Universidad de las Ciencias Informáticas

Facultad 5

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Título: Módulo Ejecución en la web para el Sistema de Adquisición Arex.

Autores: José Carlos Reyes Azcuy

Laura Suárez Díaz

Tutores: Ing. Jordanis Viltres Chávez

Ing. Naivy Pujol Méndez

Cuidad de La Habana, julio 2016.

MINISTERIO DE EDUCACIÓN SUPERIOR UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Autorizamos a dicho centro para que haga el uso que estime pertinente con este trabajo.

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

José Carlos Reyes Azcuy

_____

Firma del autor

Laura Suárez Díaz

_____

Firma del autor

Ing. Jordanis Viltres Chávez

_____

Firma del tutor

Ing. Naivy Pujol Méndez

_____

Firma del tutor

DATOS DE CONTACTO

Tutor: Ing. Jordanis Viltres Chávez.

Edad: 35

Ciudadanía: cubano.

Institución: Instituto Superior Politécnico José Antonio Echeverría (CUJAE).

Título: Ingeniero en Informática.

Categoría docente: Trabajador.

Cargo: Especialista “B” en Ciencias Informáticas.

E-mail: jviltres@uci.cu

Tutor: Ing. Naivy Pujol Méndez

Edad: 25

Ciudadanía: cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría docente: Trabajador.

Cargo: Recién graduado en adiestramiento.

E-mail: npujol@uci.cu

AGRADECIMIENTOS

Es usual comenzar los agradecimientos diciendo: quisiera agradecerle a fulanito por… y a menganito por haber…Hasta cierto punto no es justo, porque siempre aparece un esperancejo que fue olvidado en el listado. Para que ello no suceda hemos decidido no mencionar nombres y hacerlo de la siguiente manera:

Quisiéramos agradecerles:

- *A esas personas que se encargaron de decidir cuál sería la primera palabra que diríamos e incluso compitieron entre ellos para que la suya fuera la selecciona.*
- *A quienes nos protegieron de los regaños, porque nos hicieron sentir a salvo.*
- *A quienes nos regañaron, porque nos enseñaron que es disciplina.*
- *A quienes hicieron uso de la escolástica y nos tendieron un papel para que repitiéramos cada una de las palabras del vocabulario, porque esa fue otra manera de aprender a escribirlas correctamente.*
- *A quienes fueron cómplices en bromas, malas, buenas, divertidas o peligrosas, porque a fin de cuentas son un grato recuerdo que vivirá con anhelo en nuestra memoria.*
- *A quienes nos incitaron a hacer cosas buenas o malas, colectas o incorrectas, porque así aprendimos que el límite que divide a unas de otras es una línea muy fina, casi imperceptible.*

- *A quienes llegaron un día y dijeron: "No estudiemos para la prueba, mejor vamos al concierto", porque así aprendimos que el destino puede ser incierto.*
- *A quienes de una forma u otra han formado parte de nuestra vida.*
- *Y no por último menos importante queríamos hacerles un agradecimiento especial a todas las personas han influido en la realización y culminación de este trabajo de diploma.*

DEDICATORIA

Dedicatoria de Laura Suárez Díaz:

A mi súper heroína favorita (Zoraida Olano Fernández):

No era como Iron Man, con un súper traje o Wonder Woman, con súper poderes atrapando a los villanos que querían destruir el mundo, pero era todo un genio en la cocina. No era profesora de historia, pero sabía que me daría una larga conferencia de la asignatura cuando decía la frase: "En mis tiempos…". No era militar, pero era toda una general dictando órdenes, y no fue hasta mucho tiempo después que descubrí que era su manera de enseñarme disciplina. No era una gran conocedora de arte, pero me enseñó que lo bueno no pasa de moda, recuerdo que era una tortura Palmas y Cañas, ahora extraño no escuchar el guateque campesino a la 7 de la noche el domingo o un disco de Nat King Cole sonando con sus grandes éxitos. A veces solía decirme: "Estudia Laurita, para que seas alguien en la vida", y aquí estoy, graduándome de ingeniera, y es una pena que no esté para verme…Abuela.

Dedicatoria de José Carlos Reyes Azcuy:

Quiero dedicarles este trabajo a tres personas muy especiales (mi madre, Leticia, mi abuela Angela y mi tía Mayelin) por darlo todo por mí, por apoyarme en todo momento y brindarme la confianza necesaria para seguir adelante en todo momento de mi vida, gracias a ellas estoy aquí y por ellas seguiré adelante. Espero que este sueño que se hace realidad les parezca tan feliz como a mí, las amo.

RESUMEN

En la Universidad de las Ciencias Informáticas, específicamente en el Centro de Informática Industrial, está en proceso de desarrollo el Sistema de Adquisición Arex. El cual es un software domótico encargado de supervisar el estado de las luces, de la humedad, de la temperatura, en edificios pequeños, una habitación o un hogar, propiciando así confort humano. El Sistema de Adquisición Arex está compuesto por varios módulos, entre los cuales se encuentra el de Ejecución; encargado de permitir la visualización e interacción del usuario con los mímicos y sus variables y alarmas asociadas. Este módulo sólo puede ser ejecutado como aplicación de escritorio, situación que afecta la disponibilidad y la competitividad del producto. Razón por la cual el presente trabajo tiene como objetivo diseñar e implementar un Módulo Ejecución para la web, para el Sistema de Adquisición Arex.

Palabras clave: HMI, mímicos, Módulo Ejecución, Sistema de Adquisición.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

En las últimas décadas del pasado siglo XX, para reducir costos operativos y aumentar la eficiencia, se aplicaba la automatización de los dispositivos electrónicos en los edificios del mundo industrializado. Con el rápido desarrollo de las Tecnologías de la información y las comunicaciones (TIC) se han puesto en práctica estrategias de automatización para el control ambiental en los diferentes locales de un edificio, elevando el confort humano a través del acceso a la red de voz, acceso a la red de datos, climatización, seguridad y megafonía.

Según la Universidad del País Vasco se define automatización como: *"(…) un sistema donde se transfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos."* (1)

La automatización de un edificio está basada en el desarrollo de un sistema para supervisar a distancia y en tiempo real los parámetros eléctricos, la calidad y necesidad real de la energía, iluminación, humedad, control de la ventilación, logrando así calidad en las prestaciones de servicios, confort y prolongar la vida útil de los equipos. Todos los aspectos anteriores se logran a través de instalaciones domóticas.

Según el Diccionario de Informática y Tecnología se define domótica como: *"(…) serie de sistemas integrados e interrelacionados que se instalan en un hogar y que permiten la automatización del mismo y su control tanto desde adentro de la casa como desde afuera".* (2)

En el mundo, los sistemas domóticos son usados en la actualidad, por ejemplo:

- En el Instituto del Mundo Árabe (*L'Institut du Monde Arabe*) de París diseñado por Jean Nouvel. En el que la fachada se convierte en un elemento cambiante ante el nivel de luz del exterior (3).
- La empresa *Edisa Telecomunicacions* ha realizado el proyecto y la instalación del Chalet en Platja d'Aro (Gerona), donde el sistema controla la iluminación, las persianas motorizadas y la zona de home cinema. (4)

En el proceso de la automatización son utilizados los Sistemas de Control, Supervisión y Adquisición de Datos (SCADA). Estos sistemas están diseñados para funcionar sobre ordenadores de producción, proporcionando comunicación con los dispositivos autómatas programados. Según José E. Briseño Márquez se define SCADA cómo: *"(…) una tecnología que permite obtener y procesar información de procesos industriales dispersos o lugares remotos inaccesibles, transmitiéndola a un lugar para supervisión, control y procesamiento, normalmente una Sala o Centro de Control."* (5)

Los sistemas SCADA pueden ser divididos en varios niveles. Las funcionalidades de los niveles son las siguientes:

1. Nivel de Campo: También conocido como nivel de Planta, Proceso o Sensor. Está formado por los sistemas de medición y actuación de procesos. En este nivel se encuentran sensores e instrumentación de campo que proveen la información al nivel de control.
2. Nivel de Control y Regulación de Procesos o de Adquisición: Su principal función es muestrear el estado del nivel de campo. La información adquirida en el muestreo es utilizada para aplicar leyes de control para la regulación estable de procesos.
3. Nivel de Supervisión y Control o Mando de Grupos: Es el encargado de supervisar y controlar los reguladores distribuidos en la red. En este la información es transportada hacia los niveles superiores.
4. Nivel de Dirección de la Producción: Es el encargado de procesar la información de los niveles inferiores para trazar estrategias de producción del nivel macro. Estas estrategias ejercen acciones en la gestión de los niveles inferiores para ajustar la producción en función de variables de proceso y negocio. (6)

En la siguiente imagen se muestra como quedan distribuidos los niveles en la pirámide de automatización.

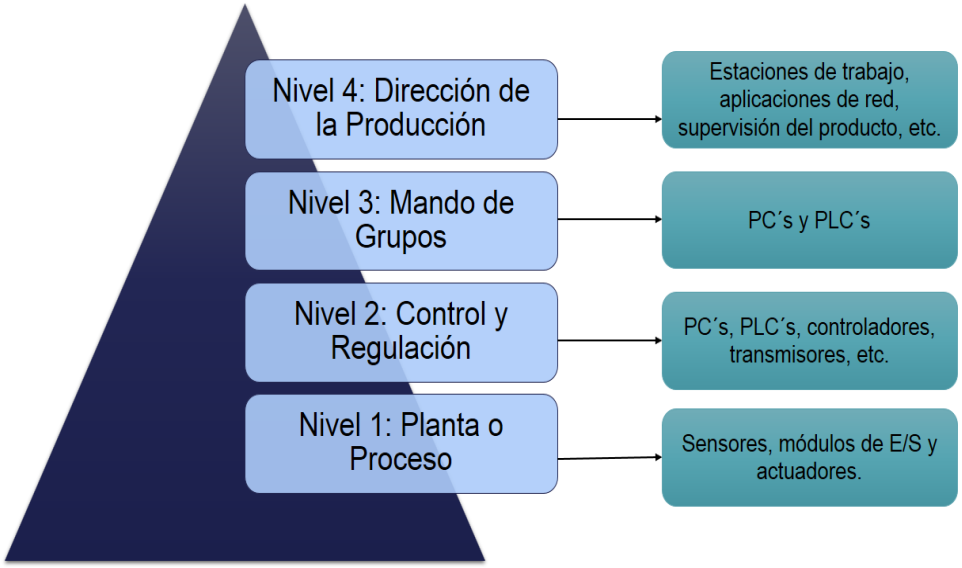


Ilustración 1: Pirámide de Automatización de 4 Niveles. (6)

En el mundo se han desarrollado numerosos productos SCADA con aplicación tanto en grandes como en pequeñas industrias. Ejemplo de ello es el SCADA SICLE **(**Sistema de Información y Control Local de Estación), desarrollado en México para la empresa CFE (Comisión Federal de Electricidad). Es un tipo de SCADA definido de manera muy concreta, con especificaciones avanzadas, especialmente diseñado para la industria eléctrica de potencia.

En Cuba también se han desarrollado productos SCADA, tal es el caso del EROS, un Sistema de Supervisión y Control de procesos diseñado y desarrollado en la División de Automatización SERCONI (Servicios Técnicos de Computación, Comunicaciones y Electrónica). Se instaló por primera vez en la empresa niquelífera Ernesto Che Guevara en Moa. EROS es un Sistema de Supervisión y Control de Procesos Industriales para operar y dirigir cualquier proceso, trabajando acoplado con diversos sistemas de colección de datos (autómatas programables, estaciones de adquisición de datos y reguladores autónomos). Además de la industria niquelífera, EROS se extiende a otros sectores como las empresas azucareras y salineras, instalaciones turísticas, entidades camaroneras, hidroenergéticas y acueductos, entre otros. (7)

También en Cuba, en la Universidad de las Ciencias Informáticas (UCI), en el Centro de Informática Industrial (CEDIN) perteneciente a la Facultad 5 se han desarrollado diferentes productos SCADA como es el caso de SAINUX (Sistema de Automatización Industrial). Es un sistema que permite

controlar y monitorear mediante un ordenador y a través de dispositivos de campo todas las operaciones del proceso. También hace posible almacenar sus datos para realizar análisis, con los cuales se pueden obtener importantes indicadores que permite la retroalimentación de procesos (8). El sistema puede ser aplicado a cualquier proceso que necesite automatización

En la UCI, está en proceso de desarrollo el mini SCADA Arex. Es un Sistema de Control y Adquisición de datos de procesos domóticos, razón por la cual está ubicado en el segundo nivel de la pirámide de automatización y es clasificado como un Sistema de Adquisición.

El Sistema de Adquisición Arex es un software que tiene como objetivo automatizar infraestructuras pequeñas (dígase edificios, oficinas o viviendas). La función principal del producto es detectar el estado de las luces, el comportamiento de la humedad y la temperatura en un lugar determinado, aportando así servicios de gestión energética, confort y seguridad. Está destinado a ser aplicado en sectores como hotelería y pequeñas industrias.

El HMI[1] del Sistema de Adquisición Arex solo puede ser ejecutado como aplicación de escritorio, en dispositivos móviles como aplicación y en dispositivos de bajas prestaciones como la computadora cubana CID/300.

Las principales dificultades del Módulo Ejecución del Sistema de Adquisición Arex es que como condición necesaria para ser ejecutado en un dispositivo tiene que contar con la aplicación instalada, y no cuenta con un HMI que permita el acceso desde la web. Situación que afecta la disponibilidad del producto, ya que depende de una aplicación instalada en el dispositivo en el cual se va a ejecutar. El Módulo Ejecución cuenta con bibliotecas de componentes gráficos o mímicos[2], de estos componentes las animaciones fueron programadas mediante QML (*Qt Meta Language),* de las cuales se necesita una variante para ser interpretada a través de un navegador web. Tales circunstancias generan como situación problemática que la cantidad de dispositivos en los que se empleará el Módulo Ejecución sea limitada, lo cual afecta la competitividad del producto en el

[1] Interfaz Hombre – Máquina (HMI por sus siglas en inglés).

[2] Objetos animados

mercado, pues las tendencias actuales de los sistemas que tienen HMI es tener acceso haciendo uso de la tecnología web.

Problema de la investigación:

¿Cómo lograr la visualización e interacción con el Sistema de Adquisición Arex mediante la web utilizando una selección de la biblioteca de mímicos?

Objeto de estudio:

Sistemas HMI.

Objetivo general:

Desarrollar una variante web del Módulo Ejecución del Sistema de Adquisición Arex.

Campo de acción:

HMI Web para sistemas de adquisición.

Tareas de investigación:

1. Revisión bibliográfica para generar el marco teórico conceptual de la investigación que represente las tendencias actuales en los HMI web[3].
2. Entrevistas a especialistas del Sistema de Adquisición Arex con el objetivo de realizar el levantamiento de información necesario para la comprensión de los procesos que ocurren dentro del mismo.
3. Investigación sobre el estado arte de los HMI.
4. Identificación de las herramientas y los sistemas existentes con los que se necesite integrar el sistema a desarrollar.
5. Caracterización del Sistema de Adquisición Arex.

[3] Interfaz Hombre – Máquina que aceptan entradas y proporcionan salidas mediante la web. (12)

6. Selección y estudio de metodologías, herramientas y tecnologías a usar durante el proceso de desarrollo.
7. Realización del análisis y diseño de la solución propuesta.
8. Implementación de la solución propuesta con las funcionalidades definidas.
9. Realización de pruebas a la solución propuesta.
10. Desarrollo de la documentación asociada a la metodología de desarrollo AUP-UCI en el escenario seleccionado, que recoja el proceso de desarrollo del sistema.
11. Redacción del informe que contenga el proceso investigativo definido.

En el transcurso del proceso investigativo para generar el marco conceptual de la investigación, así como el estado del arte de los HMI web fue necesario el empleo de métodos y técnicas de búsqueda de información. Los cuales son descritos a continuación:

- Método histórico-lógico: Se utilizó para realizar un estudio de los antecedentes del Sistema de Adquisición Arex, así como de las herramientas de desarrollo vigentes para lograr un sistema que cumpla con las exigencias y necesidades definidas.
- Método analítico-sintético: Se utilizó para hacer un estudio de la bibliografía referente a HMI Web de sistemas de Adquisición y así extraer elementos puntuales para darle solución al problema.
- Entrevista: A través de la entrevista se pudo obtener información sobre las experiencias en el desarrollo de sistemas similares en la universidad.
- Observación: Utilizado para conocer la realidad del producto mediante la observación directa de los procesos y lograr una investigación en función de los objetivos.

Estructura de los capítulos:

Capítulo 1: Marco teórico de la investigación para desarrollar un HMI Web.

Se describen los aspectos fundamentales para conformar el marco teórico de la investigación teniendo en cuenta el objeto de estudio definido. Se hace un análisis del estado del arte de los HMI web y de las tecnologías existentes para el desarrollo de los mismos, así como la descripción de las herramientas y tecnologías empleadas en el desarrollo del software.

Capítulo 2: Propuesta de solución para el Módulo de Ejecución en la web del Sistema de Adquisición Arex.

Se describe la solución propuesta al problema de investigación. Se explica el funcionamiento del sistema a desarrollar, así como se definen los principales elementos de la documentación de mayor relevancia tras la aplicación de la metodología de desarrollo.

Capítulo 3: Implementación y pruebas de la solución propuesta.

Se implementa la solución propuesta en el capítulo anterior. Al resultado obtenido se le realizarán pruebas internas para validar el código y detectar posibles errores. También serán realizadas pruebas de aceptación para comprobar si los requisitos funcionales fueron correctamente implementados.

CAPÍTULO 1: MARCO TEÓRICO DE LA INVESTIGACIÓN PARA DESARROLLAR UN HMI WEB.

INTRODUCCIÓN

En este capítulo se describen algunos elementos para conformar el marco teórico de la investigación utilizando los aspectos definidos en el objeto de estudio. Se analizan los Sistemas de Adquisición de Datos en general y particularmente se hace una investigación sobre las peculiaridades de la arquitectura del Sistema de Adquisición Arex para llevar a cabo una correcta integración con el HMI web. También se analizan los HMI en general y particularmente los que son de tipo web. Se abordan las definiciones, características y funcionalidades comunes entre los HMI Web para llegar a una conceptualización de los mismos. Se caracterizan y analizan las herramientas informáticas empleadas en la actualidad para el desarrollo web.

1.1 SISTEMA DE ADQUISICIÓN

En la actualidad el desarrollo de la electrónica ha propiciado la automatización de la vida humana. Este avance en la ciencia ha sido posible a través del desarrollo de sistemas como los de adquisición de datos.

"Los sistemas de adquisición de datos, como su nombre indica, son los productos y/o procesos utilizados para recopilar información para documentar o analizar un fenómeno." (9)

Según la Universidad Sonora de México se define Sistema de Adquisición de datos como: *"(…) cualquier sistema que permita capturar (leer, medir) datos, almacenarlos, procesarlos y exhibirlos en alguna forma."* (10)

Después de analizar y extraer los elementos fundamentales de las definiciones, se define un sistema de adquisición de datos para la presente investigación como: Sistema mediante el cual se recopila, muestrea o mide información para ser analizada o documentada, utilizando sensores y hardware de medidas. (Elaboración propia)

Se puede concluir que la diferencia entre un Sistema de Adquisición y un SCADA es que además de recopilar, muestrear o medir la información para ser analizada o documentarla, en un SCADA esta información es transmitida a un lugar para ser supervisada, procesada y controlada.

Un sistema de adquisición cuenta con diferentes módulos para ejecutar su principal función que es el muestreo de información a la cual se le aplican leyes de control[4]. Estos módulos son:

- Manejadores de protocolo (Driver): Se encargan de la adquisición de datos traduciendo los protocolos de campo a un protocolo genérico. Pueden provenir de diferentes dispositivos como autómatas programables, reguladores de autómatas, sensores inteligentes y controladores.
- Módulo de recolección: Es el responsable de construir la lista de los dispositivos de campos a consultar, agrupándolos según la frecuencia de recolección. La lista es entregada a los manejadores para que construya los bloques de encuesta.
- Base de datos: Encargado de manejar todo lo referente a recepción, procesamiento y distribución de los datos provenientes del campo. (6)

1.1.1 SISTEMA DE ADQUISICIÓN AREX

El Sistema de Adquisición Arex cuenta con los módulos mencionados en el acápite anterior, pero estos módulos son genéricos para un Sistema de Adquisición como parte de un Sistema SCADA, pero Arex no llega a ser un Sistema SCADA, sino un software que cumple con los niveles de Planta o Proceso y el de Control y Regulación en la pirámide de automatización. Específicamente está compuesto por los siguientes módulos:

- Editor: Permite al usuario crear y configurar dispositivos y variables asociadas a los mismos, los cuales pertenecen a determinado despliegue que son creados y configurados también en el Editor. Asociadas a las variables están las alarmas, las cuales se disparan dependiendo de que la variable tome un valor configurado por el usuario. La configuración

[4] Son las encargadas de la regulación estable de los procesos.

establecida por el usuario puede ser exportada en un archivo XML para ser usada posteriormente por los módulos Recolector, Ejecución e Histórico.

- Recolector: Planifica y ejecuta encuestas a dispositivos basándose en la configuración de dispositivos y variables en el archivo XML. Posee un procesador de alarmas configuradas para cada variable. Funciona como servidor de datos adquiridos para los módulos Histórico y Ejecución. Las encuestas a dispositivos se realizan mediante el protocolo MODBUS en sus variantes TCP y RTU.
- Ejecución: Se conecta al Módulo Recolector solicitándole las variables y alarmas necesarias para visualizar los despliegues configurados en un archivo XML previamente creado y configurado en el Módulo Editor.
- Histórico: Se conecta al Módulo Recolector solicitándole todas las variables y alarmas, almacena los datos adquiridos en una base de datos, y de esta manera funciona como servidor de datos adquiridos para clientes que lo soliciten.
- Cliente de Baso de Datos Histórica: Se conecta al Módulo Histórico solicitando valores históricos de variables y alarmas. Visualiza los datos de las variables y las alarmas mediante gráficas y resúmenes.
- Adquisición: Módulo para la adquisición de datos que trabaja en la tarjeta Arduino Mega. Es el encargado de obtener y convertir señales de los sensores, además, enviar señales a actuadores. También recibe información, la cual brinda a los clientes mediante el protocolo MODBUS.

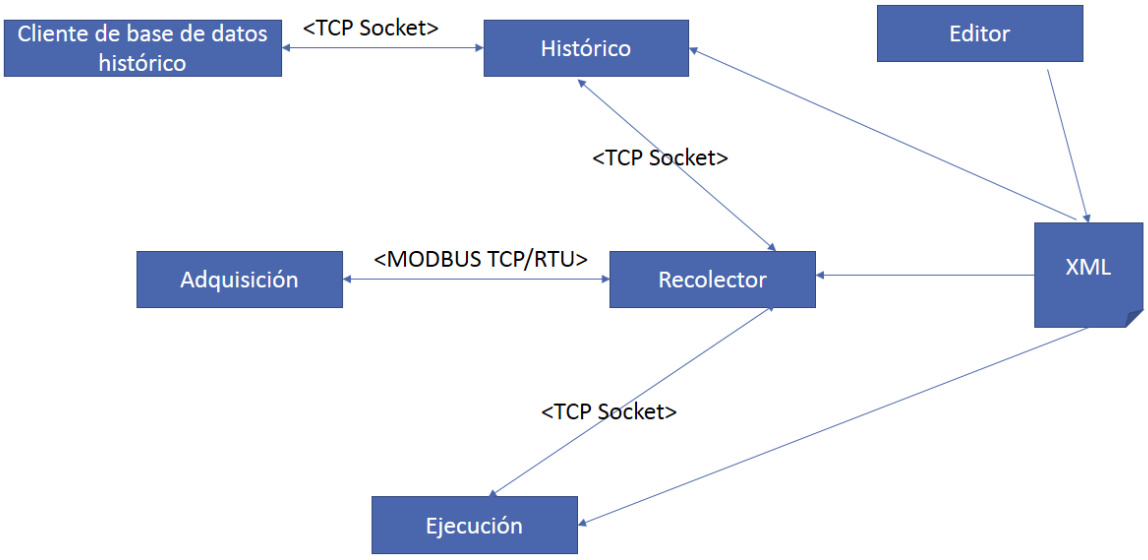


Ilustración 2: Relación entre los módulos del Sistema de Adquisición Arex.

1.2 INTERFAZ HOMBRE-MÁQUINA

La Interfaz Hombre Máquina, es utilizada comúnmente en los sistemas de computación en la actualidad, se define como*: "(…) capa intermedia que independiza a los seres humanos de las interioridades de los equipos electrónicos y digitales, permitiendo la intercomunicación entre ambas partes* (6)*".* Es donde se encuentran las personas y la tecnología, puede ser algo simple como controlar un horno microondas o tan complejo como el control y supervisión de una planta eléctrica.

Tradicionalmente estos sistemas consistían en paneles compuestos por indicadores y comandos, tales como luces pilotos, indicadores digitales y analógicos. En la actualidad es posible contar con sistemas HMI que permiten una conexión más sencilla y económica con el proceso o la máquina.

Los sistemas HMI son conocidos también como softwares de monitoreo y control de supervisión. Las señales de los procesos son conducidas al HMI a través de dispositivos como tarjetas de entrada/salida en la computadora, PLC's (controladores lógicos programables), RTU (unidades remotas de I/O (*In/Out*, en español Entrada/Salida)) o Driver (manejador). Todos los dispositivos antes mencionados deben tener una comunicación que entienda el HMI (11). En la siguiente imagen se muestra como estaría estructurada la arquitectura entre los dispositivos y los HMI.

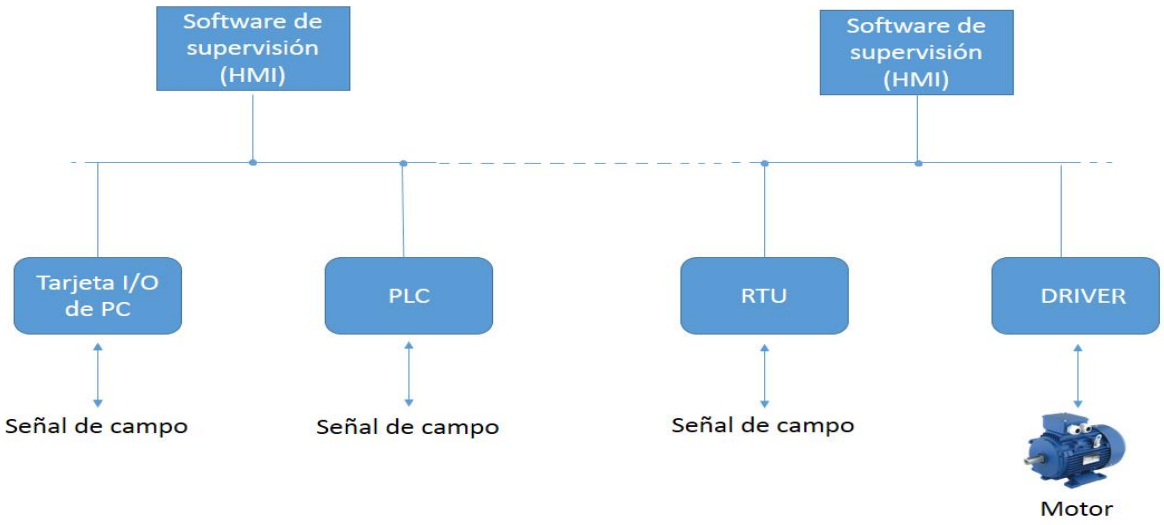


Ilustración 3: Comunicación entre el software de supervisión (HMI) y los dispositivos. (11)

Los softwares HMI están compuestos por programas y archivos. Algunos de estos programas son para el diseño y configuración del sistema, otros, son el motor del sistema. En la siguiente imagen se muestra como es la estructura general de un software HMI. Los programas como Editor de Pantalla y Editor de Base de Datos se encargan del diseño y configuración del sistema, otros como Interfaz Hombre (Pantalla, teclado y mouse), Driver y Archivo Base de Datos son el motor del HMI.

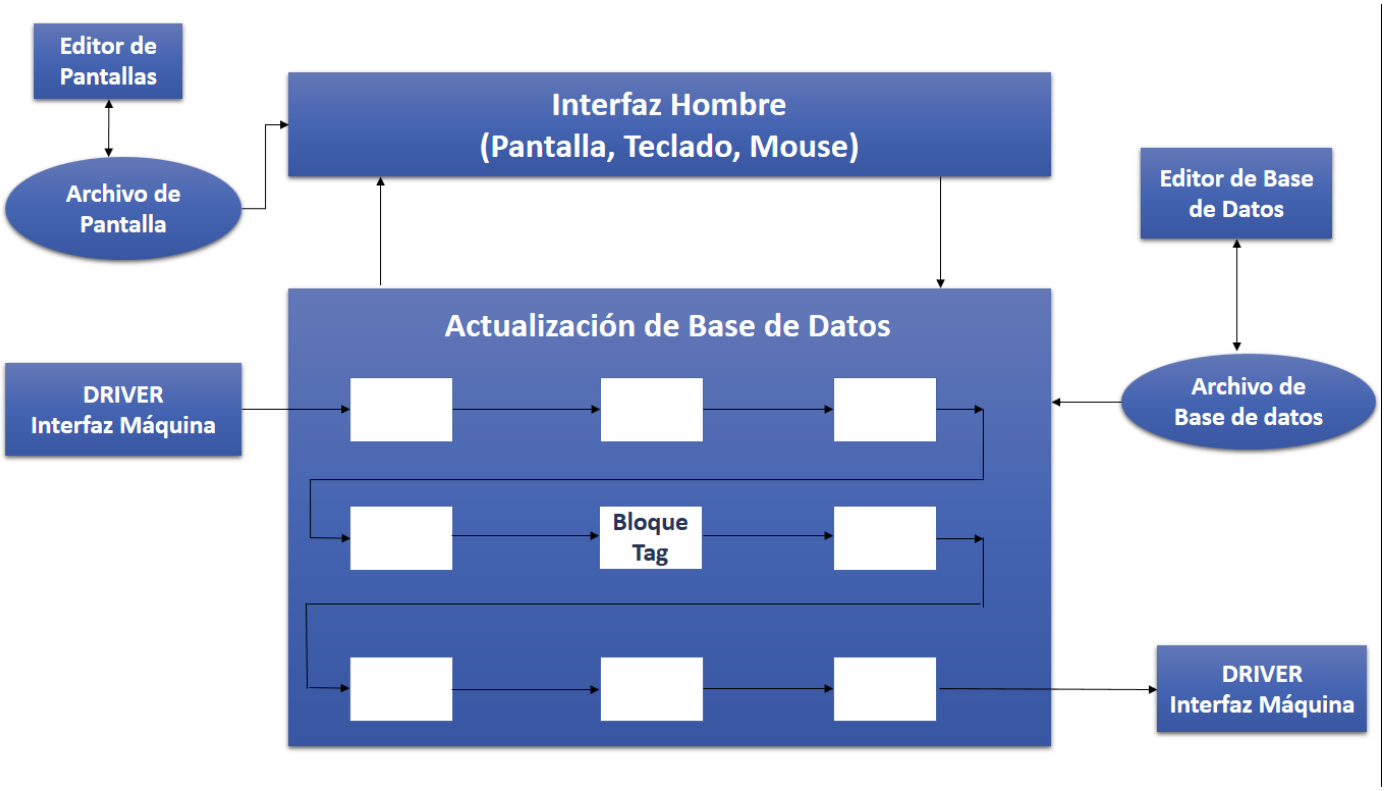


Ilustración 4: Estructura general de un software HMI (11).

Los programas como Editor de Pantallas posibilitan crear moldes[5] de pantallas para visualizar los datos del proceso. Los moldes son guardados en los Archivos de Pantalla y almacenan la forma en que serán visualizados los datos en las pantallas.

Interfaz Hombre Máquina: Programa que hace de interfaz entre la Base de Datos y el humano. Permite la interacción (supervisión y control) del recurso humano con las variables del sistema. El diseño de esta interfaz está definido en un Archivo Molde o de Pantalla, el cual debe estar previamente creado.

Base de Datos: Se encuentra en la memoria de una computadora y es donde se almacenan los datos requeridos de un proceso, los cuales varían en el tiempo, razón por la cual es considerada una base de datos dinámica. La base de datos está formada por bloques, los cuales pueden estar

[5] Prototipos.

interconectados. La creación de una base de datos, sus bloques y las relaciones entre ellos se realizan a través de Editor del Base de Datos.

Driver: La conexión entre los bloques de la base de datos y las señales del proceso se realiza por medio de drivers. Los cuales manejan también los protocolos de comunicación entre el HMI y los dispositivos de campo, por lo que son considerados la interfaz hacia la máquina.

Bloques (Tag[6]): Los bloques pueden recibir y enviar información de los drivers u otros bloques.

Funciones de un software HMI

- Monitoreo: Es la habilidad de obtener y mostrar datos de una planta en tiempo real. Estos datos pueden ser mostrados como números, texto o gráficos que permitan una lectura sencilla para ser interpretada.
- Supervisión: Permite junto al monitoreo la posibilidad de ajustar las condiciones de trabajo del proceso directamente desde la computadora.
- Alarmas: Es un evento excepcional dentro de un proceso, que se activa cuando una variable alcanza el límite de control preestablecido.
- Control: Capacidad de aplicar algoritmos que ajustan los valores del proceso y así mantener estos valores dentro de ciertos límites.
- Histórico: Es la capacidad de muestrear y almacenar datos del proceso en archivos a determinada frecuencia. Es una herramienta para la optimización y corrección de procesos.

Un software HMI debe realizar un conjunto de tareas o actividades para garantizar que cumpla correctamente sus funciones, las cuales se presentan a continuación:

Tareas de un software HMI

- Permitir la comunicación con los dispositivos de campo.
- Actualizar una base de datos dinámica con variables del proceso que se supervisa.
- Visualizar las variables mediante pantallas con mímicos.

[6] Variable.

- Permitir al operador enviar señales al proceso mediante botones, controles y ajustes continuos mediante el mouse y el teclado.
- Supervisar niveles de alarmas y alertas y/o actuar en caso que las variables excedan los límites normales.
- Controlar de forma limitada los valores de ciertas variables de los procesos.

Actualmente existen varios tipos de HMI dependiendo de su objetivo, los cuales se presentan a continuación.

Tipos de HMI:

- Interfaces gráficas de usuario: aceptan la entrada a través de dispositivos como el teclado de la computadora y el ratón, y proporcionan una salida gráfica en la pantalla del ordenador. Hay por lo menos dos principios diferentes utilizados en el diseño de interfaz gráfica de usuario: Interfaces de usuario orientada a objetos e Interfaces orientadas a aplicaciones. Es uno de los tipos más empleado actualmente porque permite un aprendizaje intuitivo.
- Interfaces basadas en Web de usuario o interfaces de usuario web: son una subclase de interfaces gráficas de usuario que aceptan una entrada y proporcionan una salida mediante las páginas web que se transmiten a través de una red y es accedida por el usuario mediante un navegador web.
- Interfaces de línea de comandos: El usuario proporciona la entrada al escribir una cadena de comando con el teclado del ordenador y el sistema proporciona una salida de impresión de texto en la pantalla del ordenador. Utilizado por los programadores y administradores de sistemas en los ambientes científicos y de ingeniería, y por los usuarios avanzados de computadoras personales. Es un sistema propenso a errores porque el usuario tiene que memorizar los comandos.
- Las interfaces de voz del usuario: la entrada del usuario se realiza pulsando las teclas o botones, o interactuando verbalmente con la interfaz.
- Multi-pantalla de interfaces: el empleo de múltiples pantallas para proporcionar una interacción más flexible. Esto se emplea a menudo en la interacción de juegos de ordenador. (12)

1.2.1 INTERFACES HOMBRE-MÁQUINA DEL SISTEMA DE ADQUISICIÓN AREX

El sistema cuenta con los siguientes HMI: Edición, Ejecución y Cliente de Base de Datos Histórico. El HMI de Edición y el de Ejecución cargan la configuración de un archivo XML.

HMI para el tiempo de Edición: Es el encargado de las actividades de configuración y edición de los componentes gráficos de los despliegues que intervienen en la realización de un proceso y del Módulo de Recolección. El Editor también permite configurar los usuarios, grupos de usuarios y permisos de acceso. Además, configura el módulo Histórico. Permite al usuario crear y configurar despliegues, así como los dispositivos y variables asociadas. Un despliegue puede representar varios dispositivos que se deseen supervisar. Los dispositivos tienen asociados variables, las cuales son representadas mediante componentes gráficos. El HMI Edición brinda además la posibilidad de configurar alarmas asociadas a las variables. Cuenta con un espacio de trabajo que permite programar rutinas de control sencillas en el lenguaje JavaScript. Además, permite exportar e importar la configuración establecida por el usuario en un archivo XML que posteriormente será utilizado por los Módulos Recolección y Ejecución.

HMI para tiempo de Ejecución: Permite la representación en tiempo de ejecución de los procesos mediante la actualización de los componentes gráficos ubicados en los despliegues. Permite además la generación de sumarios de puntos, lo que posibilita la centralización de la información de las variables por dispositivos. Ofrece opciones para la visualización y el manejo de alarmas generadas en el sistema, permitiendo modificar el estado de las mismas. Otra de las funcionalidades soportadas es el envío de comandos de escritura, que posibilita al operador ejercer determinado nivel de control sobre los procesos. El archivo XML de configuración generado por el Módulo Editor es el punto de entrada para el Arex-Ejecución. Se comunica con el Módulo Recolector a través de TCP Socket para obtener información de las variables, como puede ser: el valor, la calidad y la marca de tiempo.

HMI Cliente de Base de Datos Histórica: Es el encargado de permitir el acceso a los datos de las variables y las alarmas almacenadas por el Módulo Histórico. También permite la visualización de las variables y las alarmas mediante gráficos y formularios.

1.3 HMI WEB

En los últimos años con el desarrollo acelerado de las aplicaciones Web, se han fortalecido las variantes en cuanto a uso y acceso de la información desde diferentes lugares en el mundo. La tendencia actual es requerir aplicaciones rápidas, ligeras y robustas que satisfagan las necesidades de los usuarios. Las redes proporcionan determinados servicios, entre los que se encuentran el de correo electrónico, acceso a la información, servicios Web, transferencia de ficheros, de televisión y telefonía. El desarrollo de los HMI no ha estado ajeno a este proceso evolutivo de las tecnologías, aprovechando los servicios Web como el intercambio de datos mediante estándares como XML, logrando la interoperabilidad de los sistemas.

Como ejemplo de HMI Web desarrollados en el mundo tenemos el HMI Web PASvisu desarrollado por la compañía Pilz. Este software permite el control a distancia de una planta o local a través del acceso remoto. Para el desarrollo de este HMI se emplearon tecnologías como HTML5, CSS3 y JavaScript. PASvisu se caracteriza por el manejo intuitivo y ofrece libertad de configuración de proyectos: con él, los usuarios de Pilz pueden controlar y visualizar completamente las soluciones de control para instalaciones. El software de visualización permite crear y configurar proyectos de visualización mediante el PASvisu Builder. El acceso a todos los datos del proyecto de automatización, incluidas las variables del proceso y los espacios de nombres, elimina la introducción manual, susceptible de errores, y la asignación de variables. (13)

En la UCI en el CEDIN se realizó la aplicación WebUInt, encargada de la supervisión de procesos industriales. Fue diseñado para el SCADA Nacional o SAINUX. Es una aplicación capaz de visualizar y supervisar variables asociadas a distintos tipos de gráficos y controles previamente diseñados que permite la interacción con los componentes representados. Para el desarrollo de este HMI se utilizaron tecnologías como JavaScript 1.8 y CSS 3.

En la UCI también se realizó un HMI web orientado a la supervisión y el control de equipamientos a distancia para el Ministerio de las Fuerzas Armadas Revolucionarias. Para el desarrollo de este HMI fue utilizado como tecnología de desarrollo del lado del servidor PHP 5.2.5 y como tecnología del lado del cliente JavaScript y CSS 3.

1.4 METODOLOGÍA, TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN LA SOLUCIÓN.

1.4.1 METODOLOGÍA DE DESARROLLO.

El éxito de un software depende en gran medida de la metodología elegida por el equipo de desarrollo, pues de ella depende que se maximice el potencial del equipo o no en cuanto a los recursos y el tiempo empleado.

Existen dos grandes enfoques, tanto metodologías tradicionales como metodologías ágiles, las primeras están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y el mantener una buena relación con el cliente. (14)

En la UCI al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP (*Agile Unified Process*), de forma tal que se adecue al ciclo de vida definido para la actividad productiva. Se decide hacer cuatro variaciones (escenarios) de AUP para ajustarla a las características de cada proyecto, teniendo en cuenta el equipo de desarrollo y los recursos. A diferencia de la metodología AUP original la desarrollada en la UCI consta de solo tres fases como se puede observar en la siguiente imagen.

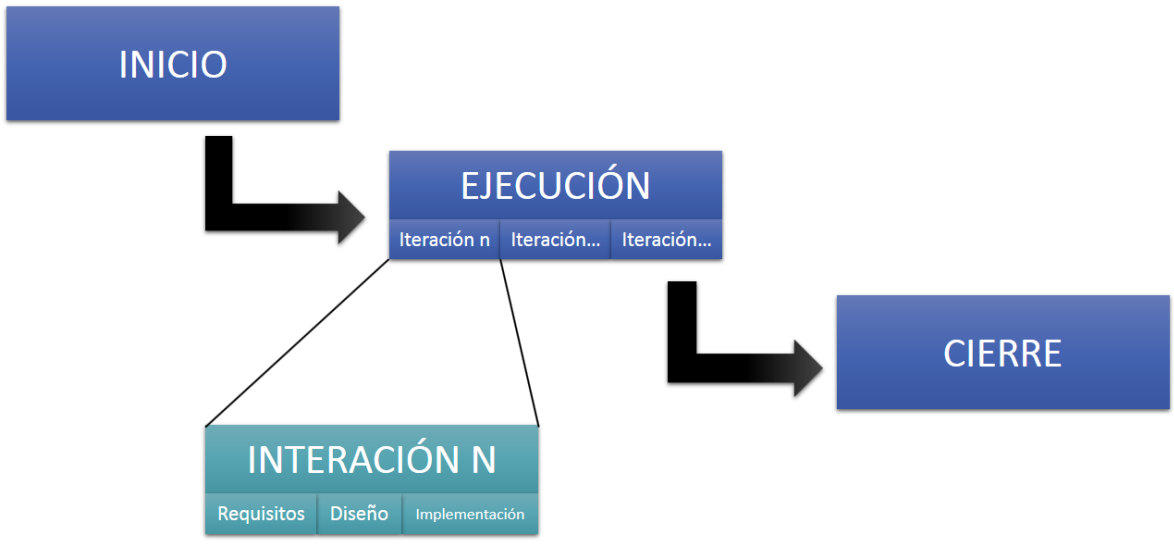


Ilustración 5: Fases e Iteraciones. (15)

Faces de AUP-UCI:

- Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo para decidir si se ejecuta o no el proyecto.
- Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa el producto, al que posteriormente se le hacen pruebas.
- Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Antes se mencionó que la metodología AUP-UCI cuenta con cuatro variaciones o escenarios, de los cuales se muestran sus características a continuación:

- Escenario No. 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema.
- Escenario No. 2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.
- Escenario No. 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.

- Escenario No. 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar 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. Se recomienda en proyectos no muy extensos, ya que una historia de usuario no debe poseer demasiada información.

Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI[7] -DEV v1.3. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. (15).

Después de analizar los escenarios propuestos por la variante AUP para la actividad productiva en la UCI es seleccionado el No. 4, ya que el equipo de desarrollo no es grande, el software no es una aplicación extensa y el cliente acompaña al equipo de desarrollo con frecuencia. Por tanto se concluye que dicho escenario es el más adecuado para las necesidades del producto que se pretende lograr.

1.4.2 TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES WEB.

El uso de aplicaciones Web presenta una serie de ventajas sobre las aplicaciones de escritorio como son: trabajar a distancia con mayor facilidad, para interactuar con la aplicación solo se necesita un dispositivo con un navegador Web, el usuario no necesita conocimientos previos de informática, la disponibilidad del producto solo dependerá de la conexión con el servidor de la aplicación. (17)

En la presente investigación se hizo un estudio de las tecnologías de desarrollo para la Web más influyentes que se ajustaran al producto que se quiere lograr. Para la selección del lenguaje de desarrollo del lado del servidor se hizo una comparación entre PHP (*Hypertext Pre-Processor*) y Python que son los más empleados actualmente por las ventajas que ofrecen en el proceso de desarrollo.

[7] *Capability Maturity Model Integration*

PHP: es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado un servidor web con las bibliotecas de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión (.php). PHP cuenta con ventajas como fácil de aprendizaje, se caracteriza por ser un lenguaje muy rápido, soporta en cierta medida la orientación a objeto. Es un lenguaje multiplataforma ya que puede ser ejecutado en sistemas operativos como GNU/Linux, Windows y está incluido en Mac desde la versión OS X 10.0.0. Cuenta con capacidad de conexión con la mayoría de los manejadores de base de datos, tales como: MySQL, PostgreSQL, Oracle, Microsoft SQL Server, entre otras, no requiere definición de tipos de variables ni manejo detallado de bajo nivel. Tiene desventajas como que se necesita instalar un servidor Web y todo el trabajo lo realiza el servidor, no delega al cliente. Por tanto, puede ser más ineficiente a medida que la cantidad de solicitudes aumente. La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP, la programación orientada a objetos es aún muy deficiente para aplicaciones grandes, dificulta la modularización y la organización por capas de la aplicación.

Python: es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos. (18). El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Cuenta con ventajas como: ser libre y de fuente abierta, sencillo de programar, licencia de código abierto (*Opensource*) y portable.

Python puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos (19). Python y su extensa biblioteca estándar están disponibles libremente, en forma de fuentes o ejecutables para las plataformas más importantes (20).

Python 2.7 proporciona una plataforma de base estable y con el apoyo de los sistemas de producción que aún no han sido portados a Python 3, es la última versión de la serie 2 de Python,

razón por la cual es la más segura y la que más tiempo de soporte tiene, contiene bibliotecas estándar, además es compatible con bibliotecas de Python 3.x.

Después de analizar los lenguajes PHP y Python se llega a la conclusión de a pesar que PHP ofrece muchas ventajas y ser un excelente lenguaje para el desarrollo web, para esta investigación es seleccionado Python como leguaje de desarrollo no solo por sus ventajosas características, sino porque es necesario interpretar datos en lenguaje C++ en el sistema a desarrollar.

Como tecnologías del lado del cliente serán empleadas: HTML (*HyperText Markup Language*) como lenguaje de marcas, JavaScript como lenguaje interpretado y CSS (*Cascading Style Sheets*).

HTML: es un lenguaje estático desarrollado por la *World Wide Web*. HTML cuenta con algunas ventajas como: ser sencillo y permitir describir hipertextos, no necesita de grandes conocimientos con la presencia de un editor de páginas Web, genera archivos pequeños, rápido despliegue y es un lenguaje de rápido aprendizaje. Cuenta con desventajas como: que es un lenguaje estático, la interpretación de cada navegador puede ser diferente, guarda muchas etiquetas que dificultan la corrección, el diseño es más lento, las etiquetas son limitadas.

JavaScript: es un lenguaje interpretado. Fue creado por Brendan Eich en la empresa *Netscape Communications*. Utilizado principalmente en páginas Web. Es similar a Java, aunque no es un lenguaje orientado a objetos. La mayoría de los navegadores en sus últimas versiones interpretan código JavaScript. Puede ser integrado dentro de las páginas Web. Para evitar incompatibilidades el *World Wide Web Consortium* (W3C) diseño un estándar denominado DOM (en inglés *Document Object Model*, en su traducción al español Modelo de Objetos del Documento). Cuenta con ventajas como ser un lenguaje de scripting seguro y fiable, el código de JavaScript es ejecutado del lado del cliente, los scripts tienen capacidades limitadas por razones de seguridad. Tiene desventajas como: que el código es visible por cualquier usuario y debe descargarse completamente, lo cual constituye un riesgo de seguridad, ya que terceras personas podrían inyectar el código JavaScript evitando así las medidas de seguridad y robar información clasificada, este tipo de ataque es conocido como XSS (*Cross Site Scripting*).

CSS: es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados *"documentos semánticos"*). Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para *marcar* los contenidos, o sea, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, entre otras opciones de formato. (21)

Framework de desarrollo

En el desarrollo de Software, un framework o marco de trabajo es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (22)

Python cuenta con cuatro principales frameworks de desarrollo, el Pyramid, Bottle, Django y Flask (24).

Framework Pyramid: ofrece flexibilidad en cuanto al trabajo de desarrollo, da la ventaja de prototipar un concepto, es empleado por desarrolladores que trabajan con proyectos con API (*Application Programming Interface*) y es empleado en el desarrollo de aplicaciones Web de gran envergadura.

Framework Bottle: proporcionando un mínimo de herramientas al desarrollador como enrutamiento, plantillas y una pequeña abstracción sobre Servidor Web *Gateway Interface*, es empleado por personas que quieren desarrollar una aplicación simple, flexible, es usado generalmente para crear una API Web.

Framework Flask: es un micro framework que se creó originalmente como una broma del *April Fools Day* (como el día de los inocentes en EEUU) que derivó en un framework en solo un único archivo. Es simple y pequeño; todo el framework consiste en un conjunto de módulos. No hay un esqueleto o una estructura de la cual partir, todo se empieza con una página en blanco. Flask no proporciona grandes funcionalidades, pero hay extensiones Flask disponibles para agregar ORM (*Object Relational Mapping*), validación de formularios, manejo de carga, etc. Es ideal para aprender a programar, para desarrolladores que quieran crear prototipos de forma rápida y que necesiten una aplicación independiente.

Framework Django: uno de los mayores beneficios de Django es su amplia comunidad, para ayuda con cualquier aspecto, ya sea desde instalación y diseño de aplicaciones, hasta diseño de bases de datos e implementaciones. (25) Django impulsa el desarrollo de código limpio al promover buenas prácticas de desarrollo Web, sigue el principio DRY (conocido también como Una vez y sólo una). Usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada MTV (*Model-Template-View*), que sería Modelo-Plantilla-Vista, esta forma de trabajar permite que sea pragmático. (26)

En el caso de la presente investigación se seleccionó Django como framework de desarrollo para el lenguaje de programación Python por las ventajas que ofrece.

Para el trabajo con CSS se hizo un estudio de los principales frameworks empleados en el mundo para dicha labor. Entre los más populares están Bootstrap, Foundation y Semantic-UI.

Foundation: ofrece una solución de extremo a extremo para la construcción de sitios Web flexibles, de cualquier tamaño sin comprometer la velocidad.

Semantic-UI: es capaz de integrarse con bibliotecas de terceros como AngularJS, Ember y Meteor y casi no necesita utilizar bibliotecas adicionales incluso en los grandes y complejos proyectos Web.

Bootstrap: Es un paquete de herramientas que simplifica el proceso de creación de diseños Web combinando CSS y JavaScript (27). Ofrece una serie de plantillas CSS y ficheros JavaScript que permiten integrar el framework de forma sencilla y potente en proyectos Web. Permite crear

interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tablets y móviles a distintas escalas y resoluciones. Se integra perfectamente con las principales bibliotecas JavaScript. Es ligero y funciona en todos los navegadores. Ofrece un diseño sólido, así como distintos componentes que pueden utilizarse con unos estilos predefinidos y fáciles de configurar.

Para la presente investigación fue seleccionado como framework de CSS Bootstrap.

Para el trabajo con JavaScript existen potentes frameworks como Backbones JS, Ember JS, Angular JS, entre otros, pero para la presente investigación no es necesario el uso de un framework para el trabajo con JavaScript, ya que el mayor peso de desarrollo se encuentra en el lado del servidor. Por tanto para la investigación fue seleccionado jQuery.

JQuery: Es una biblioteca de JavaScript creada por John Resig que permite simplificar la manera de interactuar con los documentos HTML y manipular el árbol DOM. Con el empleo de DOM se logra transformar internamente el archivo XML original en una estructura más fácil de manejar formada por una jerarquía de nodos. También permite manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX (*Asynchronous Javascript and XML*). jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código.

Para el trabajo con los componentes gráficos animados o mímicos se emplea la herramienta Inkscape la cual se presenta a continuación:

Inkscape es un editor gráfico gratuito bajo licencia GNU GPL para la creación de gráficos vectoriales. Es compatible con los estándares XML, SVG (*Scalable Vector Graphics*) y CSS. Para la creación de imágenes vectoriales emplea el formato SVG, soportando formas, trazos, texto, marcadores, clones, mezclas de canales alfa, transformaciones, gradientes, patrones y agrupamientos.

1.4.3 HERRAMIENTAS DE DESARROLLO.

Para el desarrollo con el lenguaje Python fueron investigados los entornos de desarrollo integrados (IDE por sus siglas en inglés) PyDev y PyCharm.

PyDev: es una extensión de Python para Eclipse que permite utilizar este IDE multiplataforma para programar en Python. Cuenta con autocompletado de código (con información sobre cada elemento), resaltado de sintaxis, un depurador gráfico, resaltado de errores, explorador de clases, formateo del código, refactorización. A pesar de todas las ventajas que brinda este IDE cuenta con la deficiencia de necesitar una cantidad importante de memoria, razón por la cual es considerada una herramienta pesada y no es completamente estable (29). Permite la integración con el framework Django.

PyCharm: es un IDE desarrollado por la compañía Jetbrains, está basado en IntelliJ IDEA, el IDE de la misma compañía, pero enfocado hacia Java y es la base de Android Studio. Presenta un editor de códigos inteligente que entiende los detalles específicos de Python y ofrece mejoradores de la productividad como: formateo automático de código, refactorizaciones, importación automática (30). PyCharm cuenta con ventajas como: autocompletado, resaltador de sintaxis, herramienta de análisis. Integración con frameworks Web como: Django, Flask, Pyramid, Web2Py. Debugger avanzado de Python y Javascript. Integración con lenguajes de plantillas: Mako, Jinja2, Django Template. Soporta entornos virtuales e intérpretes de Python 2.x, 3.x, PyPy, Iron Python y Jython. Compatibilidad con SQLAlchemy (ORM), Google App Engine, Cython. Soporte para modo VIM (Con extención). Sistemas de control de versiones Git, CVS, Mercurial. Soporte para XML, HTML, RelaxNG, PyQt y PyGtk. Permite la realización de pruebas al código implementado.

Después de analizar detalladamente las ventajas asociadas a PyDev y PyCharm se decide escoger para la presente investigación como entorno de desarrollo integrado a PyCharm pues a pesar que ambos cuentan con prácticamente las mismas características, PyDev puede ser una herramienta sumamente pesada.

CONCLUSIONES PARCIALES

Después de realizado el análisis del marco teórico de la investigación se puede concluir que:

- Los métodos y técnicas de búsqueda de información fueron de suma importancia en la obtención de la documentación para establecer el estado del arte de los HMI en general y

particularmente los HMI web, así como la tecnología web más adecuada para el desarrollo del producto esperado.

- La integración del Módulo Ejecución en la web para el Sistema de Adquisición Arex es una gran ventaja, ya que es una aplicación fácil de usar por el usuario final. No requiere un proceso de instalación. Tiene una alta disponibilidad, ya que mientras el usuario tenga conexión a Internet o red local podrá hacer consultas a la aplicación.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN PARA EL MÓDULO EJECUCIÓN EN LA WEB DEL SISTEMA DE ADQUISICIÓN AREX.

INTRODUCCIÓN

En este capítulo se precisan elementos para llegar a la propuesta de solución de la investigación. Se lleva a cabo un análisis de las características, componentes y particularidades de cada una de las funcionalidades del HMI Web que se desea implementar.

Se realiza la descripción de la arquitectura propuesta para la solución, así como los patrones presentes en la misma. Es aplicada la metodología de desarrollo seleccionada, además de definir la documentación que es requerida en las primeras fases de la metodología.

2 FASES DE AUP-UCI

2.1 FASE DE INICIO

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización. El cliente propicia la información necesaria acerca del proyecto, se realizan estimaciones de tiempo, esfuerzo y costo para decidir si se ejecuta o no el proyecto.

2.1.1 RECOPILACIÓN DE INFORMACIÓN PARA EL SISTEMA A DESARROLLAR

Para el levantamiento informacional del sistema se realizaron entrevistas al grupo de desarrollo del Sistema de Adquisición Arex. Fueron consultados programadores, directivos y analistas para comprender el funcionamiento del sistema y obtener información.

En la conversación realizada se recopiló la información necesaria para identificar las funcionalidades a desarrollar. También se definieron los sistemas con los cuales es necesario establecer conexión directa.

En el encuentro se identificaron funcionalidades necesarias que el sistema debe realizar como: la lectura del archivo XML el cual almacena los datos que se mostrarán en el sistema y es generado

en el Módulo Editor. Mostrar los HMI[8], así como los despliegues asociados a cada uno de los HMI. Modificar los valores o estados de los gráficos que permiten ser modificados. Los usuarios del sistema tienen que estar previamente creados en el Editor, donde le son asignados los HMI a los que tiene acceso y los permisos que tienen sobre estos HMI. La información de los usuarios tales como: nombre, contraseña, HMI y permisos están almacenados en el archivo XML.

Características de la solución propuesta

Después de realizada la entrevista y a partir de los elementos obtenidos en ella se define como nombre de la aplicación Arex Web. Para el desarrollo del sistema se utilizaron como lenguajes de programación: Python 2.7 y JavaScript, como lenguaje de marcas HTML 5 y como lenguaje de estilo CSS 3. Como frameworks de desarrollo utilizados Bootstrap 3.1.1, JQuery 1.10.2, como IDE PyCharm 4.5.4 y como servidor web Apache. Como guía para el desarrollo de la aplicación fue utilizada la metodología AUP-UCI a través de sus fases: inicio, ejecución y cierre.

2.2 FASE DE EJECUCIÓN

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, y se implementa el producto.

2.2.1 REQUISITOS NO FUNCIONALES DEL SISTEMA

Los requisitos no funcionales son características del sistema que garantizan la flexibilidad y mantenibilidad que siempre se tienen en cuenta en la fase de diseño.

Usabilidad:

- El software podrá ser usado por personas con conocimientos básicos en el trabajo con computadoras.

[8] Se refiere a un HMI virtual para representarlo dentro del verdadero.

Confiabilidad:

- La información debe estar protegida del acceso no autorizado.
- Para mantener activos los servicios deben montarse sistemas de respaldo eléctrico.

Apariencia:

- La interfaz debe ser amigable y de fácil comprensión para los usuarios.
- Mensajes sin ambigüedades.
- Los colores del diseño de la interfaz deben ser claros.

Licencias y patentes:

- Para el desarrollo de la aplicación deben utilizarse herramientas libres.

Portabilidad:

- El sistema debe funcionar sobre los navegadores web más empleados (*Chrome, Firefox, Microsoft Edge y Opera*).

Legales:

- Se debe garantizar la protección de los datos y legalidad de los mismos.

Seguridad:

- El sistema debe requerir la autenticación de los usuarios.

Hardware:

- Requisitos mínimos del sistema:
    - 256 MB de RAM.
    - Microprocesador a 1.0 GHz.

2.2.2 DESCRIPCIÓN DE LAS HISTORIAS DE USUARIO (HU)

Se utilizan para especificar los requisitos de las aplicaciones de software en las metodologías ágiles (SCRUM, XP, FDD, ASD, AUP, LD, etc.). Las historias de usuario (HU) son tarjetas dónde el interesado describe brevemente (con el fin de ser dinámicas y flexibles) las características que el sistema debe poseer (31). Según el artículo “Características de una buena historia de usuario” de Jorge Hernán Abad Londoño, miembro de la Comunidad de Scrum (Scrum Alliance), define los siguientes criterios como los más importantes a cumplir:

- Independiente: Que no requiera de otra historia.
- Negociable: Se puede reemplazar por otra de diferente prioridad.
- Valor: Necesaria y de valor para el proyecto.
- Pequeña: No muy grandes, o sea, con funcionalidades pequeñas.
- Verificables: Que se les puedan realizar pruebas.

En el proceso de confección de las HU se realizó una reunión con el cliente que también es parte del equipo de desarrollo de Arex, donde quedó definido el contenido de cada una, así como las relaciones existentes entre ellas. De acuerdo a la necesidad de desarrollo de las funcionalidades quedó precisada también la prioridad para cada una de las HU.

El equipo de desarrollo realizó un análisis de las prioridades y la dependencia de las HU para definir el costo en semanas para la implementación de cada una. Es importante destacar que si una HU demora más del tiempo previsto en la planificación para su desarrollo es aconsejable dividirla en historias más pequeñas. Estas nuevas historias pueden describirse en cualquier momento.

Las HU son representadas por tablas que contienen los siguientes campos o secciones:

- Código: Las siglas HU precedido de un número representa el identificador de la historia.
- Nombre: Nombre que identifica la HU.
- Referencia: Es el conjunto de identificadores de las HU de las cuales depende la historia actual que está en proceso de descripción.
- Prioridad: El valor de este campo lo define el cliente dependiendo de la importancia. Puede clasificarse en Baja, Media o Alta, lo que determina el orden para el proceso de desarrollo.

- Iteración Asignada: Número de la iteración en la cual se desarrollará la HU.
- Puntos Estimados: Tiempo estimado en semanas que se le asignará.
- Descripción: Breve descripción del proceso que define la historia.
- Observaciones: Algunas aclaraciones que son importante señalar acerca de la historia.

Se realizaron un total de 16 historias de usuario, tomando la semana como unidad de tiempo para los puntos de estimación. De las HU confeccionadas se tienen doce con complejidad alta, y cuatro con complejidad media. A continuación, se presentan tres ejemplos de HU en las tablas. La HU1 y la HU6 constituyen los procesos de mayor relevancia para el sistema y la HU12 al igual que las anteriores son procesos sumamente complejos de implementar.

Tabla 1: HU1 Cargar XML.

Historia de Usuario	
Código: HU1.	**Nombre:** Cargar configuración de un proyecto desde archivo XML.
Referencia: HU6.	**Prioridad:** Alta.
Iteración Asignada: 1	**Puntos Estimados:** 1.0
Descripción: La aplicación debe permitir cargar la información contenida en el archivo XML. Las etiquetas principales del archivo XML son: *collector* (Recolector), *view* y *security*.	
Observaciones: 1. El Recolector especifica el *host* y el puerto que se está empleando para la conexión y la información referente a cada uno de los dispositivos. 2. Los dispositivos agrupan las variables a leer por el recolector, y las variables contienen las alarmas que le fueron configuradas.	

3. En *view* se encuentran cada HMI con la información correspondiente a cada uno de los despliegues. 4. En *security* se encuentran los grupos de usuarios los cuales tienen un nombre y un conjunto de usuarios asignados de los cuales se tienen el nombre y la contraseña encriptada.
Prototipo de interfaz de usuario: No muestra recurso visual.

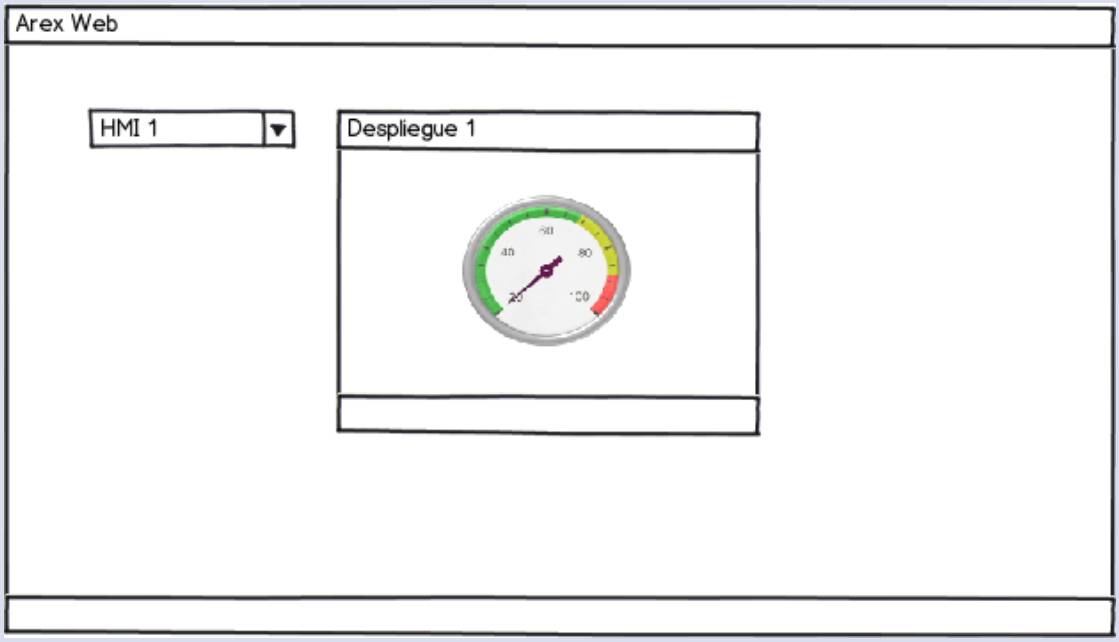
Tabla 2: HU6 Comunicar el sistema con el recolector de Arex.

Historia de Usuario	
Código: HU14.	**Nombre:** Comunicar el sistema con el recolector de Arex.
Referencia: No depende de otra HU.	**Prioridad:** Alta.
Iteración Asignada: 3	**Puntos Estimados:** 1.5
Descripción: La aplicación debe comunicarse con el módulo Recolector.	
Observaciones: 1. El recolector será el encargado de proporcionar a la aplicación HMI Web el archivo XML, el cual contiene la información necesaria para visualizar los HMI, sus despliegues y cada uno de los componentes gráficos. 2. La aplicación intercambiará con el recolector mensajes de texto plano.	
Prototipo de interfaz de usuario: No muestra recurso visual.	

Tabla 3: HU12 Configurar el componente gráfico Clock[9].

Historia de Usuario

[9] Hace referencia a galvanómetro.

Código: HU12.	**Nombre:** Configurar el componente gráfico *Clock*.
Referencia: HU1.	**Prioridad:** Alta.
Iteración Asignada: 3	**Puntos Estimados:** 1.2
Descripción: El sistema debe permitir mostrar el recurso gráfico *Clock*.	
Observaciones: 1. El *clock* obtiene las propiedades posición, rotación, escala y valor del archivo XML. **2.** El usuario no tiene acceso a modificar los valores de este recurso gráfico.	
Prototipo de interfaz de usuario:	
	

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas

de programación (32). A continuación, se presenta la tabla 4 donde se resume la estimación del esfuerzo realizado por parte de los desarrolladores.

Tabla 4: Estimación de esfuerzo por HU.

Identificador	Historias de usuario	Puntos de estimación (semanas)
HU1	Cargar configuración de un proyecto desde archivo XML.	1.0
HU2	Autenticar usuario.	2.0
HU3	Mostrar las vistas de los HMI.	0.6
HU4	Mostrar las vistas de los despliegues por cada HMI.	0.6
HU5	Visualizar sumario de alarmas.	0.5
HU6	Controlar las alarmas.	1.5
HU7	Configurar el componente gráfico *Light Button*[10].	1.0
HU8	Configurar el componente gráfico *Switch*[11]2.	1.0
HU9	Configurar el componente gráfico *Switch*3.	1.0
HU10	Configurar el componente gráfico *Humidity*[12].	1.0

[10] Luces.

[11] Interruptor.

[12] Humedad.

HU11	Configurar el componente gráfico *Light State*[13].	1.0
HU12	Configurar el componente gráfico *Clock*.	1.2
HU13	Configurar el componente gráfico *Ruler*[14].	1.2
HU14	Comunicar el sistema con el recolector de Arex.	2.5
HU15	Visualizar sumario de variables.	1.0
HU16	Configurar el componente gráfico *Circular Control*[15].	1.0

Después de realizar un análisis se estima como costo para el desarrollo 18.1 semanas. Los valores de las historias de usuarios están comprendidos entre 0.5 y 2.5 dependiendo de la complejidad correspondiente a cada una.

2.2.3 PLAN DE ITERACIONES

Después de identificar y elaborar cada una de las HU y determinar la estimación del esfuerzo continúa la confección del plan de iteraciones. Para cada iteración son seleccionadas las HU de acuerdo al orden preestablecido.

Se definieron tres iteraciones para el desarrollo de las partes funcionales de la aplicación. A continuación, se explican y justifican cada una de las iteraciones:

- Iteración 1: Para esta iteración se desarrollan las HU de la 1 a la 6 correspondientes a cargar la configuración de un proyecto desde un archivo XML, autenticar usuario, mostrar las vistas

[13] Estado de las luces.

[14] Regla de medición.

[15] Círculo de control.

de los HMI, mostrar las vistas de los despliegues por cada HMI y visualizar sumario de alarmas. Con esta iteración queda garantizada la funcionalidad de cargar la configuración que posteriormente se utilizará en el desarrollo de las próximas iteraciones. Además, queda asegurada la seguridad de la aplicación a partir de la autenticación del usuario, donde un usuario solo tendrá acceso a determinados HMI dependiendo del grupo asignado en el Módulo Editor. Con esta iteración también se logra definir una estructura visual para la aplicación.

- Iteración 2: En esta iteración se desarrollarán las HU de la 7 a la 11 correspondientes a controlar las alarmas, configurar el componente gráfico *Light Button*, el componente gráfico *Switch2*, el componente gráfico *Switch3* y el componente gráfico *Humidity*. Con la implementación de las actividades correspondientes a esta iteración se lograría una entrega funcional de la aplicación permitiendo el control de las alarmas y de los componentes gráficos. Para la implementación de las HU de esta iteración son tomadas como apoyo las historias de la iteración anterior.
- Iteración 3: En esta iteración se desarrollarán las HU de la 12 a la 16 correspondiente a configurar el componente gráfico Light State, configurar el componente gráfico *Clock*, configurar el componente gráfico *Ruler* y comunicar el sistema con el recolector de Arex. Siendo estas las actividades más complejas en cuanto a implementación. Con esta iteración quedará establecida la comunicación de la aplicación con el recolector, lo cual garantizará la obtención y actualización de la información necesaria que deberá visualizar. Al finalizar esta iteración se realizará la entrega final del producto.

Para aproximar el tiempo de ejecución de las iteraciones se tomó como medida de tiempo que cada semana consta de 5 días (de lunes a viernes), de los cuales se trabaja aproximadamente 7 horas diarias. A continuación, se muestra el plan de iteraciones en la tabla 5 con el tiempo estimado en semanas.

Tabla 5: Plan de iteraciones.

Iteración	Historia de usuario	Puntos de estimación (semanas)
1	Cargar XML.	6.2
	Autenticar usuario.	
	Mostrar las vistas de los HMI.	
	Mostrar las vistas de los despliegues por cada HMI.	
	Visualizar sumario de alarmas.	
	Controlar las alarmas.	
2	Configurar el componente gráfico *Light Button*.	5.0
	Configurar el componente gráfico *Switch2*.	
	Configurar el componente gráfico *Switch3*.	
	Configurar el componente gráfico *Humidity.*	
	Configurar el componente gráfico *Light State*.	
3	Configurar el componente gráfico *Clock.*	6.9
	Configurar el componente gráfico *Ruler.*	
	Comunicar el sistema con el recolector de Arex.	
	Visualizar sumario de variables.	
	Configurar el componente gráfico *Circular Control*.	

2.2.4 PLAN DE ENTREGAS

A partir del plan de iteraciones que se definió anteriormente y en correspondencia con el mismo se realiza el plan de entregas, que se muestra en la tabla 6. En este plan de entregas se proponen dos versiones funcionales del producto y una última entrega antes de la fase de cierre del proyecto.

Tabla 6: Plan de entrega.

Historia de usuario	**1ra Iteración** **17 de febrero del 2016**	**2da Iteración** **24 de marzo del 2016**	**3ra Iteración** **9 de mayo del 2016**
Cargar XML.	V1.0		
Autenticar usuario.			
Mostrar las vistas de los HMI.			
Mostrar las vistas de los despliegues por cada HMI.			
Visualizar sumario de alarmas.			
Controlar las alarmas.			
Configurar el componente gráfico *Light Button.*		V1.1	
Configurar el componente gráfico *Switch2.*			
Configurar el componente gráfico *Switch3.*			
Configurar el componente gráfico *Humidity*.			

Configurar el componente gráfico *Light State*.			
Configurar el componente gráfico *Clock.*			V1.2
Configurar el componente gráfico *Ruler.*			
Comunicar el sistema con el recolector de Arex.			
Visualizar sumario de variables.			
Configurar el componente gráfico *Circular Control*.			

2.2.5 TAREAS DE INGENIERÍA O PROGRAMACIÓN

La planificación de las tareas de ingeniería o programación se encuentran relacionadas con cada una de las iteraciones, pues las historias de usuario se transforman en tareas que son desarrolladas por los programadores del equipo de desarrollo. En correspondencia de las historias de usuario y sus tareas derivadas se distribuyeron entre las iteraciones planificadas.

Tabla 7: Tareas de ingeniería de la iteración 1.

Historia de usuario	Tareas por historia de usuario
Cargar XML.	1. Desarrollar funcionalidad que permita convertir la información del archivo XML a JSON. 2. Desarrollar funcionalidad que permita extraer la información del JSON. 3. Desarrollar pruebas internas. 4. Desarrollar pruebas de aceptación.

Autenticar usuario.	5. Desarrollar la interfaz para autenticar usuario. 6. Desarrollar funcionalidad para validar los datos entrados por un usuario. 7. Desarrollar pruebas internas. 8. Desarrollar pruebas de aceptación.
Mostrar las vistas de los HMI.	9. Desarrollar la interfaz de las vistas de los HMI. 10. Desarrollar pruebas internas. 11. Desarrollar pruebas de aceptación.
Mostrar las vistas de los despliegues por cada HMI.	12. Desarrollar la interfaz de la vista de los despliegues de cada HMI. 13. Desarrollar pruebas internas. 14. Desarrollar pruebas de aceptación.
Visualizar sumario de alarmas.	15. Desarrollar la interfaz para el sumario de alarmas. 16. Desarrollar las funcionalidades para actualizar los valores de las alarmas. 17. Desarrollar pruebas internas. 18. Desarrollar pruebas aceptación.
Controlar las alarmas.	19. Desarrollar interfaz para controlar alarmas. 20. Desarrollar funcionalidad silenciar alarma. 21. Desarrollar funcionalidad reconocer alarma. 22. Desarrollar pruebas internas. 23. Desarrollar pruebas de aceptación.

La tabla 8 mostrada anteriormente fue escogida, porque a pesar de ser la iteración 1 la más corta de ella dependen en gran medida el desarrollo de las próximas iteraciones, ya que en ella se realizan actividades como cargar la configuración de un proyecto desde el archivo XML. Cada tarea de desarrollo fue realizada en un período de uno a tres días. Para la confección de las tareas se utilizaron tablas que contienen los siguientes campos:

- No. de tarea: Numeración continua que identifica a la tarea.
- No. de HU: Número de la HU a la cual pertenece.
- Nombre de la tarea: Identificación literal de la tarea.
- Tipo de tarea: Tipo de tarea, dígase diseño, desarrollo, prueba.
- Puntos estimados: Representación en porciento de la cantidad de tiempo estimada de una semana que se utilizará para su realización.
- Fecha inicio: Fecha estimada de inicio de realización.
- Fecha fin: Fecha estimada de fin de realización.
- Descripción: Se describe en que consiste la tarea y que elementos deben cumplirse para declarar la tarea terminada.

En las tablas desde la 8 hasta la 11 se muestran las tareas asociadas a la HU2 pertenecientes al proceso autenticar usuario.

Tabla 8: Tarea de ingeniería 5 de la iteración 1.

Tarea de Ingeniería	
No. de tarea: 5	**No. de HU:** 2
Nombre de la tarea: Desarrollar la interfaz para autenticar usuario.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio: 21/02/2016	**Fecha fin**: 21/02/2016

Descripción: Se realizará el diseño e implementación de la vista para autenticar un usuario. Se muestran los campos usuario y contraseña.

Tabla 9: Tarea de ingeniería 6 de la iteración 1.

Tarea de Ingeniería	
No. de tarea: 6	**No. de HU:** 2
Nombre de la tarea: Desarrollar funcionalidad para validar los datos entrados por un usuario.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio: 22/02/2016	**Fecha fin**: 24/02/2016
Descripción: Se implementarán las funcionalidades para validar que los datos entrados por un usuario son correctos.	

Tabla 10: Tarea de ingeniería 7 de la iteración 1.

Tarea de Ingeniería	
No. de tarea: 7	**No. de HU:** 2
Nombre de la tarea: Desarrollar pruebas internas.	
Tipo de tarea: Pruebas.	**Puntos estimados:** 0.2
Fecha inicio: 25/02/2016	**Fecha fin**: 25/02/2016
Descripción: Se realizan las pruebas de internas al código validando que las funciones se ejecutarán al menos una vez.	

Tabla 11: Tarea de ingeniería 8 de la iteración 1.

Tarea de Ingeniería	
No. de tarea: 8	**No. de HU:** 2
Nombre de la tarea: Desarrollar pruebas de aceptación.	
Tipo de tarea: Pruebas.	**Puntos estimados:** 0.2
Fecha inicio: 26/02/2016	**Fecha fin**: 26/02/2016
Descripción: Se realizan las pruebas de aceptación a partir de los casos de pruebas propuestos por el usuario.	

2.3 FASE DE CIERRE

En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Después de ejecutar las fases de ejecución se realizaron tareas de implementación mediante la aplicación de HU para definir los requisitos funcionales del sistema, los cuales permitieron definir las funcionalidades fundamentales para que la aplicación cumpliera con las necesidades del cliente. Para dar cumplimiento a las tareas de implementación fue diseñado un plan de iteraciones y un plan de entrega, los cuales arrojaron resultados satisfactorios. Se obtuvo un producto final que cumplió con el alcance propuesto por el cliente.

CONCLUSIONES PARCIALES

Con la ejecución de las dos primeras fases de la metodología AUP-UCI se alcanzaron los siguientes resultados, que dan paso a la culminación de la metodología y a la validación de la propuesta.

1. La aplicación de los métodos científicos entrevista y observación tributó a que se realizara un proceso correcto de levantamiento de requisitos, permitiendo así que se identificaran las principales funcionales y características para el Módulo Ejecución en la web.
2. Así como que las HU y las tareas de ingeniería fueron el factor organizativo del proceso de desarrollo del mismo.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS DE LA SOLUCIÓN PROPUESTA.

INTRODUCCIÓN

En el presente capítulo el software desarrollado será sometido a pruebas internas y de aceptación. El proceso de pruebas es realizado en la fase de ejecución de la metodología seleccionada para la investigación. Se hará una descripción tanto de la selección, como de la realización de las pruebas.

3.1 IMPLEMENTACIÓN

En la fase de ejecución se encuentran las iteraciones, dentro de las mismas es donde se realiza el desarrollo del código o implementación. En esta etapa se implementan los artefactos generados en la etapa de análisis y diseño, creando scripts, ejecutables, ficheros de código y similares acordes al lenguaje de programación que se utilice.

3.1.1 ESTRUCTURA DEL CÓDIGO

La aplicación está estructurada de acuerdo a la propuesta de Django de la siguiente manera:

VisorWeb/

- app/
    - /templates/
        - index.html
    - models.py
    - apps.py
    - forms.py
    - views.py
- VisorWeb/
    - settings.py
    - urls.py
    - wsgi.py
- static/

- css
- img
- js
- Media

El archivo *'forms.py'* lo utiliza Django para el trabajo con los formularios de las vistas. El archivo *'settings.py'* describe toda la configuración de la aplicación, conexiones a la base de datos y *middlewares* a usar. El archivo *'models.py'* contiene una descripción de la tabla de la base de datos, como una clase de Python, el cual es llamado modelo. Usando esta clase se pueden crear, buscar, actualizar y borrar entradas de la base de datos usando solo código Python en lugar de escribir declaraciones SQL[16] repetitivas.

El archivo *'views.py*' contiene la lógica de la página en la función *index*, a esta función se le denomina vista. El archivo *'urls.py'* específica qué vista es llamada. En el caso específico de la aplicación, la dirección *…/index/* será manejada por la función *index*. Si el nombre del dominio es *example.com*, cualquier visita a la dirección *http://example.com/index/* llamará a la función *index*.

El archivo *'index.html*' es una plantilla HTML especial que describe el diseño de la página. Usa el lenguaje de plantillas de Django, con declaraciones básicas y lógicas, por ejemplo: *{% for libro in lista_libros %}*.

3.1.1.1 PATRONES DE DISEÑO:

Los patrones de diseño de software permiten reutilizar ideas, ayudando a beneficiarse de la experiencia. Los patrones dan nombre y forma a heurísticas abstractas, reglas y buenas prácticas de técnicas orientadas a objetos.

Una implementación hábil se funda en los principios cardinales que rigen un buen diseño orientado a objetos. En los patrones GRASP[17] se codifican algunos de ellos, que se aplican al preparar los

[16] *Structured Query Language*

[17] Patrones para asignar responsabilidades.

diagramas de interacción, cuando se asignan las responsabilidades o durante ambas actividades. (33)

Patrones GRASP:

- Experto: Asignar una responsabilidad al experto en información: la clase cuenta con la información necesaria para cumplir la responsabilidad.
- Creador: El propósito fundamental de este patrón es la asignación de responsabilidades asociadas a la creación de objetos.
- Bajo acoplamiento[18]: Asignar responsabilidad de modo que no incremente el acoplamiento (dando lugar a resultados negativos como alto acoplamiento). Soporta el diseño de clases más independientes reduciendo el impacto que pueda ocasionar la realización de cambios.
- Alta cohesión[19]: Asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.
- Controlador: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente el controlador de fachada o sistema global, el controlador de tareas y manejadores artificiales de todos los eventos del sistema (controlador de caso de uso de sistema).

En la siguiente imagen se representa el diagrama de clases de la aplicación a desarrollar donde se evidencia el uso de los patrones de diseño Bajo Acoplamiento al utilizar la menor relación posible entre las clases, Experto al asignarle a cada clase las responsabilidades necesarias y Alta Cohesión que se evidencia con la colaboración de las clases para llevar a cabo las tareas.

[18] Es una medida de la fuerza con que una clase está conectada a otras.

[19] Es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

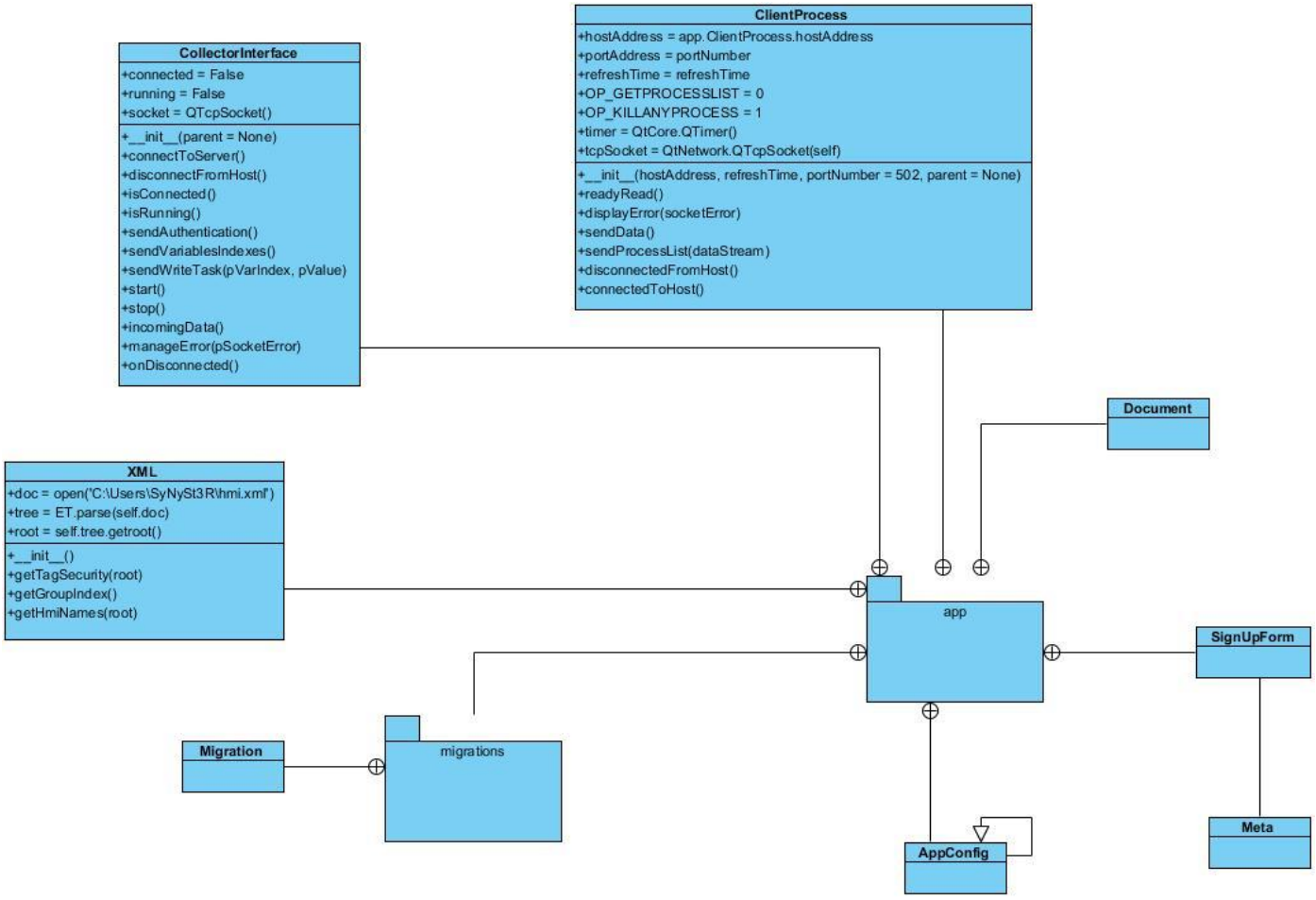


Ilustración 6: Diagrama de clases.

3.1.1.2 PATRÓN DE ARQUITECTURA

Los patrones de arquitectura son aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software. El Modelo-Vista-Controlador(MVC) es un patrón de arquitectura de las aplicaciones software, que separa la lógica de negocio de la interfaz de usuario. (35) En el caso específico del software a desarrollar. El framework Django propone el uso del patrón arquitectónico *Model-View-Template* (MVT) que es una variación del MVC donde *view* hace la función del controlador.

En la ilustración 4 se muestran las relaciones entre las tres capas fundamentales que propone el patrón arquitectónico MVC aplicado a la arquitectura de la solución propuesta.

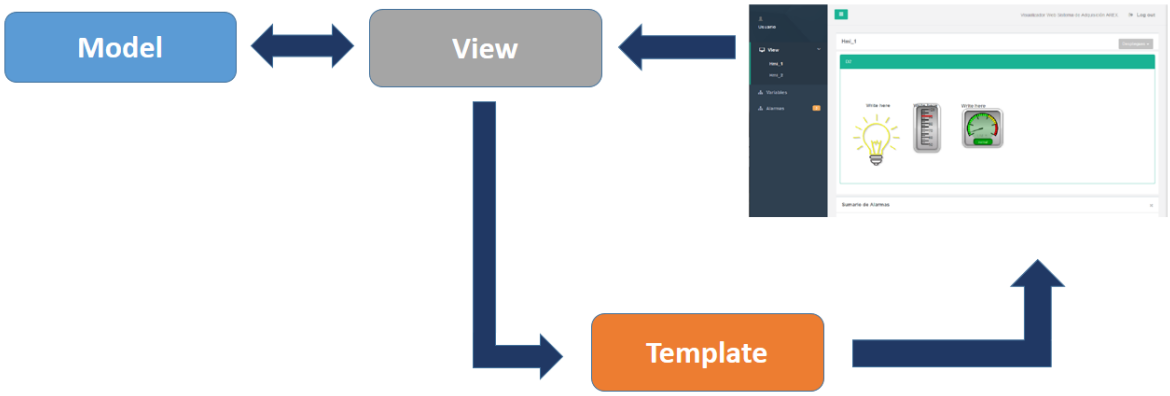


Ilustración 7: Vista de la arquitectura del software a desarrollar.

Tomadas en su conjunto, estas piezas se aproximan al patrón de diseño Modelo-Vista-Controlador (MVC). El MVC define una forma de desarrollar software en la que el código para definir y acceder a los datos (el modelo) está separado del pedido lógico de asignación de ruta (el controlador), que a su vez está separado de la interfaz del usuario (la vista). Una ventaja clave de este enfoque es que los componentes tienen un acoplamiento débil entre sí. Eso significa que cada pieza de la aplicación Web que funciona sobre Django tiene un único propósito clave, que puede ser modificado independientemente sin afectar las otras piezas. Por ejemplo, un desarrollador puede cambiar la URL de cierta parte de la aplicación sin afectar la implementación subyacente. El diseñador puede cambiar el HTML de una página sin tener que tocar el código Python que la renderiza. El administrador de base de datos puede renombrar una tabla de la base de datos y especificar el cambio en un único lugar, sin tener que buscar y reemplazar en varios archivos.

3.2 PRUEBAS

Las pruebas realizadas a un producto se clasifican en dos categorías: de caja blanca y de caja negra.

"*Las pruebas de caja blanca normalmente se denominan pruebas de cobertura o pruebas de caja transparente, al total de pruebas de caja blanca se le llama cobertura, la cobertura es un número porcentual que indica cuanto código del programa se ha probado*." (36)

Las pruebas de caja blanca, se establecen por medio del diseño de casos que usan como base las estructuras de control del flujo. Estas pruebas se pueden clasificar en tres tipos según sus objetivos:

1. Que se ejecute por lo menos una vez cada instrucción del programa.
2. Garantizar que todas las condiciones se comprueban como verdaderas y falsas.
3. Que se ejecuten los bucles, probando el caso general y los casos extremos.

Las pruebas de caja negra *"(…) son pruebas funcionales, pruebas de entrada/salida o pruebas inducidas por los datos"* (36). Se centran en lo que se espera de un módulo, intentan encontrar casos en que el módulo no se ajusta a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro

La metodología AUP-UCI propone la aplicación de tres tipos de pruebas:

- Pruebas Internas: Son también conocidas como pruebas de cobertura lógica o pruebas unitarias y se clasifican en pruebas de caja blanca.
- Pruebas de liberación: Están diseñadas y ejecutadas por una entidad certificadora de la calidad externa antes de hacerle la entrega al cliente para la aceptación del producto. En el caso del software que se está desarrollando las pruebas de liberación no serán aplicadas porque cuenta con la limitación del tiempo, pues dependen de una entidad externa que retrasaría la entrega del producto.
- Pruebas de aceptación: Es la prueba final antes de desplegar el producto. Son pruebas de caja negra.

3.2.1 PRUEBAS INTERNAS

En las pruebas internas el programador debe verificar el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.

3.2.1.1 PRUEBAS UNITARIAS

Las pruebas unitarias se caracterizan por:

- Ser ejecutadas sin la necesidad de la intervención manual. Esta característica garantiza de la automatización su ejecución.
- Poder repetirse tantas veces como sea necesario. Por este motivo, la rapidez de las pruebas constituye un factor clave.
- Cubrir casi la totalidad del código de la aplicación. Una prueba unitaria será tan buena como su cobertura. La cantidad de código que es sometido a prueba es marcada por la cobertura que tenga el mismo. Por tanto, si la cobertura es baja, significará que gran parte del código está sin probar.
- Poder ejecutarse independientemente del estado del entorno.

Con las pruebas unitarias se logra que la calidad del código mejore y se reduzcan los tiempos de depuración y la corrección de incidencias. Se subyugan drásticamente los problemas y tiempos dedicados a la integración. Las pruebas ayudan a entender mejor el código, ya que sirven de documentación. A través de las pruebas se puede comprender mejor qué hace un módulo y que se espera de él.

3.2.1.1.1 PRUEBAS UNITARIAS EN DJANGO

El framework Django ofrece la posibilidad de aplicarle al código pruebas unitarias para verificar el funcionamiento de los métodos implementados. Las pruebas en Django cubren los modelos de la base de datos y las aplicaciones instaladas de terceros.

Para construir la prueba se debe importar la clase *"TestClase"*, para ejecutarla se utiliza el comando *"python manage.py test"*, se selecciona una clase que hereda de *"TestClase"*. La prueba debe escribirse dentro de un archivo *test,py* dentro de la carpeta *app.*

En el caso de la investigación se realizaron un total de 25 pruebas unitarias. En la siguiente gráfica se muestran los ciclos de pruebas realizados por iteraciones:

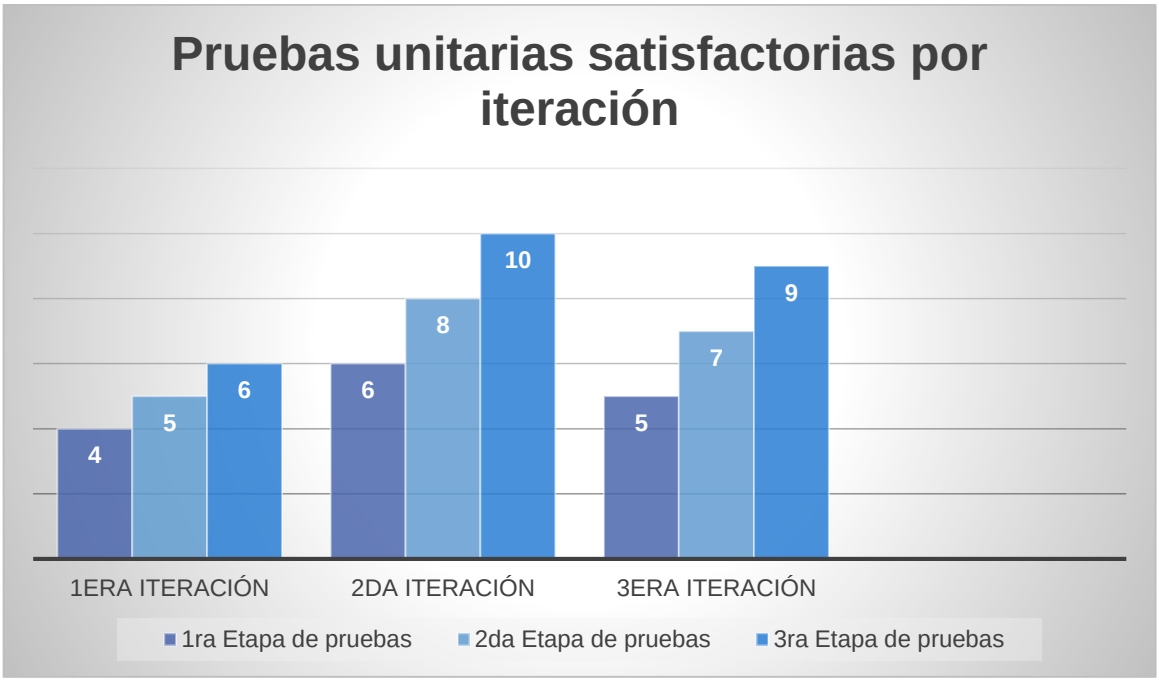


Ilustración 8: Pruebas unitarias satisfactorias por iteración.

3.2.1.1.2 PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación son creadas en base a las historias de usuarios en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada (32).

Las pruebas de aceptación son consideradas pruebas de caja negra. El cliente se considera el responsable de verificar que los resultados que arrojen las pruebas sean correctos. En caso de fallar varias pruebas el cliente debe indicar el orden de prioridad para darle solución. Las historias de usuarios no pueden darse por concluidas hasta que pasen satisfactoriamente todas las pruebas de aceptación.

Según la Guía de los Fundamentos de la Dirección de Proyectos (PMBOK por si sus siglas en inglés) las pruebas de aceptación de usuario equivalen al proceso de verificar el alcance del producto dado que consiste en formalizar la aceptación de los entregables del proyecto. Para verificar el alcance se revisa con el cliente los entregables del proyecto para asegurarse que se han completado satisfactoriamente y obtener la aceptación formal.

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo y el objetivo de estas pruebas no es abarcar todo el código implementado, sino enfocarla hacia el punto de vista del usuario.

El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. En la puesta en práctica de las pruebas de aceptación se siguieron los siguientes pasos:

- Identificar todas las acciones en la HU.
- Para cada acción escribir al menos una prueba.
- Para algunos datos, reemplazar las entradas que hacen que la acción ocurra y llenar en la casilla resultado esperado y resultado obtenido.

Para representar las pruebas de aceptación se definieron los siguientes elementos:

- **Código:** Representa al caso de prueba, incluye el número de HU, de la prueba y si posee diferentes escenarios.
- **HU:** Número de la HU a la cual pertenece.
- **Nombre:** Junto al código, conforma el identificador del caso de prueba.
- **Descripción:** Acción que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- **Entrada/Pasos de Ejecución:** Describe los pasos necesarios para realizar la prueba.
- **Resultados Esperados:** Descripción de la respuesta del sistema ante el caso de prueba.
- **Resultado Obtenido:** Respuesta visual del sistema después de realizar el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

La siguiente tabla muestra las pruebas de aceptación de la HU2 perteneciente a la primera iteración de sistema. La cual fue escogida por ser una de los procesos más relevantes en el desarrollo del sistema.

Tabla 12: HU2_P1 Autenticar Usuario.

Código: HU2_P1	**HU:** 2
Nombre: Autenticar Usuario.	
Descripción: Comprobar que solo se puedan autenticar usuarios válidos.	
Condiciones de Ejecución: **Acceder a:** *http://192.168.216.1:8000*	
Entrada/Pasos de Ejecución: Existencia del usuario en el sistema. La contraseña tiene que ser válida. **Pasos:** 1. Autenticarse en el sistema.	
Resultado Esperado: La aplicación muestra a la izquierda el o los HMI a los que tiene acceso el usuario autenticado y por defecto el primer despliegue del primer HMI.	
Resultado Obtenido: 	

Evaluación de la Prueba: Satisfactoria.

Tabla 13: HU3_P1 Visualizar los datos de los despliegues de los HMI.

Código: HU3_P1	**HU:** 3
Nombre: Visualizar los datos de los despliegues de los HMI.	
Descripción: Mostrar vista de los HMI con sus despliegues.	
Condiciones de Ejecución: **Acceder a: *http://192.168.216.1:8000/main/#***	
Entrada/Pasos de Ejecución: Visualizar los despliegues de los HMI a los que tiene acceso el usuario autenticado en el sistema. **Pasos:** 1. Autenticarse en el sistema. **2.** Seleccionar el HMI que desea desplegar en caso de ser más de uno.	
Resultado Esperado: La aplicación muestra el contenido de los despliegues del HMI seleccionado.	

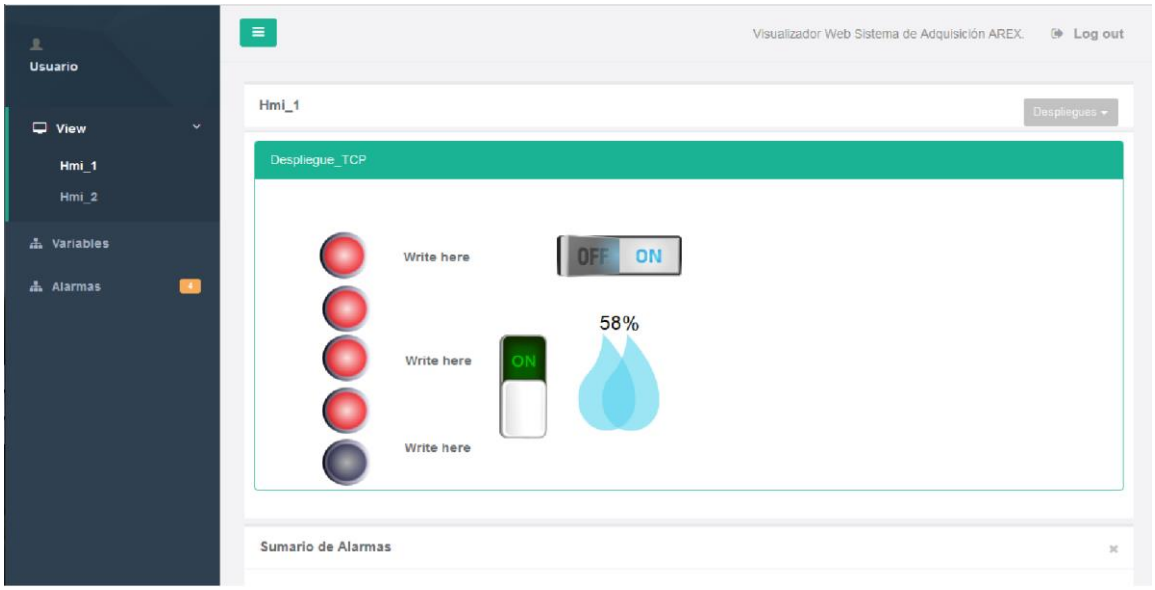
Resultado Obtenido: 
Evaluación de la Prueba: Satisfactoria.

Tabla 14:HU6_P1 Silenciar Alarma

Código: HU6_P1	**HU:** 6
Nombre: Silenciar Alarma.	
Descripción: Mostrar la alarma silenciada.	
Condiciones de Ejecución: **Acceder a: *http://192.168.216.1:8000/main/#***	
Entrada/Pasos de Ejecución: **Pasos:** 1. Ir al sumario de alarmas. 2. Seleccionar la opción silenciar.	

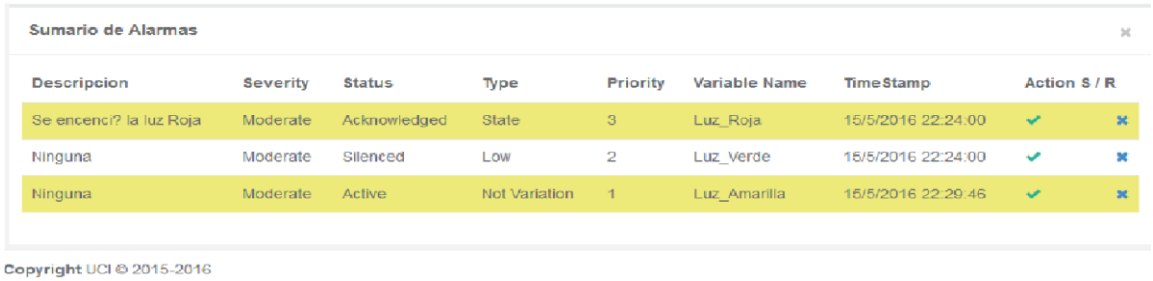
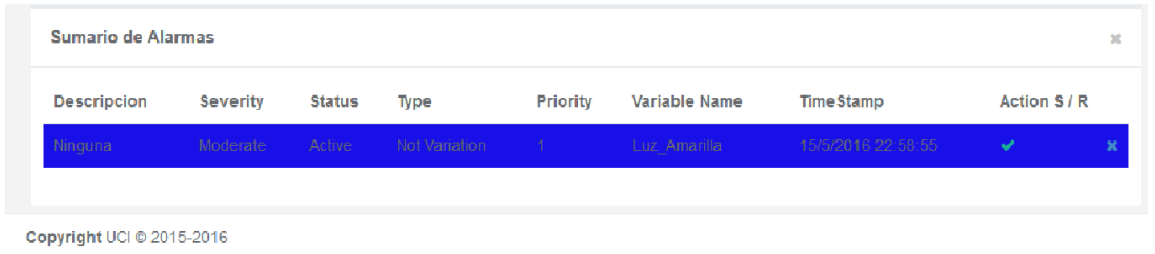
Resultado Esperado: La alarma que se encuentra en estado silenciado deja e parpadear y cambia el estado a silenciar.
Resultado Obtenido: 
Evaluación de la Prueba: Satisfactoria.

Tabla 15: HU6_P1 Eliminar Alarma.

Código: HU6_P1	**HU:** 6
Nombre: Eliminar Alarma.	
Descripción: Quitar de la lista la alarma eliminada.	
Condiciones de Ejecución: **Acceder a: *http://192.168.216.1:8000/main/#***	
Entrada/Pasos de Ejecución: **Pasos:** 1. Ir al sumario de alarmas. 2. Seleccionar la opción eliminar.	
Resultado Esperado: La alarma eliminada desaparece del listado de alarmas.	
Resultado Obtenido:	

Antes

Después



Evaluación de la Prueba: Satisfactoria.

En la primera iteración se realizaron dos etapas de pruebas, en la primera etapa se realizaron 14 casos de prueba donde se obtuvieron 4 no conformidades, las cuales representan un 28,6%. Posteriormente se corrigieron y se realizó una segunda etapa de pruebas donde no se obtuvieron no conformidades, logrando así un 100% de resultados satisfactorios y cumpliendo el criterio de aceptación.

Las no conformidades encontradas en las etapas de pruebas realizadas en la primera iteración se clasifican en funcionalidades que no realizaban la acción prevista y errores visuales donde el sistema no mostraba los datos requeridos. En la ilustración 4 se muestra el comportamiento de las no conformidades en la primera iteración.

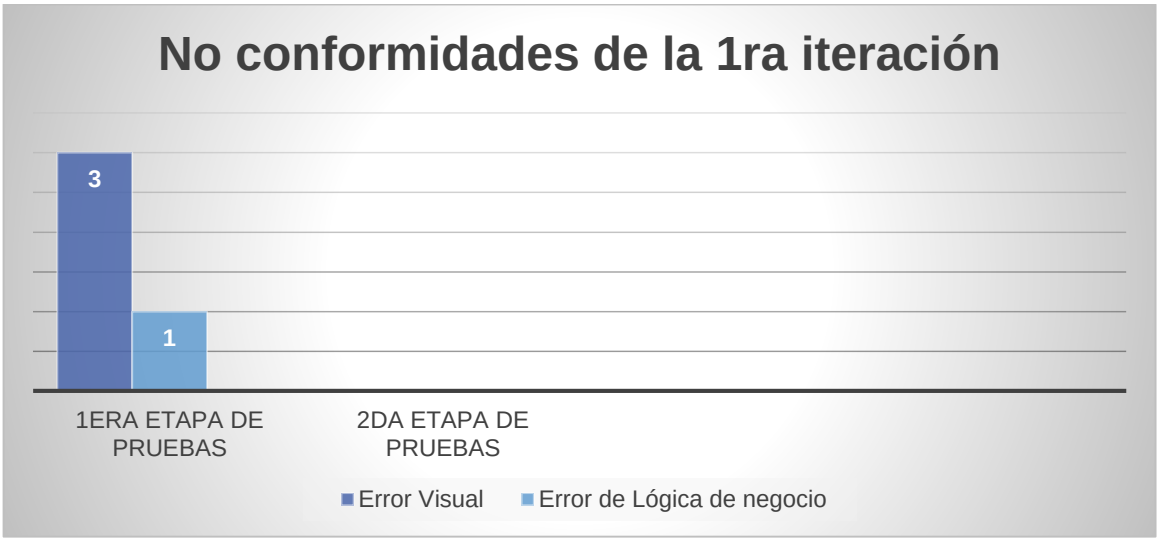


Ilustración 9: Tipo de pruebas fallidas en la primera iteración.

En la segunda iteración se realizaron 21 casos de prueba de los cuales en la primera etapa de pruebas fueron detectadas 6 no conformidades de tipo visual, las cuales representan el 28,6%. En la segunda etapa de pruebas se detectaron 3 no conformidades, también de tipo visual, que representan el 14,3%. En la tercera etapa de pruebas no se detectaron no conformidades logrando así el criterio de aceptación.

En la tercera iteración se realizaron 23 casos de prueba en los cuales se detectaron 9 no conformidades visuales y 2 de lógica de negocio para un total de 11, las cuales representan un 47,8% de pruebas insatisfactorias. En la segunda etapa de pruebas se obtuvieron 4 no conformidades visuales y 1 de lógica del negocio para un total de 5 pruebas insatisfactorias que representan un 21,7%. En la tercera etapa de pruebas se obtuvo 100% de resultados satisfactorios, logrando el criterio de aceptación y dando fin a la fase de las iteraciones. La siguiente imagen muestra las diferentes etapas de pruebas realizadas por iteraciones.

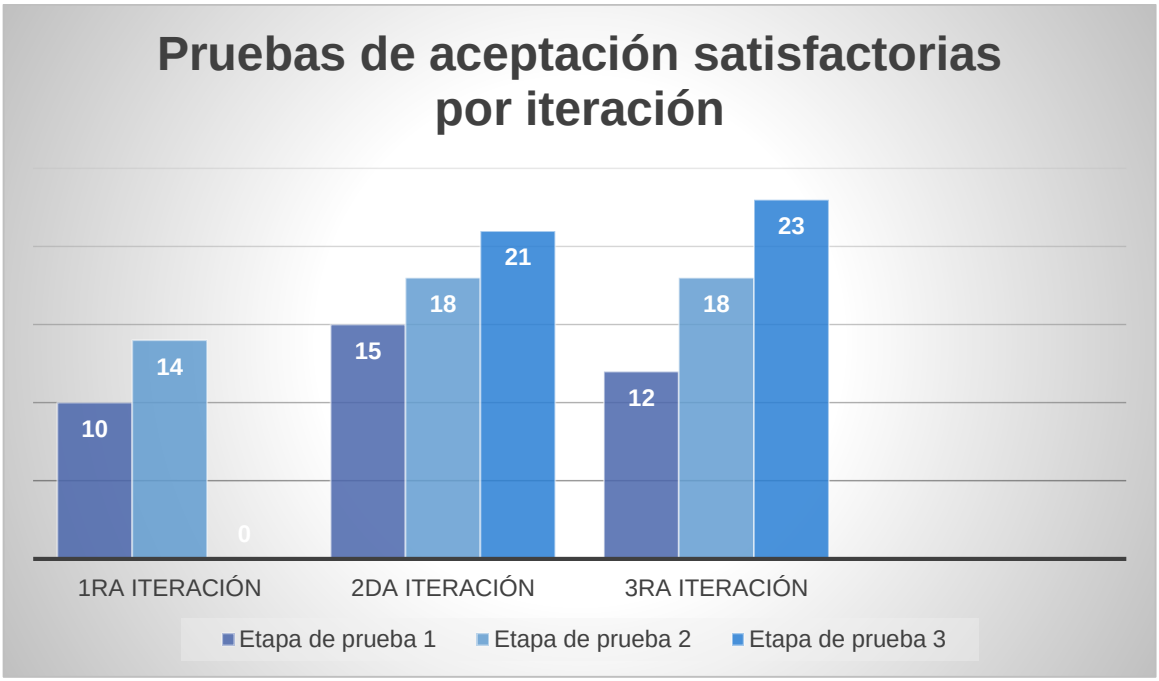


Ilustración 10: Pruebas de aceptación satisfactorias por iteración.

CONCLUSIONES PARCIALES

1. A través del proceso de implementación fue posible obtener el Módulo Ejecución para la web.
2. La aplicación de varios tipos de pruebas hizo posible la obtención de una versión estable del producto, el cual cumple con las exigencias del cliente final.

CONCLUSIONES

Completando los objetivos del presente trabajo podemos arribar a las siguientes conclusiones:

- El empleo de la metodología AUP-UCI fue un factor de importancia para la organización del proceso de desarrollo del Módulo de Ejecución en la web.
- Quedó demostrado que el Módulo Ejecución en la web es necesario para el Sistema de Adquisición Arex brindándole al producto mayor disponibilidad, usabilidad y competencia en el mercado actual.
- Con la implementación del sistema y el desarrollo de las pruebas de aceptación e internas propiciaron la obtención de una versión estable del producto, lo cual propició que cumpliera con las necesidades del cliente.

RECOMENDACIONES

1. Desarrollar los componentes gráficos restantes, tales como: las bocinas, el nivel de CO2, el estado de las puertas, el ventilador, la temperatura, etc.
2. Adicionarle un Módulo Cliente de Base de Datos Histórica para acceder a los históricos de las alarmas y de las variables. Estos valores serían visualizados mediante gráficos y resúmenes.

REFERENCIAS BIBLIOGRÁFICAS

1. **Automatización.** ***Campus Gipuzkoa.*** **[En línea] [Citado el: 26 de 04 de 2016.] http://www.ehu.eus.**

2. Diccionario de Informática y Tecnología. ***Definición ABC.*** **[En línea] 16 de 11 de 2015. [Citado el: 16 de 11 de 2015.] http://www.definicionabc.com/tecnologia/domotica.php.**

3. Domótica inalámbrica para todos los hogares y sin obras. ***COSAS de ARQUITECTOS.*** **[En línea] ASEMAS, 03 de 2014. [Citado el: 16 de 11 de 2015.] http://www.cosasdearquitectos.com/2014/03/domotica-inalambrica-para-todos-los-hogares-y-sin-obras/.**

4. Chalet en Platja d'Aro. ***Proyectos de domótica | Ejemplos de domótica en viviendas, pisos, apartamentos, oficinas y hoteles.*** **[En línea] Domintell, 02 de 2012. [Citado el: 16 de 11 de 2015.] http://www.domintell.es/proyectos-realizados.**

5. Márquez, José E. Briseño. ***TRANSMISIÓN DE DATOS.*** **Mérida : s.n., 2005.**

6. ***Introducción a la arquitectura del "Guardián del ALBA".*** **(CEDIN), Colectivo de autores del Centro de Informática industrial. Carretera a San Antonio Km 2 1⁄2 . Torrens. Boyeros. La Habana. Cuba : s.n., 2013. Arquitectura y Configuración del Guardián del ALBA. pág. 22.**

7. Anónimo. Sistema de Supervisión y Control de Procesos EROS. ***EcuRed.*** **[En línea] [Citado el: 13 de 10 de 2015.] http://www.ecured.cu/index.php/Sistema_de_Supervisi%C3%B3n_y_Control_de_Procesos_EROS.**

8. ***Catálogo de productos y servicios. CEDIN.*** **Ojeda, Yaima Antunez. La Lisa : s.n., 2016.**

9. Sistema de adquisición de Datos. ***Omega Engineering.*** **[En línea] OMEGA. [Citado el: 24 de 11 de 2015.] http://es.omega.com/prodinfo/adquisicion-de-datos.html.**

10. Munguia, Horacio. ***DAQs.*** **[Documento] Hermosillo : s.n., 2015.**

11. Tecnología, Departamento de Ciencia y. ***Introducción a HMI (Interfaz Hombre Máquina).*** **[Conferencia] Quilmes : IACI, 2008.**

12. Laura Vanessa Arenas Montaño, Arturo José Castilla De Cuba, Danilo Alfonso Rojas Méndez. ***INTERFAZ HOMBRE MAQUINA.*** **[Documento] s.l. : Scribd, 2013.**

13. HMI software PASvisu for web-based visualisation. ***PILZ.*** **[En línea] PILZ. [Citado el: 15 de 04 de 2016.] https://www.pilz.com/en-AU/eshop/00105002177122/PASvisu-HMI-Software.**

14. Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. ***METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.*** **2010.**

15. Sánchez, Tamara Rodríguez. ***Metodología de desarrollo para la Actividad productiva de la UCI.*** **Carretera a San Antonio Km 2 1⁄2 . Torrens. Boyeros. La Habana. Cuba : s.n., 2015. pág. 16.**

16. Ventajas y beneficios de las aplicaciones Web. ***Internet Ya Soluciones Web.*** **[En línea] Internet Ya Soluciones Web, 14 de 02 de 2013. [Citado el: 17 de 05 de 2016.] http://www.internetya.co/ventajas-y-beneficios-de-las-aplicaciones-web/.**

17. Duque, Raúl González. ***Python para todos.*** **2013.**

18. Andrés Marzal, Isabel Gracía. ***Introducción a la programación con Python.*** **Castellón : s.n., 2003.**

19. Rossum, Guido van. ***Guía de aprendizaje de Python.*** **s.l. : Python Software Foundation, 2014.**

20. Pérez, Javier Eguíluz. ***Introducción a CSS.*** **2008.**

21. Colaboración. Framework - EcuRed. ***EcuRed.*** **[En línea] http://www.ecured.cu/Framework.**

22. Castellano, Programacion en. Los 4 frameworks web más populares para Python. ***Programación.net.*** **[En línea] 2016. [Citado el: 31 de 05 de 2016.] http://programacion.net/articulo/los_4_frameworks_web_mas_populares_para_python_1069.**

23. Kaplan-Moss, Adrian Holovaty y Jacob. ***El libro de Django .*** **2008.**

24. MONTERO, SERGIO INFANTE. ***Maestros del web.*** **2012.**

25. Anónimo. CSS Frontend Frameworks: The Best 10 for Modern Web Design. ***Noeticforce.*** **[En línea] 2015. [Citado el: 31 de 05 de 2016.] http://noeticforce.com/css-front-end-frameworks-for-web-development-and-design.**

26. Duque, Raúl Gonzales. ***Python para todos.*** **España : Creative Commons Reconocimiento 2.5, 2008.**

27. Con PyCharm, los desarrolladores de Python finalmente obtienen una IDE poderosa. ***PR Newswire.*** **[En línea] A UBM plc company. [Citado el: 20 de 01 de 2016.] http://www.prnewswire.com/news-releases/con-pycharm-los-desarrolladores-de-python-finalmente-obtienen-una-ide-poderosa-104948549.html.**

28. Suaza, Katerine Villamizar. ***Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software.*** **Medellín : s.n., 2013.**

29. Joskowicz, José. ***Reglas y Prácticas en eXtreme Programming.*** **Vigo : s.n., 2008.**

30. Larm, Craing. ***UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado.*** **Vancouver : Rational Software Canada, 2004.**

31. ***Estructura de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC).*** **Mestras, Juan Pavón. Madrid : s.n., 2009.**

32. Díaz Vivar, Miriam Rosana Miranda, María Gabriela Molina, Sonia Elizabeth . ***PAS_PLAN DE GESTION DE PRUEBA.*** **Santa Cruz : s.n., 2010.**

33. José H. Canós, Patricio Letelier yMª Carmen Penadés. ***Métodologías Ágiles en el Desarrollo de Software.*** **Valencia : s.n., 2004.**

34. ***Unidad III Diseño y Programación de Interfaces HMI.*** **Rojas, M.C. Juan Carlos Olivares. Monterry : s.n., 2015. págs. 09-13.**

35. Los diferentes lenguajes de programación para la web. ***Maestros del Web.*** **[En línea] Platzi, 2012. [Citado el: 15 de 12 de 2015.] http://www.maestrosdelweb.com/los-diferentes-lenguajes-de-programacion-para-la-web/.**

36. ***Arex, una solución minimalista para la supervisión y control de procesos.*** **Ing. Antonio Cedeño Pozo, Ing. Karel Delgado Alón, Ing. Alejandro Jiménez López, Ing. Alexander Moreno Limonte, Álvaro Denis Acosta. La Lisa : s.n., 2013.**

37. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA . ***ALEGSA.*** **[En línea] ALEGSA, 2010. http://www.desarrolloweb.com/wiki/framework.html.**

38. ***INTRODUCCIÓN A LA ARQUITECTURA DEL "GUARDIÁN DEL ALBA".*** **Anónimo. Cuidad de la Habana : s.n., 2013.**

39. Anónimo. Sistema de adquisición de Datos. ***Omega Engineering.*** **[En línea] OMEGA, 25 de 04 de 2012. [Citado el: 18 de 01 de 2016.] http://es.omega.com/prodinfo/adquisicion-de-datos.html.**

40. ***Sistema SCADA.*** **Yanet Nieto Doce, Ariangna Garces Gilart, Luis Enrique García Hernández, Ariel Chávez Lorenzo y Amado Espinosa Hidalgo. La Habana : s.n., 2013.**

41. Santana, Alicia Bárbara Expósito. ***Pruebas de caja blanca.*** **2012.**

42. Anónimo. Capítulo 4. DOM (Document Object Model) (Introducción a AJAX). ***LibrosWeb.*** **[En línea] [Citado el: 31 de 05 de 2016.] http://librosweb.es/libro/ajax/capitulo_4.html.**

43. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley. ***Elements of Reusable Object-Oriented Software.***

44. —. ***Design Patterns: Elements of Reusable Object-Oriented Software.*** **s.l. : Produced by KevinZhang , 2011.**