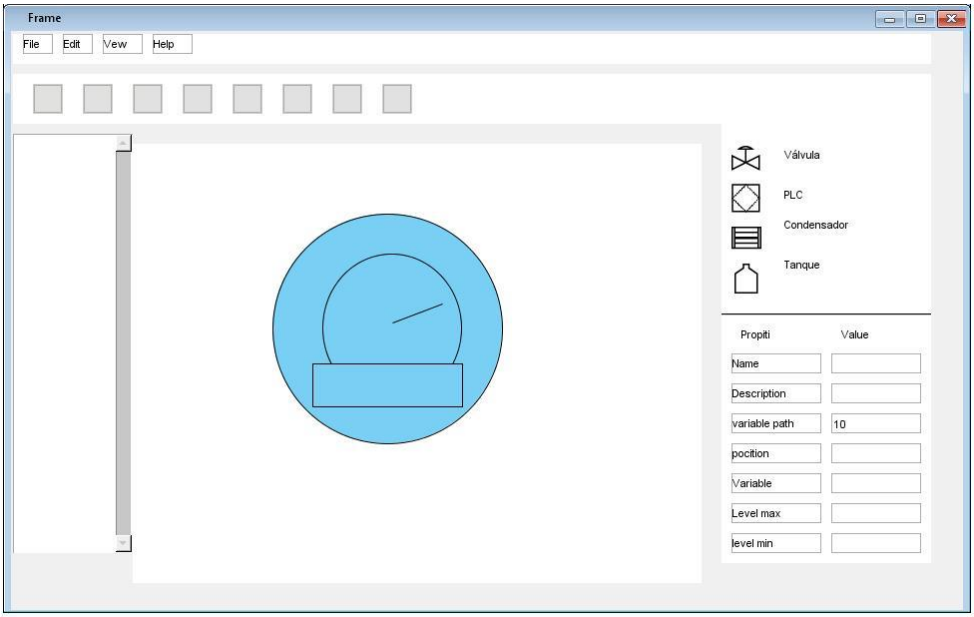
Prioridad: Alta	Tiempo estimado: 1 semana
Riesgo en desarrollo: Problemas eléctricos y técnicos.	Tiempo real: 1 semana
Descripción de la operación: El sistema permitirá visualizar el movimiento de la aguja en el componente Sensor de Presión.	
Prototipo de interfaz: 	

Figura 24. Prototipo de interfaz para el requisito 16

Anexo 2. Casos de prueba

Tabla 12 Caso de prueba

SC3 Expandir o contraer las dimensiones del componente en caso de poderse redimensionar en el HMI Editor del sistema AREX.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 3.1 Expandir o contraer las dimensiones del componente	1 Siguiendo la ruta del flujo central se da clic sostenido en las esquinas del componente y se arrastra para contraerlo y expandirlo.	1 Permite seleccionar el componente en el despliegue. 2 Aparecen las esquinas azules 3 Permite dar clic sostenido en las esquinas azules 4 El componente se contrae y se expande	Siguiendo la siguiente ruta: Se selecciona el componente en el despliegue Se da clic sostenido en las esquinas azules que aparecen en el componente y se arrastra para contraerlo y expandirlo.
SC4 Asignarle una variable a una instancia de componente en el HMI Editor del sistema AREX.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 4.1 Asignarle una	1 Siguiendo la ruta del flujo	1 Permite asignar una variable en el	Siguiendo la siguiente ruta:

variable a una instancia de componente	central se asigna un valor en el inspector de propiedades.	inspector de propiedades.	En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. En la sección "*value*" se asigna un valor.
SC5 Modificar el valor de la variable asignada al ejecutar eventos del mouse sobre la instancia del componente en el HMI Editor y HMI Ejecución del sistema AREX.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 5.1 Modificar el valor de la variable asignada al ejecutar eventos del mouse sobre la instancia del componente.	Siguiendo la ruta del flujo central se hace clic en la instancia del componente.	1 Modifica la instancia.	Siguiendo la siguiente ruta: Se da clic en el componente.
SC6 Visualizar el componente en el HMI Ejecución del sistema AREX conforme a la configuración de propiedades realizadas en el HMI Editor.			

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC6.1 Visualizar el componente en el HMI Ejecución	Siguiendo la ruta del flujo central se selecciona el despliegue configurado en el HMI Editor.	1 Se visualizan los componentes configurados en el HMI Editor.	Siguiendo la siguiente ruta: Se selecciona el despliegue configurado en el HMI Editor.
SC7 Modificar el estado de la instancia del componente al modificar el valor de la variable asignada en el HMI Editor y HMI Ejecución del sistema AREX.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
Modificar el estado de la instancia del componente al modificar el valor de la variable asignada	1 Siguiendo la ruta del flujo central se asigna un valor en el inspector de propiedades.	1 Permite modificar el valor de la variable. 2 Modifica la instancia.	Siguiendo la siguiente ruta: En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. En la sección "*value*" se asigna un valor.

SC 8 Editar las instancias de componentes en el inspector de propiedades del HMI Editor del sistema AREX.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
Editar las instancias de componentes en el inspector de propiedades	1 Siguiendo la ruta del flujo central en la sección "*Rotation*" se asigna un valor de 90.	1 Se rota el componente.	Siguiendo la siguiente ruta: En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. En la sección "*Rotation*" se asigna un valor de -90 a 90.
SC9 Definir porciento de apertura en el inspector de propiedades para el componente electroválvula.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 9.1 Definir porciento de apertura en el inspector de propiedades para el	1 Siguiendo la ruta del flujo central se selecciona el	1 Permite asignar un valor en el inspector de propiedades.	Siguiendo la siguiente ruta: Se selecciona el componente electroválvula.

componente electroválvula.	componente electroválvula. 2 Se asigna un valor de 50.		En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. En la sección “porciento de apertura” se asigna un valor de 1 a 100.
SC10 Definir estado en el inspector de propiedades para el componente electroválvula.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC10.1 Definir estado en el inspector de propiedades para el componente electroválvula.	1 Siguiendo la ruta del flujo central se selecciona el componente electroválvula. 2 Se selecciona el estado (Abierto)	1 Se visualiza el estado en el componente.	Siguiendo la siguiente ruta: Se selecciona el componente electroválvula. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. Se selecciona el estado (Abierto o Cerrado)

EC10.2	1 Siguiendo la ruta del flujo central se selecciona el componente electroválvula. 2 Se selecciona el estado (Cerrado)	1 Se visualiza el estado en el componente.	Siguiendo la siguiente ruta: Se selecciona el componente electroválvula. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. Se selecciona el estado (Abierto o Cerrado)
SC 11 Definir la velocidad angular en el inspector de propiedades para el componente Bomba centrífuga.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
Definir la velocidad angular en el inspector de propiedades para el componente Bomba centrífuga.	1 Siguiendo la ruta del flujo central se selecciona el componente Bomba centrífuga. 2 Se asigna un valor de 100.	1 Permite asignar una velocidad angular en el inspector de propiedades.	Siguiendo la siguiente ruta: Se selecciona el componente Bomba centrífuga. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. En la sección “velocidad angular” se asigna un valor.

SC12 Definir el nivel máximo en el inspector de propiedades para el componente tanque.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 12.1 Definir el nivel máximo en el inspector de propiedades para el componente tanque.	1 Siguiendo la ruta del flujo central se selecciona el componente Tanque. 2 Se asigna un valor máximo de 500.	1 Permite asignarle el valor máximo al componente.	Siguiendo la siguiente ruta: Se selecciona el componente tanque. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. En la sección *"Level_max"* se asigna un valor.
SC13 Definir el nivel mínimo en el inspector de propiedades para el componente tanque.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**

EC 13.1 Definir el nivel mínimo en el inspector de propiedades para el componente tanque.	1 Siguiendo la ruta del flujo central se selecciona el componente Tanque. 2 Se asigna un valor mínimo de 20.	1 Permite asignarle el valor mínimo al componente.	Siguiendo la siguiente ruta: Se selecciona el componente tanque. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. En la sección *"Level_min"* se asigna un valor.
SC 14 Visualizar animación de aumento de nivel en el componente tanque.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC14.1 Visualizar animación de aumento de nivel en el componente tanque	1 Siguiendo la ruta del flujo central se selecciona el componente tanque. 2 Se modifica el valor del componente.	1 Permite modificar el valor del componente tanque. 2 Se visualiza la animación.	Siguiendo la siguiente ruta: Se selecciona el componente tanque. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. Se modifica el valor del componente.

SC15 Visualizar en los componentes Sensor de Presión, Sensor de temperatura, Sensor de nivel y Sensor de concentración el valor de la variable asignada en el inspector.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC15.1 Visualizar en el componente Sensor de Presión, el valor de la variable asignada en el inspector.	1 Siguiendo la ruta del flujo central se selecciona el componente sensor de presión. 2 Se modifica el valor a 56	1 Permite seleccionar el componente 2 Permite modificar el valor de la variable. 3 Se visualiza el valor en el componente.	Siguiendo la siguiente ruta: Se selecciona el componente sensor de presión. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. Se modifica el valor de la variable.
EC15.2 Visualizar en el componente Sensor de temperatura, el valor de la variable	1 Siguiendo la ruta del flujo central se selecciona el componente Sensor de temperatura.	1 Permite seleccionar el componente 2 Permite modificar el valor de la variable.	Siguiendo la siguiente ruta: Se selecciona el componente Sensor de temperatura. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. Se modifica el valor de la variable.

asignada en el inspector.	2 Se modifica el valor a 70	3 Se visualiza el valor en el componente.	
EC15.3 Visualizar en el componente Sensor de nivel el valor de la variable asignada en el inspector.	1 Siguiendo la ruta del flujo central se selecciona el componente Sensor de nivel. 2 Se modifica el valor a 500	1 Permite seleccionar el componente 2 Permite modificar el valor de la variable. 3 Se visualiza el valor en el componente.	Siguiendo la siguiente ruta: Se selecciona el componente Sensor de nivel. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. Se modifica el valor de la variable.
EC15.4 Visualizar en el componente Sensor de concentración el valor de la variable asignada en el inspector.	1 Siguiendo la ruta del flujo central se selecciona el componente sensor de concentración. 2 Se modifica el valor a 30	1 Permite seleccionar el componente 2 Permite modificar el valor de la variable. 3 Se visualiza el valor en el componente.	Siguiendo la siguiente ruta: Se selecciona el componente sensor de concentración. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. Se modifica el valor de la variable.
SC16 Visualizar movimiento de la aguja en componente Sensor de Presión.			

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 16.1 Visualizar movimiento de la aguja en componente Sensor de Presión.	1 Siguiendo la ruta del flujo central se selecciona el componente sensor de presión. 2 Se modifica el valor a 90	1 Permite seleccionar el componente 2 Permite modificar el valor de la variable. 3 Se visualiza el movimiento de aguja en componente.	Siguiendo la siguiente ruta: Se selecciona el componente sensor de presión. En la parte inferior derecha del HMI Editor se encuentra el inspector de propiedades. Se modifica el valor de la variable.