Universidad de las Ciencias Informáticas

Título: Sistema informático para el control automatizado de dispositivos actuadores y sensores

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

Autor: Carlos Pérez Mateu

Tutores: MSc. Edistio Yoel Verdecia Martínez
Ing. Ailyn Gutiérrez Ferrera

Ciudad de La Habana

Junio de 2010

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 1 de la Universidad de las Ciencias Informáticas, así como a dicho centro, para que lo usen según estimen pertinente.

Para que así conste, firmo la presente a los **4** días del mes de **Junio** del año **2010**

Carlos Pérez Mateu
Firma del Autor

MSc. Edistio Yoel Verdecia Martínez
Firma del Tutor

Ing. Ailyn Gutiérrez Ferrera
Firma del Tutor

DEDICATORIA

Les dedico este trabajo a mis padres que han sido parte fundamental en el desarrollo del mismo y que desde el propio nacimiento de esta idea han trabajado incansablemente para que saliese el producto final. También dedicárselo a mi principal inspiradora en la investigación científica en los últimos cinco años, la UCI, sin la cual no hubiese alcanzado el desarrollo científico y técnico con que cuento.

AGRADECIMIENTOS

Agradezco en primer lugar a mi familia, no solo por todo el trabajo que han asumido a mi lado, sino también por el apoyo emocional y la seguridad que me han dado durante todo este proceso. A mi madre por ser casi una compañera de tesis y por soportarme en los momentos de estrés y preocupación y a mi padre, que aunque su apoyo no fue intelectual, le agradezco toda la ayuda técnica y esa infinita creatividad sin las cuales no fuese posible este trabajo.

Agradezco además a mis tutores que han jugado un importante papel y a mis compañeros, en especial a Richard Rivero, que han compartido conmigo todas las ideas y sueños que dieron vida a esta investigación.

RESUMEN

El objetivo del uso de la domótica es generar servicios para el aumento del confort, la gestión energética y la seguridad; estos se conocen como Pilares de la domótica. En la actualidad existen diversas tecnologías y programas informáticos para las aplicaciones domóticas, pero son costosas o no se ajustan a las condiciones socio-económicas que caracterizan la realidad cubana. Es por ello que este trabajo tiene como objetivo desarrollar un sistema informático que permita controlar dispositivos y procesos domésticos de forma automatizada, mediante la conexión con el hardware necesario. Como metodología para el desarrollo del sistema domótico se seleccionó RUP y como herramienta de modelado *Rational Rose*. Además se utilizó *NetBeans* como herramienta de desarrollo Java, como lenguaje de modelado UML y como lenguaje de programación Java. El modelo de dominio, los requerimientos y la especificación de los casos de uso permiten entender el contexto en que se ubica la aplicación. Se ofrece una visión de cómo está estructurada y qué función tienen las partes. Con la fase de análisis y diseño se logró asignar las responsabilidades a los componentes del software. Además, se crearon los artefactos necesarios para la implementación del sistema. Durante el proceso de implementación se desarrolló la aplicación, fueron implementadas las clases obtenidas en el diseño y se obtuvieron los diagramas de componentes y de despliegue. La fase de prueba se concretó mediante el método de caja negra, el cual permitió demostrar que la aplicación cuenta con las características y funcionalidades para las que fue concebida.

ÍNDICE

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE TABLAS

Pág.

INTRODUCCIÓN

En las últimas décadas se ha producido un vertiginoso crecimiento de las tecnologías de la electrónica y la informática, el cual ha alcanzado todas las esferas y ha inundado nuestro entorno con televisores, telefotos, equipos de fax y módem, redes, computadoras y sistemas informáticos, tanto en oficinas como en viviendas particulares.

En la actualidad las oficinas y viviendas son parte fundamental en la vida del hombre, por lo que el control del sistema eléctrico, la climatización, ventanas, gas, red telefónica y otros; se hacen necesarios para garantizar una adecuada gestión energética, mayor confort y mecanismos de seguridad personal y patrimonial. Como parte de este fenómeno tecnológico y cultural surge la domótica, que ha venido a dar respuesta a las necesidades de control y automatización de las viviendas modernas, y a la humana y natural aspiración de una mejor calidad de vida reflejada en los esfuerzos por dotar al hogar de mayor confort, esperando disfrutar de un ambiente protector para la familia.

La etimología del término domótica se encuentra en la unión de las palabras “domus” (en latín casa) y “tica” de automática (que en griego significa que funciona por sí sola). Por domótica se entiende al conjunto de sistemas que controlan y automatizan una vivienda, ofreciendo servicios de gestión energética, seguridad y confort. Como se aprecia uno de los pilares de la domótica es la gestión energética, aspecto esencial en el mundo de hoy, donde el irracional uso de los combustibles fósiles constituye una amenaza para la humanidad.

El agotamiento de las fuentes tradicionales de energía, así como el calentamiento global y otros fenómenos que están afectando el medio ambiente, imponen la necesidad de ahorrar energía para poder tener un mundo mejor y en equilibrio; además exigen de todos llevar a cabo planes de ahorro de energía e implementar nuevas y renovadoras ideas. La enorme revolución energética llevada a cabo en Cuba en los últimos años y las reflexiones hechas por el Comandante sobre la necesidad de ahorrar energía; se enmarcan dentro de los esfuerzos globales por preservar la especie humana.

El Comandante en Jefe, Fidel Castro Ruz, en la Conferencia de Naciones Unidas sobre Medio Ambiente y Desarrollo, efectuada en Río de Janeiro, Brasil, en julio de 1992 expresó:

> ***“Una importante especie biológica está en riesgo de desaparecer por la rápida y progresiva liquidación de sus condiciones naturales de vida: el hombre…***
>
> ***… Si se quiere salvar a la humanidad de esa autodestrucción, hay que distribuir mejor las riquezas y las tecnologías disponibles en el planeta…***

…No más transferencias al Tercer Mundo de estilos de vida y hábitos de consumo que arruinan el medio ambiente. Hágase más racional la vida humana…

…Utilícese toda la ciencia necesaria para el desarrollo sostenido sin contaminación. Páguese la deuda ecológica y no la deuda externa. Desaparezca el hambre y no el hombre."[1]

En la mencionada Conferencia sobre Medio Ambiente y Desarrollo se adoptó la agenda 21 en la cual se establecieron líneas estratégicas y metas a alcanzar en estos temas, sin embargo a escala global poco se ha avanzado pues, los que más pueden hacer, los poderosos, han destinado sólo una parte insignificante de sus recursos y esfuerzos a tales propósitos. Cuba, a pesar de ser un país pequeño, ha realizado grandes esfuerzos intelectuales y materiales en defensa del medio ambiente y de la conservación y desarrollo del mundo. El potencial científico creado por la Revolución y la sociedad en general, han trabajado de forma coherente y mancomunada en la estrategia trazada por el estado cubano para dar cumplimiento a la agenda 21.

Especial importancia tiene dentro de la labor desarrollada, la Revolución Energética, que ha alcanzado la generación, la distribución y el consumo de cada uno de los hogares del país, aspecto este último en el que el uso de la domótica puede contribuir al logro de un ahorro significativo, mediante acciones entre las que se pueden mencionar:

- ✓ El control de la iluminación, que puede ser regulada en función del nivel de luminosidad ambiente, evitando su encendido innecesario o adaptándola a las necesidades del usuario.
- ✓ La activación de la iluminación puede estar en función de la ocurrencia de uno u otros eventos.
- ✓ La regulación de la climatización de acuerdo con las necesidades concretas.
- ✓ La eliminación de los consumos de energía de los equipos por concepto de stand-by mediante la conexión y desconexión de los mismos desde el sistema domótico.

Otro de los pilares del uso de la domótica está relacionado con la seguridad y el mismo puede estar vinculado a múltiples aristas que van desde la toma de decisiones frente a diferentes eventos climáticos, hasta la implementación de diversos tipos de alarmas y métodos de seguridad, como pueden ser:

- ✓ Control de puertas y ventanas atendiendo a causas climatológicas (lluvias, fuertes vientos y otros). Las puertas y ventanas también pueden ser controlados de forma automática en función de la seguridad frente a la detección de fuego, humo, gas, o la presencia de un intruso.

[1] CASTRO RUZ, F. Discurso. Conferencia de Naciones Unidas sobre Medio Ambiente y Desarrollo. Río de Janeiro, Brasil, 1992.

- ✓ La activación de sistemas de alarma contra incendios, humo, gas o contra intrusos
- ✓ Seguridad de los bienes, gestión del control de acceso y control de presencia, así como la simulación de presencia.

La revisión bibliográfica realizada permitió identificar la existencia en el mercado, a escala mundial, de diversas tecnologías en el área de la domótica. En realidad, con los nuevos sistemas que se están comercializando, el control y su programación son intuitivos. También, las posibilidades que dan la conexión a Internet, con redes de banda ancha, o la conexión a través de redes móviles para el control remoto y la vigilancia, hacen que se extienda muchísimo el campo de aplicación de la domótica.

La realidad antes descrita está fuera del alcance de las mayorías en los países pobres y en particular de Cuba, que está sometida a un férreo bloqueo económico y cuyas posibilidades de inversión están dirigidas a otros sectores prioritarios.

Los elementos hasta aquí descritos motivaron al autor a trabajar en el diseño e implementación de un sistema informático para la domótica, que satisfaga las necesidades básicas de control y automatización de la vivienda cubana de hoy. Se trata de un sistema que permitirá operar los dispositivos electrónicos de la casa sin tener que acudir a la ayuda de profesionales que deben realizar complicadas reprogramaciones cada vez que se desee incorporar un nuevo equipo al sistema de control inteligente.

El camino hacia el hogar digital es cada vez más inminente dados los avances tecnológicos que se suceden gracias al trabajo y permanente afán de innovación, no sólo de empresas sino también de instituciones educativas o de investigación. El presente trabajo es una muestra de ello, y en el mismo se identificó como **problema científico** ¿Cómo controlar dispositivos y procesos domésticos de forma automatizada con el uso de la informática?, de él se deriva el **objeto de estudio** que son los sistemas domóticos.

El **objetivo** de este trabajo es desarrollar un sistema informático que permita controlar dispositivos y procesos domésticos de forma automatizada, mediante la conexión con el hardware necesario. En consecuencia el **campo de acción** se refiere a los sistemas informáticos en la domótica.

Para conducir la investigación encaminada a solucionar el problema científico planteado se formuló como **hipótesis:** con la utilización de un sistema informático en la domótica se logra el control de dispositivos y procesos domésticos de forma automatizada.

Variable Independiente: Sistema informático.

Variable Dependiente: Control de dispositivos y procesos domésticos.

	Variables	Dimensiones	Indicadores	Índice de los Indicadores
Operacionalización de las variables	Sistema informático	Apariencia o interfaz externa	Claridad de la información mostrada	Buena
				Regular
				Mala
			Facilidad de interacción	Sencilla
				Media
				Compleja
		Rendimiento	Velocidad de procesamiento	Alta
				Media
				Baja
			Tiempo de respuesta	Alto
				Medio
				Bajo
		Portabilidad	Del código	Verdadero
				Falso
			De las librerías y drivers	Verdadero
				Falso
		Ayuda y documentación	Claridad de la información mostrada	Buena
				Regular
				Mala
			Facilidad de interacción	Sencilla
				Media
				Compleja
	Control de dispositivos y procesos domésticos	Control de dispositivos	Cantidad de actuadores que se controlan	Todos
				Algunos
				Ninguno
			Velocidad de lectura de los sensores	Alta
				Media
				Baja
			Eficiencia en el uso de la lectura de los sensores para el control de los actuadores.	Alta
				Media
				Baja
		Control de procesos	Cantidad de procesos que se controlan	Todos
				Algunos
				Ninguno
			Eficiencia en el uso de la lectura de los sensores para el control de los procesos.	Alta
				Media
				Baja
			Eficiencia de la automatización de los procesos.	Alta
				Media
				Baja
		Reglas para el control	Frecuencia de cumplimiento de las reglas	Siempre
				A veces
				Nunca
			Eficiencia de las reglas definidas	Alta
				Media

Tabla 1: Operacionalización de las variables

Durante el proceso investigativo fueron planteadas y resueltas las **tareas** de investigación que se presentan a continuación:

1. Determinación de los antecedentes y referentes teóricos del uso de sistemas informáticos en el control doméstico.
2. Selección de la metodología, herramientas y tecnologías para satisfacer las necesidades y características del sistema informático.
3. Elaboración del sistema para controlar dispositivos y procesos domésticos de forma automatizada; incluyendo análisis, diseño, implementación y prueba.

A continuación se reseña el uso dado a diferentes **métodos teóricos** en el desarrollo de la investigación:

- ✓ Mediante el **método analítico – sintético** fueron analizados teorías y documentos relacionados con la informática en el control doméstico automatizado, lo cual permitió buscar rasgos que lo caracterizan y aspectos para elaborar y fundamentar la propuesta de solución al problema planteado.
- ✓ La **modelación** fue utilizada para crear modelos que permitieron investigar la realidad y modelar la solución. Mediante la metodología RUP (*Rational Unified Process*) se hizo la modelación de negocio y de sistema.

La tesis está estructurada en introducción, cuatro capítulos, bibliografía, conclusiones y anexos. En el capítulo 1 se presentan los principales conceptos asociados a la domótica y al uso de la informática en esta rama, haciéndose un análisis de la situación actual. Se realiza además la fundamentación de la metodología, los lenguajes y las herramientas utilizadas para el diseño e implementación del sistema que da solución al problema planteado.

En el desarrollo del capítulo 2 se presenta el modelo de dominio, los requerimientos funcionales y no funcionales con los que debe cumplir el sistema propuesto, así como sus actores y diagrama de casos de uso con la descripción expandida de cada uno de ellos.

El capítulo 3 abarca todo lo referido al análisis y diseño del sistema a través del modelo de análisis como solución de la propuesta. Se describen los patrones de diseño utilizados para la confección de los diagramas de clase e interacción, así como la arquitectura a utilizar en el desarrollo del sistema.

En el capítulo 4 y final se muestra el modelo de implementación además del diagrama de despliegue, se especifican las pautas usadas para la implementación del sistema, así como las principales funcionalidades de la aplicación.

Capítulo 1 Fundamentación teórica

En este capítulo se realiza una caracterización del estado del arte de la domótica, que es el tema central de la tesis, para ello se abordan aspectos como: el concepto de domótica, sus pilares, características de los sistemas, dispositivos usados y ejemplos de tecnologías y sistemas que existen en la actualidad. Además se refieren las principales características del puerto paralelo, dado que es la vía que se utilizará para la entrada y salida de información de la computadora al resto del sistema. El capítulo concluye con la presentación de la metodología empleada para la elaboración de la aplicación informática, así como los lenguajes y herramientas utilizadas durante el proceso.

1.1 Domótica

La domótica nace a escala pública en EEUU en la década del 70, con el objetivo principal de generar un ahorro en los consumos de energía. En sus inicios se introdujo en las grandes industrias de sectores como el espacial y el químico. Con el estudio en este campo tecnológico y el alcance de nuevas soluciones, comienzan a aparecer los primeros edificios inteligentes, con tecnologías de altos costos por lo cual eran utilizada principalmente en edificios con gran consumo tales como, hospitales, hoteles y sedes de grandes corporaciones. En los últimos años se ha producido un abaratamiento en los costos de fabricación de los productos tecnológicos y en consecuencia un incremento en la accesibilidad a ellos, surgiendo así múltiples aplicaciones en amplios sectores sociales.

1.1.1 ¿Qué es la domótica?

Millán Tejedor, en si libro Domótica: edificios inteligentes, plantea: “El término domótica proviene de la unión de las palabras domus (que significa casa en latín) y tica (de automática, palabra en griego, “que funciona por sí sola”). Se entiende por domótica al conjunto de sistemas capaces de **automatizar** una vivienda, aportando **servicios** de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo **control** goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto.”[2]

Por otra parte en el portal Web Casadomo se define “La domótica es la **automatización** y **control** centralizado y/o remoto de aparatos y sistemas eléctricos y electrotécnicos en la vivienda. Los

[2] MILLAN TEJEDOR, R. Domótica: edificios inteligentes. Barcelona, Anaya Multimedia S.A., 2004. 17

objetivos principales de la domótica son aumentar **el confort, ahorrar energía y mejorar la seguridad**"[3]

Pérez Saucedo y Reyes Padilla en su artículo "Domótica", publicado en el sitio Web Monografías.com; la definen como: "Domótica es el término "científico" que se utiliza para denominar la parte de la tecnología (electrónica e informática), que integra **el control** y supervisión de los elementos existentes en un edificio de oficinas o en uno de viviendas o simplemente en cualquier hogar."[4]

Otras definiciones que se le han ido dando al término Domótica en los últimos años se recogen en el artículo Introducción y tecnologías de la Domótica, publicado en el sitio Web Domótica Soluciones Integrales SL, las cuales se citan a continuación:

1) "La nueva tecnología de los **automatismos de maniobra**, gestión y **control** de los diversos aparatos de una vivienda, que permiten aumentar **el confort del usuario, su seguridad y el ahorro en el consumo energético**.
2) Un conjunto de **servicios** en las viviendas, asegurados por sistemas que realizan varias funciones, pudiendo estar conectados, entre ellos, y a redes internas y externas de comunicación.
3) La informática aplicada a la vivienda. Agrupa el conjunto de sistemas de **seguridad y de la regulación de las tareas domésticas** destinadas a facilitar la vida cotidiana **automatizando** sus operaciones y funciones."[5]

El término de domótica o casa inteligente presenta múltiples versiones en diferentes países e idiomas, pero las más utilizadas son: "casa inteligente" (*smart house*), automatización de viviendas (*home automation*), domótica (*domotique*), sistemas domésticos (*home systems*), entre otros.

De manera general, un sistema domótico dispondrá de una red de comunicación y diálogo que permite la interconexión de una serie de equipos a fin de obtener información sobre el entorno doméstico y, basándose en ésta, realizar determinadas acciones sobre dicho entorno. En esencia su funcionamiento consiste en que mediante elementos de campo (detectores, sensores, captadores, etc.), se captan y transmiten señales a una unidad central inteligente, que procesa la información recibida y actúa sobre determinados circuitos de potencia relacionados con las señales recogidas por los elementos de campo correspondientes, ejecutando las acciones pertinentes en cada momento.

[3] Casadomo Soluciones SL, Servicios de una casa inteligente, 2006, http://www.casadomo.com

[4] REYES PADILLA, K. G. Domótica, 2004. http://www.monografias.com/trabajos35/domotica/domotica.shtml

[5] TRIANA GONZÁLEZ, R., Domótica Soluciones Integrales SL. Introducción y tecnologías de la Domótica, 2008, http://www.domotica.net

Como se puede apreciar en las definiciones presentadas se han destacado palabras o frases claves que expresan la esencia del concepto de domótica y en la tabla siguiente se muestra un resumen de ellas, que permite fundamentar los puntos de vista adoptados por el autor con respecto al concepto de domótica.

Palabras y frases claves	Definiciones de domótica presentadas					
	1era	2da	3era	4ta	5ta	6ta
Automatización	x	x		x		x
Control	x	x	x	x		
Servicios para el aumento del confort, la gestión energética y la mejora de la seguridad	x	x		x	x	x

Tabla 2: Rasgos comunes de las definiciones de domótica.

En resumen el concepto domótica se refiere a la **automatización** y **control** (encendido / apagado, apertura / cierre y regulación) de aparatos y sistemas de instalaciones eléctricas y electrotécnicas (iluminación, climatización, persianas y toldos, puertas y ventanas motorizados, etc.) de forma centralizada y/o remota. El objetivo del uso de la domótica es generar **servicios para el aumento del confort, la gestión energética y la mejora de la seguridad**; estos se conocen como “Pilares de la domótica” y serán abordados a continuación.

1.1.2 Pilares de la domótica

En la mayor parte de la bibliografía consultada se utiliza este término para referirse a los servicios que ofrece la domótica y aunque algunos autores incluyen también “las comunicaciones” y “la telegestión y accesibilidad”; lo más generalizado es considerar tres pilares que son: gestión energética, confort y seguridad.

La gestión energética está dada por la posibilidad del sistema domótico de encargase de gestionar el consumo de energía, mediante actuadores y sensores que controlan el apagado y encendido de luces y equipos electrodomésticos en función de las condiciones de presencia, temperatura, horarios y otras, de un modo automático.

Con respecto al confort, la domótica proporciona una serie de comodidades relacionadas con el control automático de servicios como: la climatización, la iluminación, la apertura y cierre de puertas y ventanas entre otros.

Relacionado con la seguridad se debe destacar que la que proporciona un sistema domótico es más amplia que la que puede proporcionar cualquier otro sistema, pues integra tres campos de la seguridad que normalmente están controlados por sistemas distintos que son los siguientes:

1. Seguridad de los bienes: en esta dirección se gestiona el control de acceso y de presencia y es posible simular mediante un sistema domótico la presencia de personas en el inmueble.
2. Seguridad de las personas: especialmente para las personas mayores, personas minusválidas y enfermas. Se puede tener acceso automático mediante un nodo telefónico por ejemplo a la policía.
3. Incidentes y averías: mediante sensores, se pueden detectar los incendios y las fugas de gas y agua, la ocurrencia de tormentas o descargas eléctricas y tomar las medidas necesarias que pueden ser entre otras la activación de una alarma, la llamada a los bomberos o a la policía, la apertura o cierre de ventanas o la desconexión de equipos electrodomésticos.

1.1.3 Características de los sistemas domóticos

Los sistemas domóticos en la actualidad tienen características generales entre las que se destacan: integración, interrelación, facilidad de uso, control remoto, fiabilidad y actualización, a continuación se presenta una breve explicación de cada una de ellas.

- ✓ ***Integración***: El sistema integra bajo su control el funcionamiento de los múltiples sistemas que se decida incluirle, de este modo, los usuarios no tienen que estar pendientes de los diversos equipos autónomos, con su propia programación, indicadores situados en diferentes lugares, dificultades de interconexión entre equipos de distintos fabricantes; sino que es el propio sistema quien lo hace centralizadamente. Esta es la principal virtud que debe tener un sistema domótico, dado que permite el control del inmueble y por tanto deberá integrar todos sus sistemas
- ✓ ***Interrelación***: Una de las principales características que debe ofrecer un sistema domótico es la capacidad para relacionar diferentes elementos y obtener una gran versatilidad y variedad en la toma de decisiones. Así, por ejemplo, es sencillo relacionar el funcionamiento del aire acondicionado con el de otros electrodomésticos, con la apertura de ventanas, con que la habitación esté ocupada o con la temperatura existente.
- ✓ ***Facilidad de uso***: Con sólo mirar las interfaces de control se puede tener una completa información del estado de todos los equipos o sistemas del inmueble y desde allí se puede realizar cualquier modificación de modo sencillo, incluida la incorporación o eliminación de elementos (partes del inmueble, equipos electrodomésticos, etc.) del control del sistema. Por lo general el uso de una interfaz gráfica propicia una comunicación fácil con el usuario y le permite rápidamente obtener información sobre aspectos como: la temperatura, humedad, los equipos que están funcionando, si hay alguien en las proximidades, o el nivel de agua que tiene el tanque.

- ✓ ***Fiabilidad***: Está avalada por la potencia, rapidez y fiabilidad que tienen las computadoras actuales, unida a la utilización de un sistema de alimentación ininterrumpida que garantice el funcionamiento de las prestaciones básicas del sistema.
- ✓ ***Actualización***: Mantener actualizado el sistema es tan sencillo como la simple instalación de nuevas versiones o actualizaciones del programa informático en la computadora, lo cual constituye un proceso elemental y con consumo de tiempo casi nulo. Esto es posible dado que la lógica del funcionamiento completo del sistema recae sobre el software y no en los equipos que se encuentren instalados al sistema, que serán automáticamente reconocidos por la nueva versión instalada.
- ✓ ***Control remoto***: Esta es una potencialidad de los sistemas domóticos surgida con el desarrollo experimentado por las comunicaciones y está dada por el uso desde cualquier lugar del mundo, de las mismas posibilidades de supervisión y control disponibles localmente, mediante conexión telefónica desde otra computadora.

1.1.4 Dispositivos usados en la domótica

Con el desarrollo experimentado por la electrónica y la informática a escala mundial, han surgido múltiples dispositivos y medios de comunicación que son utilizados por los sistemas domóticos, además la domótica propiamente ha demandado el surgimiento de nuevos dispositivos y todos ellos se pueden clasificar en uno de los cinco grupos siguientes:

- ✓ ***Controladores***: Los controladores son los dispositivos que gestionan el sistema según la programación y la información que tienen o que reciben. Puede haber uno o varios controladores distribuidos por el sistema.
- ✓ ***Actuadores***: Los actuadores son los dispositivos capaces de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema (apagar/encender, subir/bajar, apertura/cierre, etc.).
- ✓ ***Sensores***: Los sensores son los dispositivos que monitorean el entorno captando información y transmitiéndola al sistema, por ejemplo, sensores de movimiento, agua, gas, humo, temperatura, viento, humedad, lluvia, iluminación, etc.
- ✓ ***Bus***: Es el medio de transmisión que transporta la información entre los distintos dispositivos y puede ser por un cableado propio, por las redes de otros sistemas (red eléctrica, red telefónica, red de datos) o de forma inalámbrica.

- ✓ ***Interfaces***: Las interfaces son los dispositivos (pantallas, móviles, Internet, teclados) y los formatos (binario, audio, gráficos) en que se muestra la información del sistema para los usuarios, con las cuales pueden interactuar.

Es preciso destacar que todos los dispositivos del sistema de domótica no tienen que estar físicamente separados, sino varias funcionalidades pueden estar combinadas en un dispositivo, por ejemplo un equipo puede ser compuesto por un controlador, actuadores, sensores y varias interfaces. En la figura 1 se muestran ejemplos de dispositivos usados por la domótica y del modo en que interactúan.

Figura 1: Ejemplos de dispositivos de un sistema domótico

1.1.5 Tipos de arquitecturas

Al referirse a la arquitectura de los sistemas de domótica se tiene en cuenta la estructura de su red y tomando como base dónde reside "la inteligencia del sistema" existen clasificaciones de varios autores, en la tesis se asume la que aparece en el "portal CASADOMO.com - Todo sobre Domótica e Inmótica del Edificio y Hogar Digital"; donde se plantea que existen cuatro arquitecturas que son: centralizada, descentralizada, distribuida y mixta o híbrida. A continuación se presentarán las características básicas de cada una de estas arquitecturas y figuras que las ilustran.

"Arquitectura centralizada – En un sistema de domótica de arquitectura centralizada, un controlador centralizado, envía la información a los actuadores e interfaces según el programa, la configuración y la información que recibe de los sensores, sistemas interconectados y usuarios."[6] Observe figura 2.

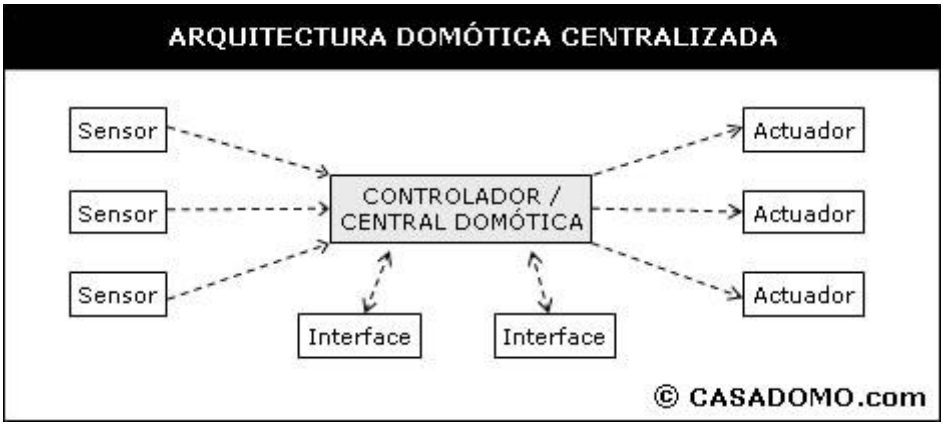

Figura 2: Esquema de Arquitectura Centralizada

"Arquitectura Descentralizada – En un sistema de domótica de Arquitectura Descentralizada, hay varios controladores, interconectados por un bus, que envía información entre ellos y a los actuadores e interfaces conectados a los controladores, según el programa, la configuración y la información que recibe de los sensores, sistemas interconectados y usuarios." [7]. Vea figura 3.

El sistema elaborado en la tesis se corresponde, básicamente, con este tipo de arquitectura descentralizada, donde existen varios controladores que se comunican entre sí, a los cuales se les conectan diversos sensores y actuadores.

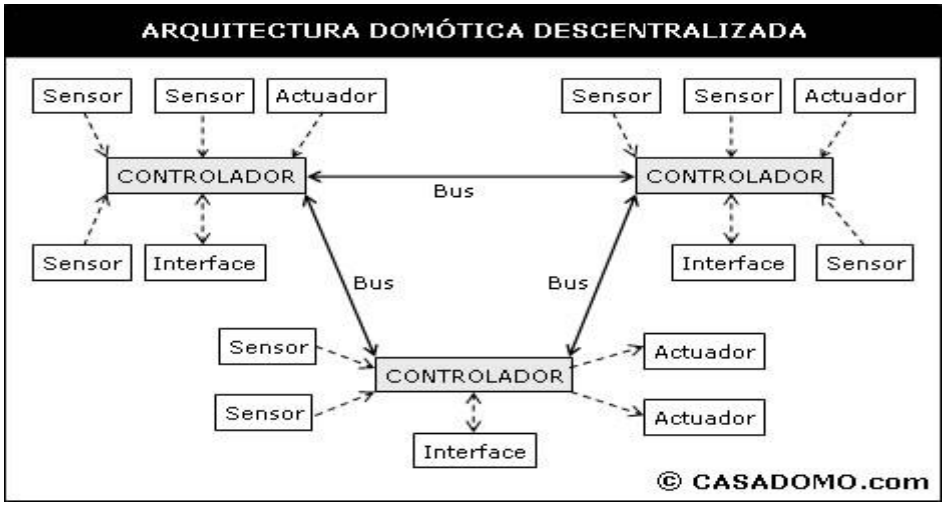

Figura 3: Esquema de Arquitectura Descentralizada.

[6] Casadomo Soluciones SL, Servicios de una casa inteligente, 2006, http://www.casadomo.com

[7] Casadomo Soluciones SL, Servicios de una casa inteligente, 2006, http://www.casadomo.com

En la figura 4 se ilustra la arquitectura distribuida, y sobre ella se puede afirmar que "…en un sistema de domótica de arquitectura distribuida, cada sensor y actuador es también un controlador capaz de actuar y enviar información al sistema según el programa, la configuración, la información que capta por sí mismo y la que recibe de los otros dispositivos del sistema."[8]

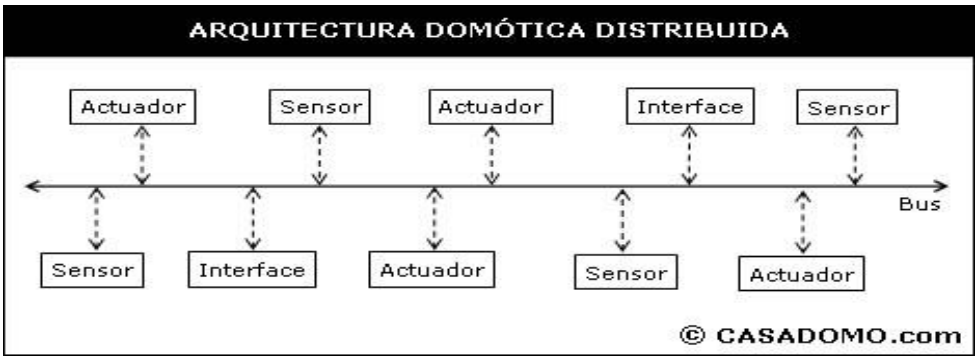

Figura 4: Esquema de Arquitectura Distribuida.

"Arquitectura Híbrida / Mixta – En un sistema de domótica de arquitectura híbrida (también denominado arquitectura mixta) se combinan las arquitecturas centralizadas, descentralizadas y distribuidas. A la vez que puede disponer de un controlador central o varios controladores descentralizados, los dispositivos de interfaces, sensores y actuadores pueden también ser controladores (como en un sistema "distribuido") y procesar la información según el programa, la configuración, la información que capta por sí mismo, y tanto actuar como enviarla a otros dispositivos de la red, sin que necesariamente pase por otro controlador."[9]

En la figura 5 se aprecia el funcionamiento de este tipo de arquitectura

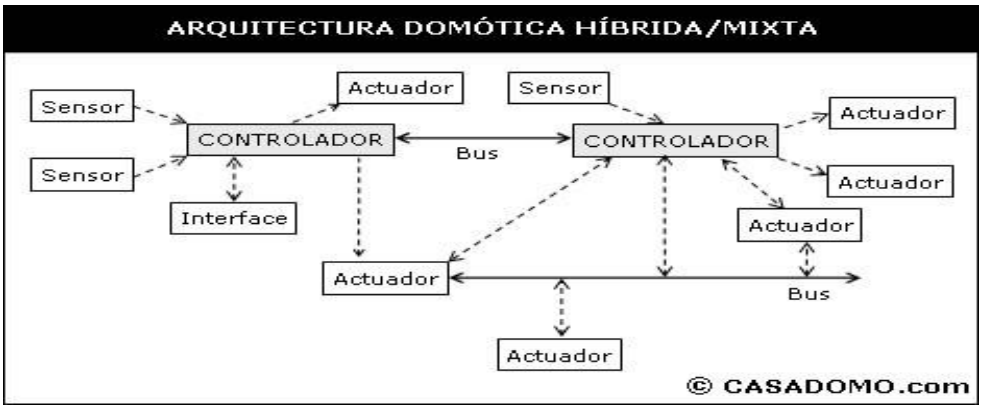

Figura 5: Esquema de Arquitectura Mixta

[8] Casadomo Soluciones SL, Servicios de una casa inteligente, 2006, http://www.casadomo.com

[9] Casadomo Soluciones SL, Servicios de una casa inteligente, 2006, http://www.casadomo.com

1.1.6 Algunas tecnologías usadas en la domótica

El estudio del desarrollo de la domótica ha estado matizado por múltiples aspectos, que incluyen áreas diversas entre las que se pueden mencionar: los precios, los dispositivos a conectar y las tecnologías a emplear. En este acápite se presenta una breve reseña de tres de las tecnologías más utilizadas en la domótica actual, que son: X-10, *Universal Plug&Play* (UPnP) y *Jini*, como referencia del estado del arte del tema y para propiciar el análisis de similitudes y diferencias entre estas tecnologías. El sistema que se propone en la tesis. **X-10** es una de las tecnologías más antiguas que se está usando en aplicaciones domóticas, la figura 6 muestra su logotipo.

Figura 6: Logotipo X10 que usan las compañías dedicadas a la venta de dispositivos.

Esta tecnología usa las líneas eléctricas de la vivienda, por lo que no es necesario tender nuevos cables para conectar dispositivos. Existen tres grandes familias de productos basadas en X-10, teóricamente compatibles entre sí, que son: *Netzbus, Timac y Home Systems*.

Los sistemas X-10 son sencillos y las actividades de instalación pueden ser realizadas por los usuarios finales o electricistas sin conocimientos de automatización, se puede afirmar que en estos momentos es la tecnología más accesible para realizar una instalación domótica no muy compleja.

Universal Plug&Play **(UPnP)** es una tecnología, que posee una arquitectura de software abierta y distribuida, en la figura 7 se representa su logotipo.

Figura 7: Logotipo del Universal Plug & Play arquitectura desarrollada por Microsoft.

UPnP permite que las aplicaciones de los dispositivos conectados a una red intercambien información y datos de forma sencilla y transparente para el usuario, sin necesidad de que éste tenga que ser un experto en la configuración de redes, dispositivos o sistemas operativos. Directamente se encarga de

todos los procesos necesarios para que un dispositivo o computadora conectada a una red pueda intercambiar información con el resto y ha sido diseñado de forma que sea independiente del fabricante, sistema operativo, del lenguaje de programación de cada dispositivo o computadora, y del medio físico usado para implementar la red.

Esta tecnología es capaz de descubrir un nuevo elemento (equipo o dispositivo) cuando se conecta a la red, le asigna una dirección IP, un nombre lógico, así como envía y recibe información sobre funciones y capacidad de procesamiento de él y de los demás respectivamente. Como se puede apreciar el usuario no tiene que preocuparse de configurar la red ni de perder el tiempo instalando drivers o controladores de dispositivos, sino que UPnP se encarga de todos estos procesos cuando se conecta o se desconecta un equipo y optimiza la configuración de los mismos. Como protocolo de comunicación UPnP utiliza SOAP (*Simple Object Access Protocol*), y para dispositivos no IP trabaja con SCP (*Simple Control Protocol*).

Jini es una tecnología desarrollada por *Sun Microsystems*, en la figura 8 se representa su logotipo.

Figura 8: Logo de Jini por parte de Sun Microsystems

Esta tecnología proporciona un mecanismo sencillo para que diversos dispositivos conectados a una red puedan colaborar y compartir recursos sin necesidad de que el usuario final tenga que planificar y configurar dicha red. En esta red de equipos, llamada "comunidad", cada uno proporciona a los demás los servicios, controladores e interfaces necesarios para distribuirse de forma óptima la carga de trabajo o las tareas que deben realizar.

“Al igual que el UPnP de Microsoft, el Jini tiene un procedimiento, llamado "*discovery*" para que cualquier dispositivo recién conectado a la red sea capaz de ofrecer sus recursos a los demás, informando de su capacidad de procesamiento y de memoria además de las funciones que es capaz de hacer (tostar el pan, sacar una foto digital, imprimir, etc.). Una vez ejecutado el *discovery,* se ejecutará el procedimiento "*join*", asignándole una dirección fija, una posición en la red; después empieza el servicio *Lookup* a proporcionar todos los servicios que se encuentran disponibles en la red.”[10]

[10] TEXEIRA LÓPEZ, M. Jini – Sistemas Distribuidos en Java. http://ciberia.ya.com/pxai/ma.html, 2002

La arquitectura de la tecnología Jini está totalmente distribuida, todos los dispositivos pueden comunicarse y ofrecer sus servicios a los demás, sin que sea necesaria la presencia de algún dispositivo que haga la función de controlador central de la red, es decir que sin contar con una computadora personal que controle los dispositivos conectados a la red, se pueden realizar todos los procesos en correspondencia con las demandas y exigencias del sistema. Además, Jini puede funcionar en entornos dinámicos donde la aparición o desconexión de dispositivos sea constante.

1.1.7 Ejemplos de software empleados en la domótica

El estudio bibliográfico realizado reveló que en la actualidad se emplean diversos programas informáticos en los sistemas domóticos, entre los que se encuentran: PROCESO: VISIR (Simulador de Instalaciones Domóticas), PROCESO: PROSIMAX (Simulador de Procesos), Active home, HomeSeer, Indigo y TERMVIS; de los cuales se presenta a continuación una breve caracterización.

PROCESO: VISIR (Simulador de Instalaciones Domóticas) y PROCESO: PROSIMAX (Simulador de Procesos).

Son aplicaciones para entorno *Windows* que permiten diseñar instalaciones domóticas y efectuar la simulación del comportamiento de las mismas en conexión directa con los autómatas programables de las series *SIMATIC* S5 y *SIMATIC* S7-200 de *SIEMENS*. Tienen dos módulos básicos, que son el de edición y el de simulación.

Módulo de Edición: Permite diseñar la instalación a simular mediante la selección de objetos dinámicos, organizados en grupos: detección de inundaciones, incendio, fugas de gas, seguridad, iluminación, toldos y persianas, control de cargas, etc. Se configuran tamaños y posición de objetos, modos de comportamiento, conexiones y representaciones gráficas, sin necesidad de programación. Adicionalmente se puede incorporar un dibujo de fondo.

Módulo de Simulación: Permite la conexión al autómata a través del cable serie y se pueden comprobar las reacciones de la instalación domótica controlada por el programa de control real en el PLC. El usuario puede intervenir de igual manera que lo haría en una instalación real. Tiene la posibilidad de comunicación con autómatas de las series Simatic S5 y Simatic S7-200.

Estas aplicaciones tienen ventajas entre las que se encuentran:

1. La posibilidad de realizar prácticas más reales.
2. Facilitan la determinación de errores de programación.
3. Poseen una gran flexibilidad, son económicas.
4. Constituyen un complemento de las rígidas y costosas maquetas.

5. Garantizan rapidez de operación.
6. Son de fácil aprendizaje

Como se muestra en las figuras del anexo 1, esta simulación puede ser fácilmente usada por cualquier persona de forma sencilla y rápida, permite la opción de que cada usuario coloque la cantidad de dispositivos que necesite y con la distribución que desee.

ACTIVE HOME.

Se trata de un software domótico creado para proyectos de domótica con tecnología X-10 controlados por una computadora y en los que se necesita la utilización de un módulo que se conecta al equipo mediante puerto serie o por usb. El software suele venir ya incluido con este dispositivo, y su utilización es sencilla apoyada en una interfaz gráfica, que se puede ver en el anexo 1. Los requerimientos técnicos de la computadora son elementales, si cuenta con un 386 y 4 Mb de RAM puede utilizarlo.

Este sistema puede ser instalado en pocos minutos por cualquier persona y permite que rápidamente se puedan estar controlando las luces y otros dispositivos del inmueble en la computadora, y llegar incluso a generar macros o rutinas programadas según horarios o días específicos. La comunicación con los distintos dispositivos es a través del cableado eléctrico existente. Luego de programar con el software los eventos que se deben desarrollar, se puede apagar la computadora y todo quedará almacenado en la memoria del Active Home.

HomeSeer

Este es otro programa pensado para controlar viviendas domóticas con **tecnología X-10**, pero a diferencia del anterior, sí tiene que estar encendida siempre la computadora. **HomeSeer** ha sido diseñado para controlar iluminación, aparatos, seguridad, climatización, y cine en casa, todo integrado en una única interfaz y utilizando un ordenador compatible con Windows. Al estar basado en un servidor web, es posible monitorear la vivienda remotamente a través de internet.

Indigo

Programa para proyectos con tecnología X-10, similar a ActiveHome, para programar cualquier dispositivo para ordenador, solo que·en este caso está pensado para el Sistema Operativo Macintosh de Apple. Este software necesita una versión 10.4 o superior de Mac OS X. Desde un teléfono móvil que se comunique con Indigo (*email, bluetooth*), se pueden ejecutar órdenes. Remotamente permite controlar decenas de aplicaciones, así como la programación de múltiples “escenas domóticas“. En el anexo 1 se muestra la interfaz gráfica.

TermVIS 2.1

Es un software que permite controlar el funcionamiento del sistema domótico Simón VIS. Posee una interfaz gráfica que propicia una exposición detallada de las diferentes opciones y de los apartados que contiene. Es un sistema amigable y de fácil operación, en el anexo 1 se muestra una de las pantallas del sistema, en particular la referida al control de toldos y persianas.

1.2 Vías de comunicación más utilizadas en la domótica

Los puertos de comunicación de las computadoras personales son de particular interés para el desarrollo de aplicaciones domóticas, ya que permiten su utilización para controlar los circuitos electrónicos utilizados, principalmente, en actividades de automatización de procesos, adquisición de datos, tareas repetitivas y otras actividades que demandan precisión.

Existen dos métodos básicos para transmisión de datos en las computadoras modernas. En un esquema de transmisión de datos en serie un dispositivo envía datos a otro a razón de un bit a la vez a través de un cable. Por otro lado, en un esquema de transmisión de datos en paralelo, un dispositivo envía datos a otro a una tasa de n número de bits a través de n número de cables a un tiempo. Un dispositivo serial utiliza un protocolo de comunicación que es estándar para casi cualquier computadora personal. La mayoría de las computadoras incluyen dos puertos seriales RS-232.

El puerto paralelo, también denominado puerto de impresora o LPT, dispone de tres canales de comunicaciones cuyos pines envían señales desde y hacia la computadora todos al mismo tiempo, de ahí el término paralelo. En el epígrafe siguiente se describen aspectos sobre su funcionamiento que fueron utilizados en la implementación de la capa de comunicación del sistema diseñado.

1.2.1 Funcionamiento del puerto paralelo

En la actualidad el puerto paralelo se encuentra comúnmente incluido en la placa madre de la computadora (*MotherBoard*) y en una típica computadora personal se utiliza un conector de 25 pines (DB-25 S), el orden de los pines del conector se muestra en la figura 9

El puerto paralelo está formado por 17 líneas de señales y 8 líneas de tierra. Las líneas de señales están formadas por tres grupos: 4 líneas de control, 5 líneas de estado y 8 líneas de datos.

Los terminales del puerto paralelo sólo pueden manejar señales digitales, cuyos valores de tensión representan estados altos o bajos. Cuando no hay tensión en el pin se asume un estado lógico bajo mientras que cuando hay una tensión cercana a los 5v el estado asumido es el alto.

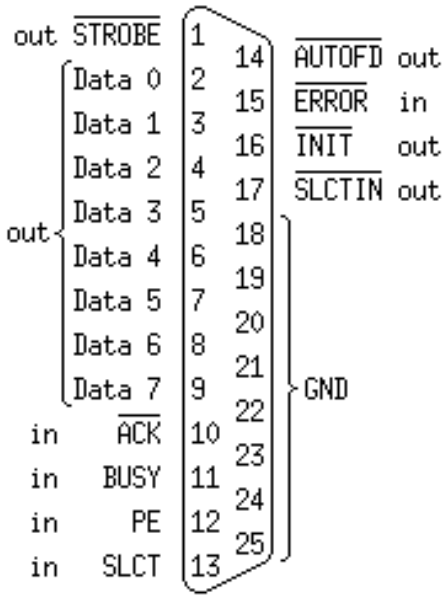

Figura 9: Distribución de pines del puerto paralelo.

La siguiente tabla describe la función de los pines de este conector.

Pines	E/S	Polaridad activa	Descripción
1	Salida	0	*Strobe*
2 ~ 9	Salida	-	Líneas de datos (bit 0/patita 2, bit 7/pin 9)
10	Entrada	0	Línea *acknowledge* (activa cuando el sistema remoto toma datos)
11	Entrada	0	Línea busy (si está activa, el sistema remoto no acepta datos)
12	Entrada	1	Línea Falta de papel (si está activa, falta papel en la impresora)
13	Entrada	1	Línea *Select* (si está activa, la impresora se ha seleccionado)
14	Salida	0	Línea *Autofeed* (si está activa, la impresora inserta una nueva línea por cada retorno de carro)
15	Entrada	0	Línea Error (si está activa, hay un error en la impresora)
16	Salida	0	Línea INIT (Si se mantiene activa por al menos 50 micro-segundos, ésta señal autoinicia la impresora)
17	Salida	0	Línea *Select* input (Cuando está inactiva, obliga a la impresora a salir de línea)
18 ~ 25	-	-	Tierra eléctrica

Tabla 3: Funciones de los pines del puerto paralelo.

Hay tres direcciones de E/S asociadas con el puerto paralelo, estas direcciones pertenecen al registro de datos, el registro de estado y el registro de control. El registro de datos es un puerto de lectura-escritura de ocho bits, leer el registro de datos (en la modalidad unidireccional) retorna el último valor escrito en el registro de datos. Los registros de control y estado proveen la interfaz a las otras líneas de E/S. Cada una de estas líneas (control, estado, datos) puede ser referenciada de modo independiente mediante un registro.

Puesto que la computadora es mucho más rápida que cualquier periférico con el que se comunique, puede fácilmente transmitir más datos que los que el periférico puede manejar. Por ello, los periféricos utilizan señales especiales para decirle a la computadora que detenga momentáneamente el envío de datos cuando tienen suficientes para trabajar. Esto le permite al periférico alcanzar a la computadora, que puede realizar otras tareas mientras tanto. Una vez que el periférico queda libre, le pide a la computadora que transmita más datos, y el proceso continúa.

Este juego computarizado de "luz roja, luz verde" se logra enviando señales por cables dedicados a ese propósito. El proceso de utilizar señales para controlar el flujo de datos se denomina diálogo (*handshaking*), de modo que las señales empleadas para ello se llaman "señales de diálogo".

1.3 Metodología para el desarrollo de software.

El proceso de desarrollo de software debe estar guiado por una metodología y en la actualidad existen varias entre las que se encuentran XP y Proceso Unificado de Software (RUP del inglés *Rational Unified Process*) que es una de las más usadas. RUP es una metodología de desarrollo de software orientado a objeto, cuya utilización permite la construcción de sistemas más grandes, más completos y de más calidad. Hace uso del UML como lenguaje de modelado.

Como se puede apreciar en la figura 10 en RUP se han agrupado las actividades en 9 flujos de trabajo principales, los 6 primeros son conocidos como flujos de ingeniería (Modelado del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba, Instalación) y los tres últimos de apoyo (Administración del proyecto, Administración de configuración y cambios, Gestión de entorno). Además un proyecto realizado siguiendo RUP se divide en cuatro fases: Inicio (puesta en marcha), Elaboración (definición, análisis, diseño), Construcción (implementación) y Transición (fin del proyecto y puesta en producción). En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto).

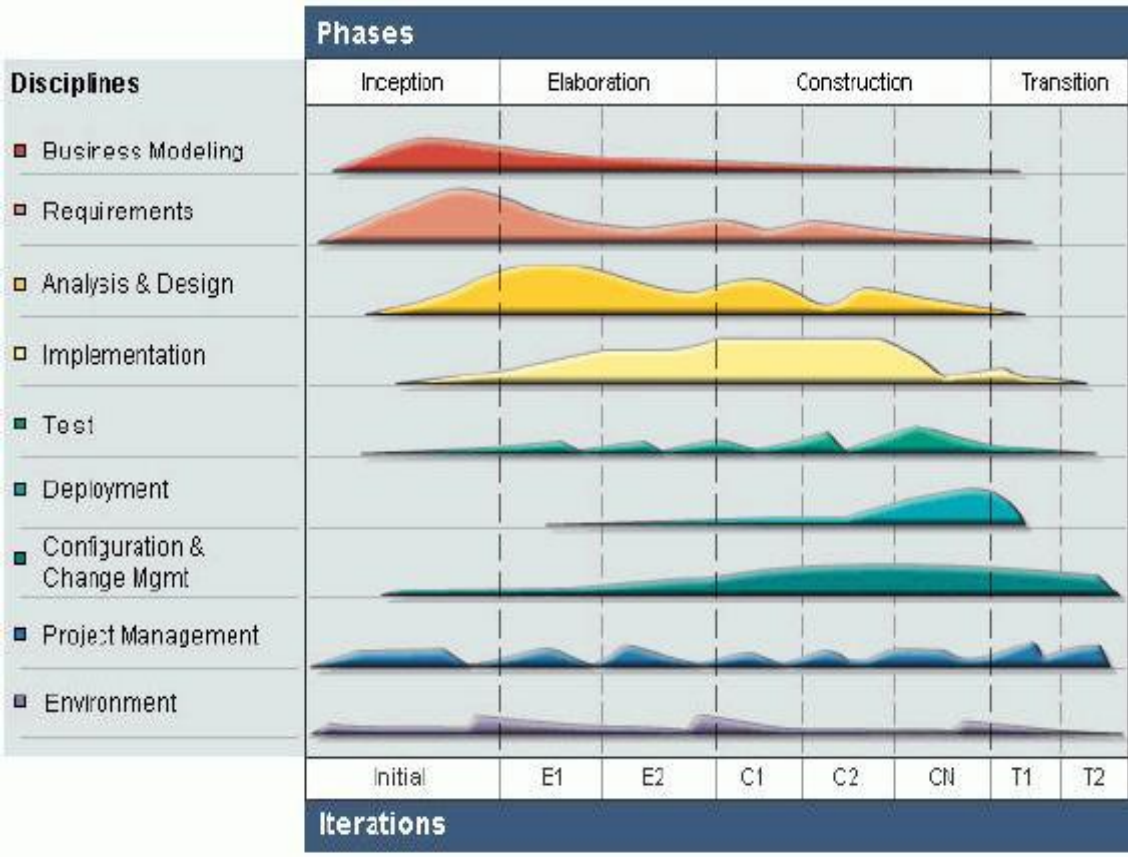

Figura 10: Diagrama de RUP en dos dimensiones.

Características de RUP:

- Está dirigido por casos de uso, avanza a través de los flujos de trabajo que parten de los casos de uso y estos son el instrumento para validar la arquitectura del software y extraer los casos de prueba.
- Está centrado en la arquitectura, los modelos son proyecciones del análisis y el diseño, constituye la arquitectura del producto a desarrollar.
- Es iterativo e incremental, durante todo el proceso de desarrollo se obtienen versiones incrementales que se acercan al producto terminado.

A modo de resumen resulta oportuno referir que la selección de RUP como metodología de desarrollo estuvo motivada por sus potencialidades, expresadas básicamente en aspectos como: unifica los mejores elementos de las restantes metodologías, está preparada para desarrollar grandes y complejos proyectos, utiliza el UML como lenguaje de representación visual y por las características antes mencionadas.

1.4 Lenguajes

De los múltiples lenguajes de modelado y desarrollo que existen en el contexto actual de la informática, fueron seleccionados para la elaboración del sistema, el Lenguaje Unificado de Modelado (UML) y JAVA, respectivamente. A continuación se presentan las características básicas de cada uno y los motivos por los que fueron elegidos.

1.4.1 Lenguaje Unificado de Modelado (UML)

Lenguaje Unificado de Modelado (UML del inglés *Unified Modeling Language*). Es ante todo un lenguaje, que proporciona símbolos y reglas que de forma sencilla expresan los elementos básicos para la construcción de un sistema, se centra en su representación gráfica y nos indica cómo crear y leer los modelos, sin embargo no dice cómo hacer el sistema. El UML es un lenguaje para visualizar, especificar, construir y documentar los elementos que componen un sistema basado en la programación orientada a objetos, ya que es el resultado de la unión de las mejores cualidades de los tres lenguajes existentes que le dieron paso por el trabajo en conjunto de sus autores.

A partir del surgimiento de UML, muchas de las metodologías existentes han sido adaptadas para utilizar este lenguaje, como es el caso de la Metodología de Análisis y Diseño Orientado a Objetos de Sistemas Informáticos en su versión 5.0 y en otras como el Proceso Unificado de Desarrollo se concibió desde sus inicios utilizar UML. Este lenguaje es lo suficientemente flexible para manejar diferentes conceptos inherentes al desarrollo de software moderno, entre los que se pueden mencionar: distribución física, concurrencia, réplicas, seguridad y carga balanceada. Es capaz de modelar toda la gama de sistemas que se necesite construir.

1.4.2 JAVA

En este acápite se pretende dar respuesta a la pregunta ¿Por qué JAVA para el desarrollo de la aplicación?, y para ello resulta oportuno comenzar recordando lo expresado en el libro blanco de Java al referir que los objetivos de su diseño eran ser "un lenguaje sencillo, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portátil, de gran rendimiento, multitarea y dinámico". [11] A continuación se presentarán las ideas básicas que explican cada una de las mencionadas características de Java, que fundamentan su elección.

[11] ZUKOWSKI, J. Libro blanco de Java. http://java.sun.com/people/jag/OriginalJavaWhitepaper.pdf. 2001; p. 29.

- ✓ **Sencillo**: Porque incorpora nuevas tareas como un recolector de elementos no utilizados automático y elimina aspectos de C++ confusos y muy poco utilizados como la sobrecarga de operadores. Otro aspecto de la simplicidad de Java es que nada es realmente nuevo, es decir, su conjunto de funciones procede de algún otro lenguaje.
- ✓ **Orientado a objetos**: "Esencialmente, la programación orientada a objetos (OOP) es un modo de desarrollar software describiendo problemas mediante el uso de elementos u objetos desde el espacio del problema y no mediante un conjunto de pasos secuenciales que se ejecutarán en el ordenador. Un buen diseño conlleva a componentes reutilizables, extensibles y sostenibles."[12]. Java cuenta con bibliotecas de clases previamente diseñadas que se van enriqueciendo de una a otra versión y constituyen la base sobre la cual el programador trabaja.
- ✓ **Distribuido**: Este lenguaje reconoce la red y admite el acceso a objetos distribuidos tan fácilmente como a los objetos locales. Mediante protocolos comunes y los servicios Web, se invocan métodos en un equipo remoto tan fácil e invisiblemente como puede hacerlo en su mismo espacio de ejecución.
- ✓ **Interpretado**: "Los programas de Java son interpretados. En lugar de ser compilado en ejecutables nativos, el código de Java es traducido en códigos de bytes no asociados a una plataforma. Estos códigos de bytes se pueden transferir a cualquier plataforma que tenga *Java Runtime Environment* (JRE), que consiste en una Máquina Virtual de Java (JVM) y de este modo pueden ejecutarse sin volver a compilarlos o revincularlos."[13]
- ✓ **Robusto:** La robustez es la medida de la fiabilidad de un programa y Java contiene varias funciones integradas que la garantizan, por ejemplo es un lenguaje basado en tipos, no tiene punteros, realiza automáticamente la recolección de elementos no utilizados y fomenta el uso de interfaces.
- ✓ **Seguro:** La seguridad está ligada a la robustez, por ejemplo, al no tener punteros no se corrompe la memoria. Además tiene integradas otras funciones de seguridad como son: el verificador de código de bytes que proporciona el primer nivel de defensas, a continuación el cargador de clases y luego el administrador de seguridad, que refuerza las normas de seguridad para el entorno de ejecución.
- ✓ **De arquitectura neutral:** Los programas de Java se realizan para que se ejecuten en cualquier equipo sin recompilarlos o revincularlos.

[12] ZUKOWSKI, J. Java2 J2SE 1.4. Anaya Multimedia S. A.,2003. p. 38

[13] ZUKOWSKI, J. Java2 J2SE 1.4. Anaya Multimedia S. A.,2003, p. 38

- ✓ **Portátil e independiente de la plataforma:** Significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, *write once, run everywhere*. La portabilidad viene dada por la capacidad que tiene Java para ejecutar un mismo programa en cualquier arquitectura. Por otra parte el espacio que ocupan los programas es relativamente pequeño.
- ✓ **De gran rendimiento:** Los códigos de bytes de plataforma neutral realmente pueden convertirse en tiempo de ejecución en código maquina específico de la CPU, ejecutándose rápidamente. Hay dos herramientas de traducción incluidas con Java que lo hacen automáticamente: el compilador Justo a tiempo (JIT) y *HotSpot*.
- ✓ **Multitarea:** Un programa que ejecuta varias tareas o subprocesos simultáneamente, se dice que es multitarea, en Java se incluyen múltiples recursos con el objetivo de facilitar la comunicación garantizando la seguridad de los subprocesos.
- ✓ **Dinámico:** Las bibliotecas de Java se encuentran evolucionando constantemente, sin embargo eso no implica que los programas anteriores dejen de funcionar. Otro elemento que explica el dinamismo es la preferencia de Java de las interfaces sobre las clases.

Las características del leguaje anteriormente explicadas, unidas a que la plataforma Java forma parte de la familia de software libre; fueron las causas que motivaron la selección de este lenguaje para el desarrollo de la aplicación.

1.5 Herramientas

Las herramientas utilizadas en la elaboración del sistema fueron *Rational Rose y NetBeans*, las cuales serán brevemente caracterizadas en los subepígrafes siguientes.

1.5.1 Rational Rose

Rational Rose es la herramienta de Ingeniería de Software Asistida por Ordenador (CASE, en inglés *Computer Aided Software Engineering*) desarrollada por los creadores de UML, con el objetivo de realizar la modelación gráfica de sistemas. Es considerada una de las herramientas más usadas en el mundo para la modelación visual de los procesos de: modelado del negocio, análisis de requerimientos y diseño de la arquitectura de componentes.

Rational Rose es una herramienta con plataforma independiente que facilita el trabajo en equipos, ya que ayuda a la comunicación entre sus miembros, propicia también monitorear el tiempo de desarrollo y la comprensión del entorno de los sistemas. Al utilizar el UML como lenguaje, permite a los

arquitectos y desarrolladores de software visualizar el sistema completo utilizando un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción. Por otra parte cada integrante del equipo puede modelar sus componentes e interfaces de forma individual y luego unirlos con otros componentes del proyecto, gracias a que cada cual tiene sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue).

Las mencionadas características de esta herramienta, unidas a que la misma está basada en la metodología de desarrollo de software RUP; fueron elementos determinantes en la decisión de utilizarla como herramienta de modelación visual en el desarrollo del software.

Finalmente es oportuno referir que *Rational Rose* cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

1.5.2 NetBeans

NetBeans IDE es una herramienta de desarrollo Java, escrita puramente sobre la base de la tecnología Java, de modo que puede ejecutarse en cualquier ambiente que ejecute Java. Es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, lo cual ha permitido una mayor capacidad de uso y ha proporcionado a los desarrolladores mayor flexibilidad y numerosas ventajas en la creación de aplicaciones multiplataforma.

En una era en la cual la arquitectura orientada al servicio (SOA) requiere servicios con cierta relación que manejen procesos específicos del negocio, *NetBeans* satisface los requisitos conjuntos de herramientas independientes de la plataforma, modulares y orientadas al objeto.

Al integrar múltiples herramientas y protocolos facilita la migración, sus amplias posibilidades de desarrollo multiplataforma atraen a muchos desarrolladores que han trabajado con otras herramientas y que comprenden que con *Netbeans* se pueden acelerar sus esfuerzos de desarrollo.

La facilidad de uso de *NetBeans* durante todo el ciclo de desarrollo, resulta llamativa gracias a su interfaz de usuario y a sus múltiples funciones y componentes. El acceso inmediato a los perfiles hace posible que los equipos de desarrollo sepan cuán bien trabajará un software para los usuarios finales. De igual forma, NetBeans prueba el propio ciclo de desarrollo y este proceso ayuda a ampliar el alcance de las aplicaciones creadas con el IDE.

NetBeans maneja la complejidad de la arquitectura orientada al servicio (SOA), en las que los desarrolladores trabajan generalmente con múltiples tecnologías y protocolos, lo cual ocasiona por lo general un gasto de análisis y depuración, que pueden retrasar la producción, cuando se realizan

manualmente, sin embargo *NetBeans* reduce el tiempo que se invierte trabajando con estos protocolos, generando automáticamente el código asociado.

De igual manera, *NetBeans* ayuda a los equipos de desarrollo a utilizar otras mejores prácticas y estándares de la industria para la productividad general del grupo. Un conjunto de estas mejores prácticas incluye soluciones a problemas comunes de configuración que se encuentran en la guía pública de Patrones de Diseño. Ésta incluye patrones de diseño estándares de la industria ampliamente aceptados para *Enterprise Java Beans* y otros patrones, cada uno de los cuales puede ser reutilizado por todo el equipo de desarrollo, a fin de incrementar la eficiencia de la organización.

La fácil generación de documentos de estos proyectos de modelado hace también que el desarrollo sea más eficiente. Se pueden generar informes en HTML o en Map y los informes de uso sencillo incluyen diagramas y pueden colocarse fácilmente en intranets o pueden imprimirse para su revisión.

En general se puede afirmar que es una herramienta creada para acelerar el desarrollo, ya que su flexibilidad entre plataformas, el cumplimiento de UML y la capacidad de administrar la complejidad, ayudan a garantizar que las aplicaciones cumplan con los requerimientos específicos del negocio. Además los desarrolladores pueden confiar en una plataforma de desarrollo que integra todas las piezas críticas, necesarias en cada nivel de creación de aplicaciones.

Conclusiones del capítulo

El concepto domótica se refiere a la automatización y control (encendido / apagado, apertura / cierre y regulación) de aparatos y sistemas de instalaciones eléctricas y electrotécnicas (iluminación, climatización, persianas y toldos, puertas y ventanas motorizados, etc.) de forma centralizada y/o remota. El objetivo del uso de la domótica es generar servicios para el aumento del confort, la gestión energética y la mejora de la seguridad; estos se conocen como Pilares de la domótica. En la actualidad existen diversas tecnologías para las aplicaciones domóticas y se emplean múltiples programas informáticos en los sistemas domóticos.

Atendiendo a sus potencialidades se seleccionó como metodología para el desarrollo del sistema domótico, RUP y como herramienta de modelado *Rational Rose*. Además se decidió utilizar *NetBeans* como herramienta de desarrollo Java dada su facilidad de uso, flexibilidad y su posibilidad de integración con múltiples plataformas y lenguajes. Como lenguaje de modelado fue utilizado el UML y como lenguaje de programación Java, que se caracteriza por ser seguro, orientado a objetos, distribuido, interpretado, robusto, sencillo, de arquitectura neutral, portátil e independiente de la plataforma que significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware.

Capítulo 2 Características del sistema

Este capitulo está dedicado a conocer lo referente al objeto que se quiere informatizar, se presenta el modelo de dominio, como alternativa al modelo de negocio, para entender el contexto en que se ubica el sistema. Se muestran los requerimientos funcionales y no funcionales especificados por el usuario y se describen los casos de uso y la propuesta del sistema a desarrollar.

2.1 Problema y situación problémica

Son diversos los procesos domésticos que pueden automatizarse en aras de garantizar una adecuada gestión energética, mayor confort y mecanismos de seguridad personal y patrimonial.

El actual desarrollo tecnológico y cultural que se ha producido a escala mundial ha impulsado el surgimiento y desarrollo de la domótica, que ha venido a dar respuesta a las necesidades de control y automatización de las viviendas modernas, para lo cual se comercializan diversas tecnologías y sistemas muy intuitivos, los que unidos a las posibilidades que dan la conexión a Internet, con redes de banda ancha, o la conexión a través de redes móviles GSM o de otro tipo, para el control remoto y la vigilancia, hacen que se extienda muchísimo el campo de aplicación de la domótica.

La aspiración a una mejor calidad de vida tiene un carácter universal, sin embargo la realidad descrita en el párrafo anterior está fuera del alcance de las mayorías en los países pobres y en particular de Cuba, que está sometida a un férreo bloqueo económico y cuyas posibilidades de inversión están dirigidas a otros sectores prioritarios. Es por ello que el autor se motivó a trabajar en el diseño e implementación de un sistema informático para la domótica, que satisfaga las necesidades básicas de control y automatización de la vivienda cubana de hoy, en la cual el nivel de automatización que existe es casi nulo.

Partiendo de esta situación problémica se identificó como problema científico ¿Cómo controlar dispositivos y procesos domésticos de forma automatizada con el uso de la informática?

2.2 Objeto de automatización

Los procesos objeto de automatización son los relacionados con el control doméstico, entre ellos se encuentran los siguientes:

- ✓ Apagado y encendido de dispositivos: Proceso mediante el cual se apagan o encienden los diferentes equipos electrodomésticos disponibles, que pueden estar relacionados con iluminación, climatización o cualquier otra esfera.

- ✓ Control de instalaciones hidráulicas y llenado de tanques: Proceso consistente en el control de válvulas, bombas y demás componentes de la instalación hidráulica.
- ✓ Gestión de seguridad personal y patrimonial: Proceso encargado de gestionar la seguridad de las personas y sus bienes, frente a eventualidades como la presencia de intrusos, el escape de gas, la ocurrencia de incendios, tormentas u otros fenómenos climatológicos.
- ✓ Apertura y cierre de puertas, toldos y ventanas: Proceso mediante el cual se abren o cierran puertas, toldos y ventanas, para controlar el acceso, la iluminación y la ventilación.

En los últimos tiempos dichos procesos han sido objeto de la domótica y han surgido a escala mundial múltiples sistemas y tecnologías que permiten su automatización.

2.3 Información que se maneja

- ✓ Reportes de la activación de actuadores: Se cuenta con un historial de aquellos dispositivos actuadores que se han activado o desactivado y el motivo por el cual cambió de estado.
- ✓ Reportes del comportamiento de los sensores: El sistema le brinda al usuario la posibilidad de saber qué dispositivos han cambiado su estado, la hora en que lo hicieron y el tiempo que duró cada evento ocurrido durante un período determinado.
- ✓ Reportes de alarmas activadas: La aplicación brinda la posibilidad de obtener un reporte de todas las alarmas que se dispararon durante un periodo de tiempo dado.
- ✓ Estado de todos los dispositivos sensores y actuadores: El sistema conoce y ofrece al usuario el estado en el que se encuentran todos los dispositivos conectados.
- ✓ Reglas existentes: La aplicación da la posibilidad al usuario de conocer todas las reglas existentes para el control de los dispositivos actuadores y sensores disponibles.

2.4 Propuesta de sistema

Para dar solución al problema científico se creó un sistema informático, que mediante la conexión con el hardware necesario, permite controlar dispositivos y procesos domésticos de forma automatizada. El mismo, mediante interfaces de usuario sencillas posibilita interactuar con los diferentes dispositivos del hogar que se acoplen al sistema y para la automatización de los dispositivos y procesos se usan reglas que pueden ser creadas y/o modificadas por los usuarios de un modo elemental.

La aplicación está compuesta por tres módulos, a continuación se explica cada uno de estos.

- ✓ **Módulo de comunicación:** es el encargado de la escritura y la lectura de datos en los controladores, es decir, es el componente de software que sirve de intermediario entre el sistema

informático y el resto del sistema domótico. Este módulo garantiza total independencia entre el sistema informático y las características y funcionamiento del hardware utilizado.

- ✓ **Módulo Pizarra:** se encarga de gestionar los dispositivos del sistema domótico, es decir adicionar, modificar y/o eliminar controladores, actuadores y sensores; con lo cual se configura y personaliza el software atendiendo a las características y condiciones del inmueble donde se utilice. Además permite interactuar con los dispositivos (subir, bajar, apagar, encender, abrir, cerrar).
- ✓ **Módulo de Automatización:** permite gestionar, de un modo sencillo, reglas que garantizan la automatización de los dispositivos actuadores y sensores que se encuentren acoplados al sistema domótico, por ejemplo el apagado y encendido de luminarias y equipos electrodomésticos en determinados horarios o intervalos de tiempo.

2.5 Modelo de dominio

El sistema domótico desarrollado informatiza procesos domésticos que no son palpables en toda su extensión, es decir, no son del todo visibles y sus fronteras no están plenamente establecidas. Esto implica que no exista un negocio bien definido ni claridad en sus procesos; por lo que se realizó un modelado del dominio y no del negocio. El objetivo del modelado del dominio es capturar, comprender y describir las clases más importantes dentro del contexto del sistema domótico y junto al Diccionario de Clases del Dominio, ayudar a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común. En la figura 11 se muestra el Diagrama de Clases del Dominio.

Por su parte, el Diccionario de Clases del Dominio describe textualmente las clases identificadas durante el modelado del dominio del problema. Este diccionario sirve como un glosario de términos y se muestra a continuación:

- ✓ ***Usuario:*** Persona que recibe todas las funcionalidades del producto, interactúa con la aplicación, la configura o está a cargo de su supervisión.
- ✓ ***Pizarra:*** Componente de software que permite al usuario gestionar los dispositivos en el sistema, para configurar el software de acuerdo con las características y condiciones propias del inmueble donde se utilice. Este componente permite, de forma manual, interactuar con los dispositivos actuadores, así como conocer el estado de los dispositivos sensores.
- ✓ ***Autómata:*** Es la parte del sistema informático que permite al usuario gestionar reglas para automatizar el funcionamiento de dispositivos actuadores y sensores que se encuentren acoplados al sistema. Las reglas de automatización son postulados en los cuales, partiendo de la ocurrencia de una o más condiciones, se ejecuta un grupo de acciones, por ejemplo el apagado y encendido de luminarias y equipos electrodomésticos en determinados horarios o

intervalos de tiempo, la apertura o cierre de puertas, toldos y ventanas dado algún fenómeno climatológico, entre otros.

- ✓ ***Comunicación***: Es el componente de software encargado de la escritura y la lectura de datos en los controladores, es decir, es el componente de software que sirve de intermediario entre el sistema informático y el resto del sistema domótico. Este módulo garantiza total independencia entre el sistema informático y las características y funcionamiento del hardware utilizado.
- ✓ ***Controlador:*** Hardware creado específicamente para el control de dispositivos actuadores y sensores.
- ✓ ***Actuadores:*** son los dispositivos capaces de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema (apagar/encender, subir/bajar, apertura/cierre, etc.).
- ✓ ***Sensores***: Los sensores son los dispositivos que monitorean el entorno captando información y transmitiéndola al sistema, por ejemplo sensores de movimiento, agua, gas, humo, temperatura, viento, humedad, lluvia, iluminación, etc.

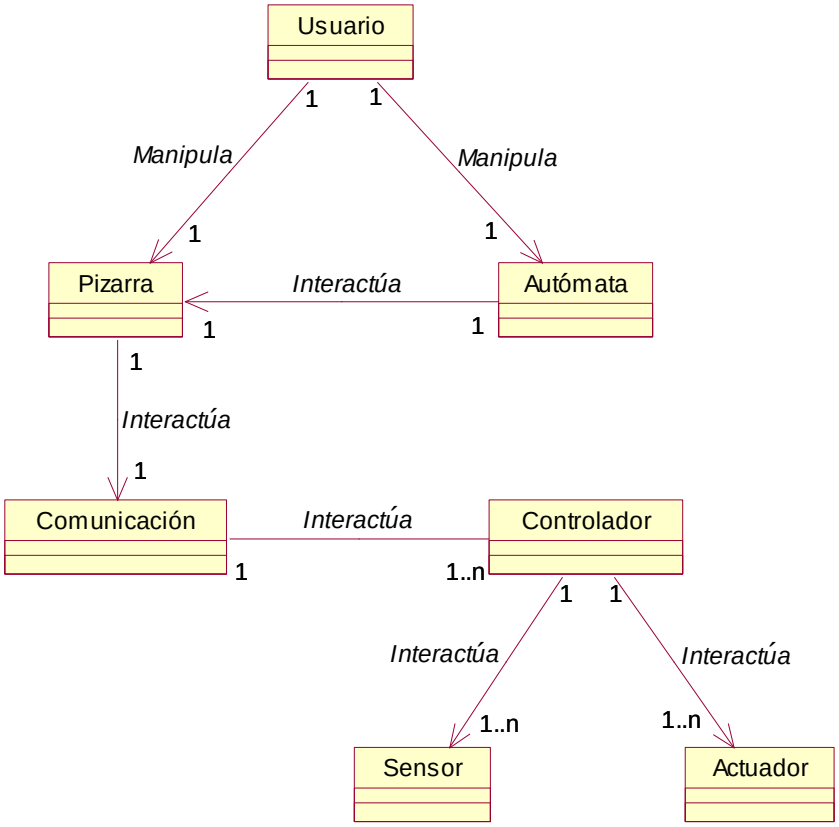

Figura 11: Modelo de dominio

2.6 Especificación de los requisitos de software

Los requerimientos funcionales especifican las acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física, es decir, especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de su existencia; es por ello que para su desarrollo fueron identificados los requerimientos funcionales que a continuación se presentan:

RF 1 Gestionar dispositivos.
- RF 1.1 Gestionar controladores.
  - RF 1.1.1 Adicionar controlador.
  - RF 1.1.2 Modificar controlador.
  - RF 1.1.3 Eliminar controlador
- RF 1.2 Gestionar actuadores.
  - RF 1.2.1 Adicionar actuador.
  - RF 1.2.2 Modificar actuador.
  - RF 1.2.3 Eliminar actuador.
- RF 1.3 Gestionar sensores.
  - RF 1.3.1 Adicionar sensor.
  - RF 1.3.2 Modificar sensor.
  - RF 1.3.3 Eliminar sensor.

RF 2 Controlar conexión lógica y física de controladores
- RF 2.1 Conectar controlador lógicamente.
- RF 2.2 Desconectar controlador lógicamente.
- RF 2.3 Verificar conexión física de controlador.

RF 3 Controlar conexión lógica de actuadores.
- RF 3.1 Conectar actuador lógicamente.
- RF 3.2 Desconectar actuador lógicamente.

RF 4 Controlar conexión lógica de sensores.
- RF 4.1 Conectar sensor lógicamente.
- RF 4.2 Desconectar sensor lógicamente.

RF 5 Controlar actuadores.
- RF 5.1 Encender actuador.
- RF 5.2 Apagar actuador.

RF 6 Obtener estados de sensores.

RF 7 Gestionar reglas de automatización.

RF 7.1 Adicionar reglas de automatización.

RF 7.2 Modificar reglas de automatización.

RF 7.3 Eliminar reglas de automatización.

RF 8 Gestionar alarmas.

RF 8.1 Adicionar alarmas mediante reglas de automatización.

RF 8.2 Modificar alarmas mediante reglas de automatización.

RF 8.3 Eliminar alarmas mediante reglas de automatización.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, y partiendo de las acciones que el mismo debe ser capaz de realizar se definieron los siguientes:

Apariencia o interfaz externa

- ✓ Las ventanas del sistema contendrán los datos claros y bien estructurados, y al mismo tiempo permitirán la interpretación correcta e inequívoca de la información.
- ✓ Se utilizará un criterio que permita la identificación visual de los elementos en la interfaz a través del uso de colores y formatos de fuente para la letra, entre otras técnicas.
- ✓ Mostrar mensajes de errores en la introducción de datos de una forma sencilla y explicativa, la entrada incorrecta de datos será detectada claramente por el sistema.
- ✓ Diseñar su funcionamiento de modo que sea intuitivo, y requiera de información mínima.
- ✓ Mostrar todos los textos y mensajes en español.

Rendimiento:

- ✓ Como herramienta de tiempo real, debe tener alto grado de velocidad de procesamiento o cálculo, tiempo de respuesta y de recuperación, y disponibilidad.
- ✓ Debe completar las transacciones en un tiempo menor a 0,5 segundos.
- ✓ La latencia del sistema no debe ser mayor de 0,2 segundos.

Portabilidad:

- ✓ El software está desarrollado en Java, con código totalmente portable; y cada uno de los drivers de dispositivos externos que utiliza la aplicación deben estar disponibles para Linux y para Windows.

Seguridad:

- ✓ Existencia de distintos roles que establezcan que la información sólo sea vista por aquellos usuarios que posean los privilegios suficientes.
- ✓ Restringir la ejecución de acciones por usuarios sin privilegios a acceder a las mismas.
- ✓ Realizar la verificación sobre acciones irreversibles como eliminaciones.

Software:

- ✓ Se utilizarán los sistemas operativos Linux y Windows indistintamente.
- ✓ Se necesitan tener otros programas instalados como la plataforma de Java JDK 6.0 o superior
- ✓ Disponer de los drivers necesarios para cada controlador.

Hardware:

- ✓ Para los requisitos de hardware de la PC se necesita ver los requerimientos mínimos de la maquina virtual de java (JDK) que se vaya a instalar.
- ✓ El sistema se comunica con los dispositivos controladores a través del puerto paralelo.
- ✓ Disponer de controladores para el control de dispositivos actuadores y sensores.

Ayuda y documentación

- ✓ La ayuda del sistema contiene datos claros y bien estructurados que permiten la interpretación correcta e inequívoca de la información
- ✓ Entregar documentos técnicos y las guías de usuario, que incluyen presentaciones realizadas en cada tema.
- ✓ Entregar carpeta del proyecto, con la documentación técnica generada en el desarrollo para la especificación del sistema.

2.7 Definición de los casos de uso

Definición de los actores.

En esta etapa fueron definidos dos actores de sistema que se presentan y justifican en la tabla siguiente:

Actores	Justificación
Administrador	Es el encargado de configurar el sistema inicialmente y cada vez que se instale un dispositivo.
Usuario	Es el que interactúa con el sistema y recibe las potencialidades y funcionalidades del mismo; entre las que se destaca la gestión de reglas que garantizan la automatización de los dispositivos actuadores y sensores que se encuentren acoplados al sistema domótico, por ejemplo el apagado y encendido de luminarias y equipos electrodomésticos en determinados horarios.

Tabla 4:Definición de actores

Listado de casos de uso.

Luego de definidos los dos actores antes mencionados se dio paso al análisis y fundamentación de los casos de uso que describen el funcionamiento del sistema, que son los siguientes:

CU-1	Gestionar Dispositivos
Actor	Administrador
Descripción	En este caso de uso se gestionan los dispositivos controladores, actuadores y sensores que estén conectados en el sistema. Es inicializado por el administrador del sistema y maneja lo referente a la configuración de cada dispositivo lo que permite poder adaptar la aplicación a cualquier entorno de trabajo. Cuenta con tres interfaces fundamentales, la primera de estas para gestionar los controladores con sus características, la segunda para gestionar los actuadores y la tercera para los sensores.
Referencia	RF1 (RF1.1, RF1.2, RF1.3)

Tabla 5: Caso de uso “Gestionar Dispositivos.”

CU-2	Controlar Conexión Lógica y/o Física de Dispositivos.
Actor	Administrador
Descripción	Este caso de uso es el responsable de mantener el control lógico y/o físico de los dispositivos. El sistema mantendrá informado al administrador si cada controlador (hardware) está conectado o desconectado de la red (físicamente), además permitirá que los controladores, actuadores y sensores pueda conectarse o desconectarse por software (lógicamente).
Referencia	RF2 (RF2.1, RF2.2, RF2.3); RF3 (RF3.1, RF3.2); RF4 (RF4.1, RF4.2)

Tabla 6: Caso de uso “Controlar Conexión Lógica y/o Física de Dispositivos”.

CU-3	Controlar Actuadores
Actor	Administrador, Usuario
Descripción	El sistema permite cambiar el estado de cada uno de los actuadores conectados, es decir, que se podrá encender y apagar cada uno de los dispositivos ya sean electrodomésticos u otros que se encuentren acoplados y configurados en el sistema de forma sencilla. El software cuenta con interfaces de gran maniobrabilidad que de manera gráfica muestren el estado de cada uno de los actuadores y permitan cambiarlo.
Referencia	RF5 (RF5.1, RF5.2)

Tabla 7: Caso de uso “Controlar Actuadores”.

CU-4	Obtener Estados Sensores
Actor	Administrador, Usuario
Descripción	Con este caso de uso el software permite conocer el estado de cada uno de los sensores conectados al sistema domótico y mostrarlo al usuario. Dicha información se recoge en la interfaz principal del software donde se encuentran representados los diferentes dispositivos del sistema.
Referencia	RF6

Tabla 8: Caso de uso “Obtener Estados Sensores”.

CU-5	Gestionar Reglas
Actor	Administrador, Usuario
Descripción	Este caso de uso permite gestionar reglas para la automatización de los dispositivos y procesos del entorno, por ejemplo el apagado y encendido de luminarias y equipos electrodomésticos en determinados horarios o intervalos de tiempo. Cuenta con una interfaz amigable que posibilita adicionar, eliminar y modificar reglas de manera fácil y sencilla.
Referencia	RF7 (RF7.1, RF7.2, RF7.3)

Tabla 9: Caso de uso “Gestionar Reglas”.

CU-6	Gestionar Alarmas
Actor	Administrador, Usuario
Descripción	El software tiene la potencialidad de construir distintos tipos de alarmas usando reglas de automatización anteriormente creadas o nuevas. Brinda mediante reportes y mensajes de alerta la funcionalidad de conocer el estado de diferentes alarmas en el momento en que estas se disparan y permite llevar un registro de los incidentes ocurridos.
Referencia	RF8 (RF8.1, RF8.2, RF8.3)

Tabla 10: Caso de uso “Gestionar Alarmas”.

Diagrama de casos de uso.

En el diagrama de casos de uso del sistema que se muestra en la figura 12, se representan gráficamente los procesos y su interacción con los actores.

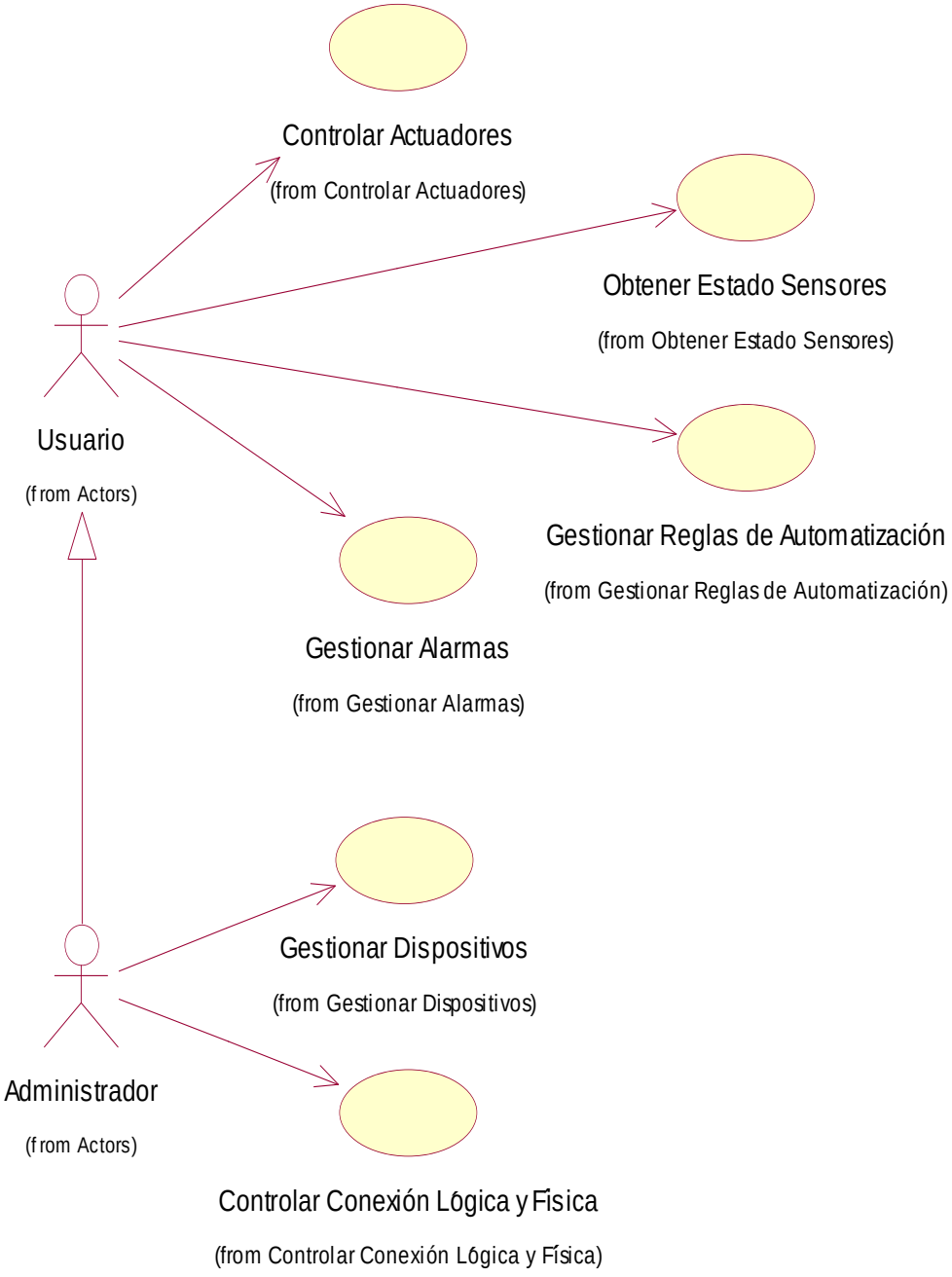

Figura 12: Diagrama de casos de uso.

Casos de uso por ciclo.

En el primer ciclo de desarrollo se lleva a cabo la realización de los casos de uso siguientes:

Código	Nombre de caso de uso	Paquete
CU-1	Gestionar Dispositivos	Pizarra
CU-2	Controlar Conexión Lógica y/o Física de Dispositivos	Pizarra
CU-3	Controlar Actuadores	Pizarra
CU-4	Obtener Estados Sensores	Pizarra

Tabla 11: Casos de uso primer ciclo.

Después de haber desarrollado los casos de uso previstos para el primer ciclo, se pasó a la realización de los siguientes casos de uso en un segundo ciclo.

Código	Nombre de caso de uso	Paquete
CU-5	Gestionar Reglas	Autómata
CU-6	Gestionar Alarmas	Autómata

Tabla 12: Casos de uso segundo ciclo.

Casos de uso expandidos.

Caso de uso	CU-1	Gestionar Dispositivos
Actores	Administrador	
Resumen:		
El caso de uso inicia cuando el actor selecciona alguna de las opciones relacionadas con gestionar dispositivos (Controlador, Actuador y Sensor); a partir de lo cual la aplicación muestra los controles necesarios para el registro de la información que se desea gestionar del dispositivo seleccionado. El software queda configurado con dichos dispositivos finalizando así el caso de uso.		
Precondiciones	El actor debe estar autenticado.	
Referencias	RF1 (RF1.1, RF1.2, RF1.3)	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. El actor selecciona gestionar dispositivos		1.1 El sistema da las opciones de gestionar Controlador Actuador Sensor
2. El administrador selecciona una opción Si selecciona controlador, ver sección 1 Si selecciona actuador, ver sección 2 Si selecciona sensor, ver sección 3		

Sección 1	
	2.1.1 El sistema muestra la interfaz para gestionar un controlador.
2.1.2 El administrador introduce la información para añadir, modificar o eliminar un controlador, según corresponda.	2.1.3 El sistema recepciona la información, terminando así el caso de uso.
Sección 2	
	2.2.1 El sistema muestra la interfaz para gestionar un actuador.
2.2.2 El administrador introduce la información para añadir, modificar o eliminar un actuador, según corresponda.	2.2.3 El sistema recepciona la información, terminando así el caso de uso.
Sección 3	
	2.3.1 El sistema muestra la interfaz para gestionar un sensor.
2.3.2 El administrador introduce la información para añadir, modificar o eliminar un sensor, según corresponda.	2.3.3 El sistema recepciona la información, terminando así el caso de uso.
Poscondiciones	El software queda configurado para controlar todos los dispositivos conectados al sistema domótico.

Tabla 13: Caso de uso expandido “Gestionar Dispositivos”.

Caso de uso	CU-2	Controlar Conexión Lógica y/o Física de Dispositivos
Actores	Administrador	
Resumen:		
Este caso de uso inicia cuando el administrador selecciona alguna de las opciones relacionadas con la conexión lógica o física de los dispositivos, disponibles en varias de las interfaces de la aplicación. Permite conectar y desconectar por software parte del sistema domótico y mantener informado al administrador sobre el estado de cada uno de los controladores y dispositivos. El caso de uso finaliza con la configuración de la aplicación.		
Precondiciones	El actor debe estar autenticado.	
Referencias	RF2 (RF2.1, RF2.2, RF2.3); RF3 (RF3.1, RF3.2); RF4 (RF4.1, RF4.2)	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. El administrador selecciona la opción correspondiente a la conexión del dispositivo en cuestión.		1.1 El sistema conecta o desconecta el dispositivo de acuerdo a la acción del administrador
		1.2 Refleja el estado del controlador o dispositivo en la interfaz, y finaliza el caso de uso.

Flujo alternativo	
Acción del actor	**Respuesta del sistema**
2. El actor selecciona la opción correspondiente a la conexión del dispositivo en cuestión.	2.1 El sistema envía un mensaje al actor informando que el estado del dispositivo no puede ser cambiado y finaliza el caso de uso
Poscondiciones	El software queda configurado para controlar los dispositivos conectados al sistema.

Tabla 14: Caso de uso expandido “Controlar Conexión Lógica y/o Física de Dispositivos”.

Caso de uso	CU-3	Controlar Actuadores
Actores	Administrador, Usuario	
Resumen:		
Este caso de uso inicia cuando el actor decide conocer y/o cambiar el estado de uno o varios de los actuadores. Estas opciones se encuentran en diferentes interfaces que de manera gráfica muestran el estado de cada uno de los actuadores y permiten cambiarlo. Una vez realizadas las acciones sobre los actuadores el sistema registra esta información y finaliza el caso de uso.		
Precondiciones	El actor debe estar autenticado.	
Referencias	RF5 (RF5.1, RF5.2)	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. El actor selecciona la opción correspondiente al control del estado de un actuador.		1.1 El sistema cambia el estado del dispositivo actuador de acuerdo a la acción del actor
		1.2 Refleja el estado del dispositivo actuador en la interfaz del sistema, y finaliza el caso de uso.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
2. El actor selecciona la opción correspondiente al control del estado de un actuador que no puede ser cambiado en ese momento.		2.1 El sistema envía mensaje al actor informando que el estado del dispositivo actuador no puede ser cambiado y finaliza el caso de uso
Poscondiciones	El software queda configurado para controlar los dispositivos conectados al sistema.	

Tabla 15: Caso de uso expandido “Controlar Actuadores”.

Caso de uso	CU-4	Obtener Estados Sensores
Actores	Administrador, Usuario	
Resumen:		
El caso de uso inicia cuando el actor accede a los controles de cada uno de los sensores para conocer su estado en el sistema domótico. Estas opciones se encuentran en diferentes interfaces que gráficamente muestran la información de cada sensor, incluyendo datos como hora de activación, tiempo activado, entre otros. Luego de mostrar dicha información el sistema registra la configuración si hay algún cambio y regresa a la interfaz principal, finalizando así el caso de uso.		
Precondiciones	El actor debe estar autenticado.	

Referencias	RF6.	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. El actor selecciona la opción correspondiente al control del estado de un sensor.		1.1 El sistema muestra el estado del dispositivo sensor de acuerdo al seleccionado por el actor
		1.2 Refleja información como es hora de activación y tiempo de activado del dispositivo sensor en la interfaz del sistema, finalizando el caso de uso.
Poscondiciones	El software obtiene el estado de cada uno de los sensores y lo muestra al usuario.	

Tabla 16: Caso de uso expandido “Obtener Estados Sensores”.

Caso de uso	CU-5	Gestionar Reglas
Actores	Administrador, Usuario	
Resumen:		
El caso de uso inicia cuando el actor selecciona alguna de las opciones relacionadas con gestionar reglas, a partir de lo cual la aplicación muestra los controles necesarios para el registro de la información que se desea gestionar de la regla en cuestión. Los controles y la representación gráfica de cada dispositivo acoplado al sistema permiten ordenarse y armar reglas para automatizar algún proceso, por ejemplo el apagado y encendido de luminarias y equipos electrodomésticos en determinados horarios o intervalos de tiempo. Luego de gestionadas las reglas el actor guardará los cambios efectuados finalizando de esta forma el caso de uso.		
Precondiciones	El actor debe estar autenticado.	
Referencias	RF7 (RF7.1, RF7.2, RF7.3)	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. El actor selecciona gestionar reglas de automatización		1.1 El sistema muestra las opciones para gestionar reglas de automatización
2. El actor elige la opción correspondiente.		2.1- Si elige a) Adicionar regla, ir a la sección Adicionar Regla b) Modificar regla, ir a la sección Modificar Regla c) Eliminar regla, ir a la sección Eliminar Regla
Sección “Adicionar Regla”		
		1- El sistema muestra la interfaz de adicionar regla, mediante un formulario con los campos generales que se deben introducir.

2- El actor construye la regla ordenando los controles y dispositivos necesarios.	2.1 El sistema valida la información, verificando que ésta es consistente. Si la información validada es incorrecta ver flujo alternativo
	2.2 El sistema guarda la regla.
	2.3 Se pone en ejecución la regla elaborada, finalizando así el caso de uso.
Sección "Modificar Regla"	
	1.1- El sistema muestra las reglas existentes.
2- El actor selecciona la regla que desea modificar.	2.1- El sistema muestra la información de la regla seleccionada.
3- El actor modifica la regla ordenando los controles y dispositivos necesarios.	3.1 El sistema valida la información, verificando que ésta es consistente. Si la información validada es incorrecta ver flujo alternativo
	3.2 El sistema guarda la regla.
	3.3 Se pone en ejecución la regla modificada, finalizando así el caso de uso.
Sección "Eliminar Regla"	
	1.1- El sistema muestra las reglas existentes.
2- El actor selecciona la regla que desea eliminar.	2.1- El sistema pide confirmación de que desea eliminar la regla seleccionada.
3- El actor confirma la acción.	3.1- El sistema elimina la regla seleccionada, finalizando el caso de uso.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	1 Se muestra un mensaje de error en el proceso de gestión de la regla
	2 Se retorna a la acción 2 del flujo normal
Poscondiciones	El software queda configurado con las reglas necesarias para ejecutar los procesos de automatización.

Tabla 17: Caso de uso expandido "Gestionar Reglas".

Caso de uso	CU-6	Gestionar Alarmas
Actores	Administrador, Usuario	
Resumen:		

El caso de uso inicia cuando el actor accede a alguna de las opciones correspondientes a gestionar alarmas. Con esto se muestra un conjunto de controles y la representación gráfica de aquellos sensores que pueden utilizarse en la confección de algún tipo de alarma, como por ejemplo el caso de sensores de humo, fuego, agua, gas entre otros. Concluida la gestión de las distintas alarmas el actor guarda la configuración finalizando el caso de uso.	
Precondiciones	El actor debe estar autenticado.
Referencias	RF8 (RF8.1, RF8.2, RF8.3)
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El actor selecciona gestionar alarmas.	1.1 El sistema muestra las opciones para gestionar alarmas.
2. El actor elige la opción correspondiente.	2.1- Si elige a) Adicionar alarma, ir a la sección Adicionar Alarma b) Modificar alarma, ir a la sección “Modificar Alarma” c) Eliminar alarma, ir a la sección “Eliminar Alarma”.
Sección “Adicionar Alarma”	
	1- El sistema muestra la interfaz de adicionar alarma, mediante un formulario con los campos generales que se deben introducir.
2- El actor construye la alarma ordenando los controles y dispositivos necesarios.	2.1 El sistema valida la información, verificando que ésta es consistente. Si la información validada es incorrecta ver flujo alternativo
	2.2 El sistema guarda la alarma.
	2.3 Se pone en ejecución la alarma elaborada, finalizando así el caso de uso.
Sección “Modificar Alarma”	
	1.1- El sistema muestra las alarmas existentes.
2- El actor selecciona la alarma que desea modificar.	2.1- El sistema muestra la información de la alarma seleccionada.
3- El actor modifica la alarma ordenando los controles y dispositivos necesarios.	3.1 El sistema valida la información, verificando que ésta es consistente. Si la información validada es incorrecta ver flujo alternativo
	3.2 El sistema guarda la alarma.
	3.3 Se pone en ejecución la alarma modificada, finalizando así el caso de uso.

Sección “Eliminar Alarma”	
	1.1- El sistema muestra las alarmas existentes.
2- El actor selecciona la alarma que desea eliminar.	2.1- El sistema pide confirmación de que desea eliminar la alarma seleccionada.
3- El actor confirma la acción.	3.1- El sistema elimina la alarma seleccionada, finalizando el caso de uso.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	2.1.1 Se muestra un mensaje de error en el proceso de gestión de la alarma.
	2.1.2 Se retorna a la acción 2 del flujo normal de eventos.
Poscondiciones	El software queda configurado con las alarmas necesarias para garantizar una adecuada seguridad.

Tabla 18: Caso de uso expandido “Gestionar Alarmas”.

Conclusiones del capítulo

Con la presentación del modelo de dominio, los requerimientos y la especificación de los casos de uso se puede entender el contexto en que se ubica la aplicación que se desarrolla. Este capítulo ofrece una visión de cómo está estructurada la aplicación y qué función tiene cada una de las partes componentes, lo cual conduce a afirmar que el producto cumple su objetivo de una forma bien organizada y satisfactoria. Después de definidas las características del sistema domótico se está en condiciones de hacer el análisis y el diseño partiendo de las mismas.

Capítulo 3 Análisis y diseño del sistema

En este capítulo se presentan los principales artefactos resultantes de la fase de análisis y diseño siguiendo la metodología de desarrollo RUP y teniendo en cuenta que el objetivo central de esta fase es transformar los requisitos en una especificación que describa cómo implementar el sistema. Se presentan: los diagramas de clases del análisis, mostrando las clases participantes y relaciones entre ellas; sus diagramas de secuencia, evidenciando el orden de las acciones en los casos de uso cuando son invocados. Se abordará además lo referente al tratamiento de errores y la seguridad.

3.1 Análisis

Durante el análisis, los 8 requisitos funcionales capturados en el capítulo anterior fueron especificados de manera más precisa y estructurados de un modo que facilita su mantenimiento. Además mediante el lenguaje de los desarrolladores se esbozó de forma clara cómo llevar a cabo la funcionalidad dentro del sistema incluida la funcionalidad significativa para la arquitectura; por lo que se logró una primera aproximación al diseño. El modelo de clases del análisis está estructurado por clases y paquetes estereotipados que proporcionan la estructura de la vista interna del sistema. Este modelo define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo. A continuación se muestran los diagramas de clase del análisis para los 6 casos de uso.

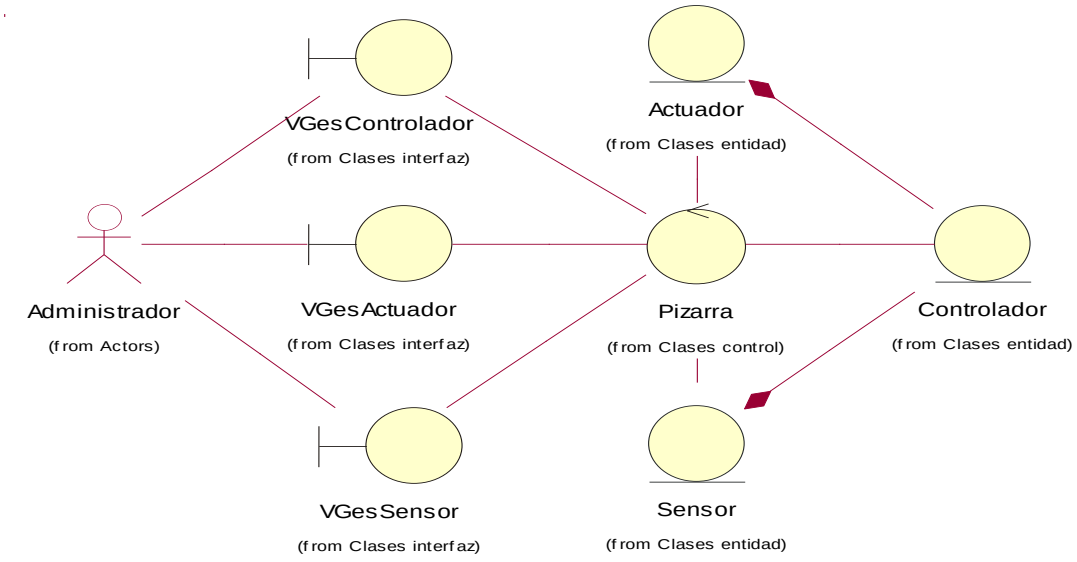

Figura 13: Diagrama de clases del análisis CU: Gestionar Dispositivos.

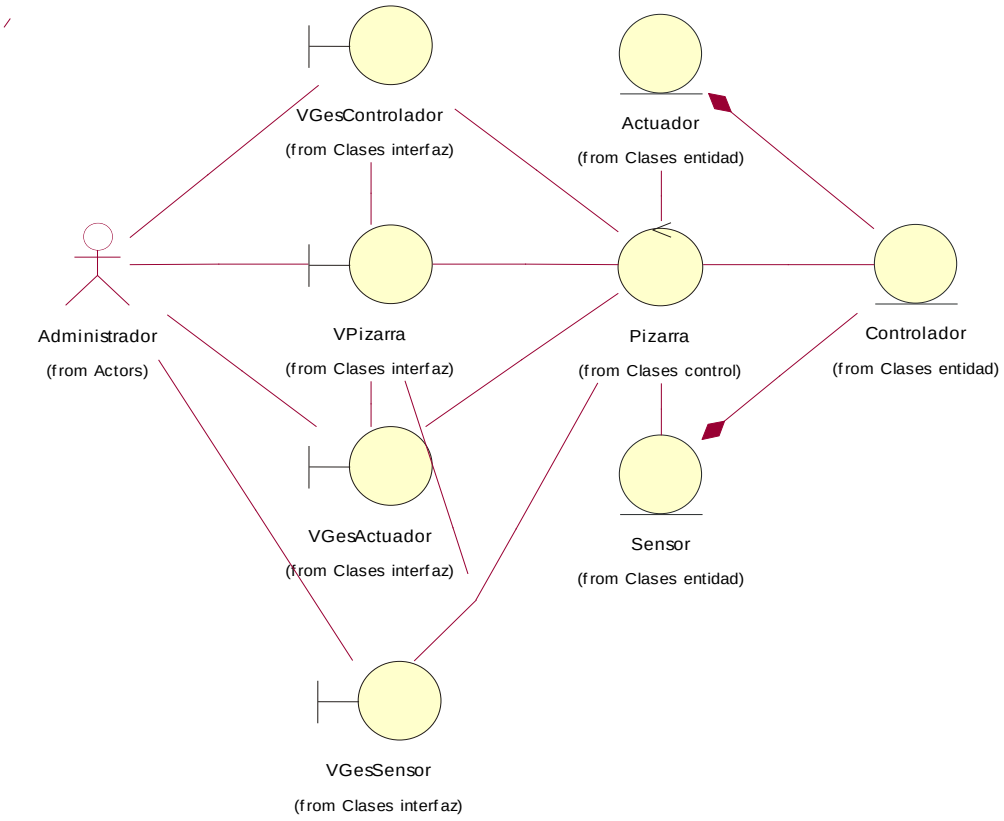

Figura 14: Diagrama de clases del análisis CU: Controlar Conexión Lógica y/o Física de Dispositivos

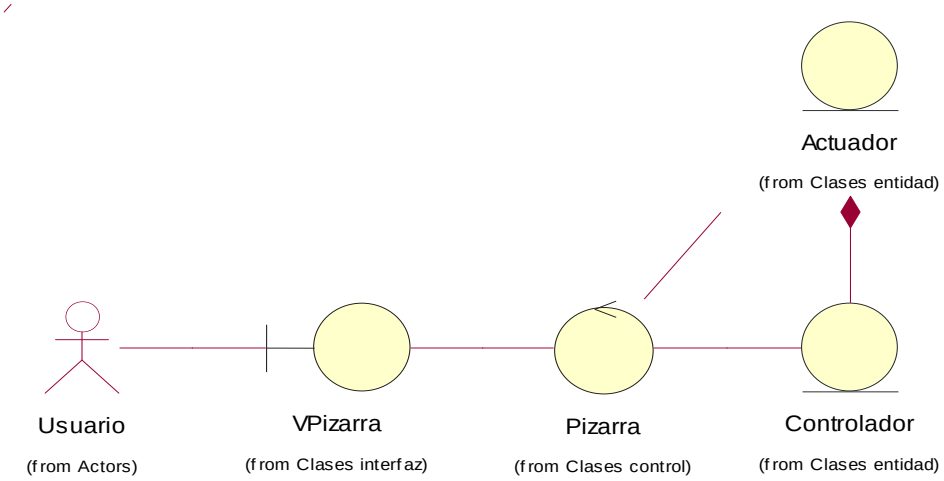

Figura 15: Diagrama de clases del análisis CU: Controlar Actuadores

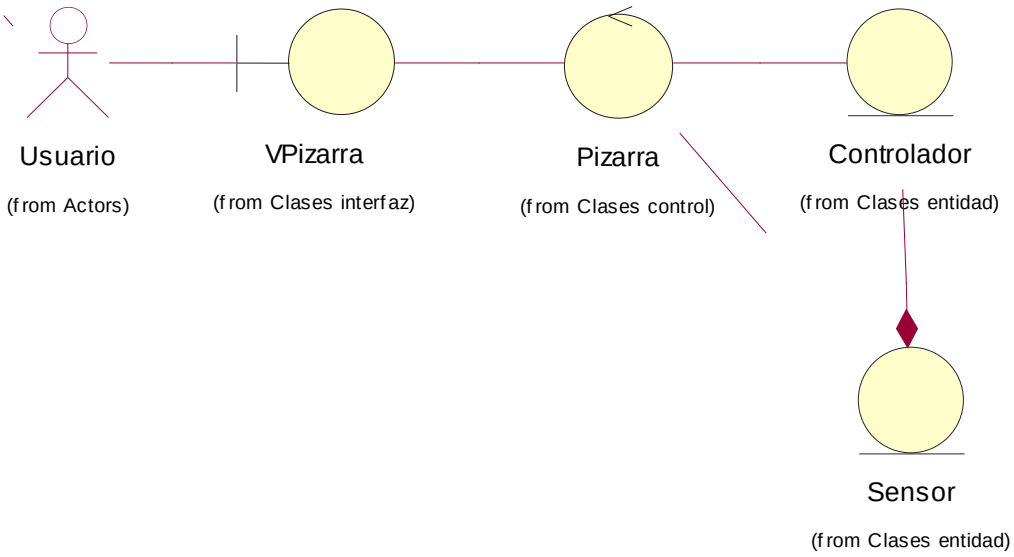

Figura 16: Diagrama de clases del análisis CU: Obtener Estados Sensores

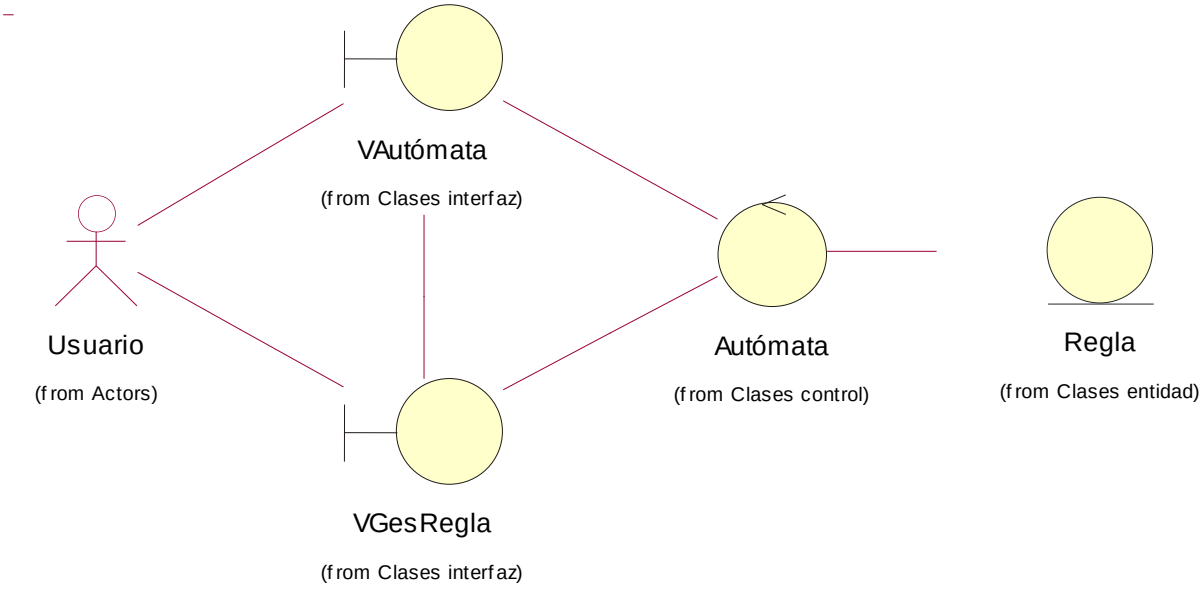

Figura 17: Diagrama de clases del análisis CU: Gestionar Reglas.

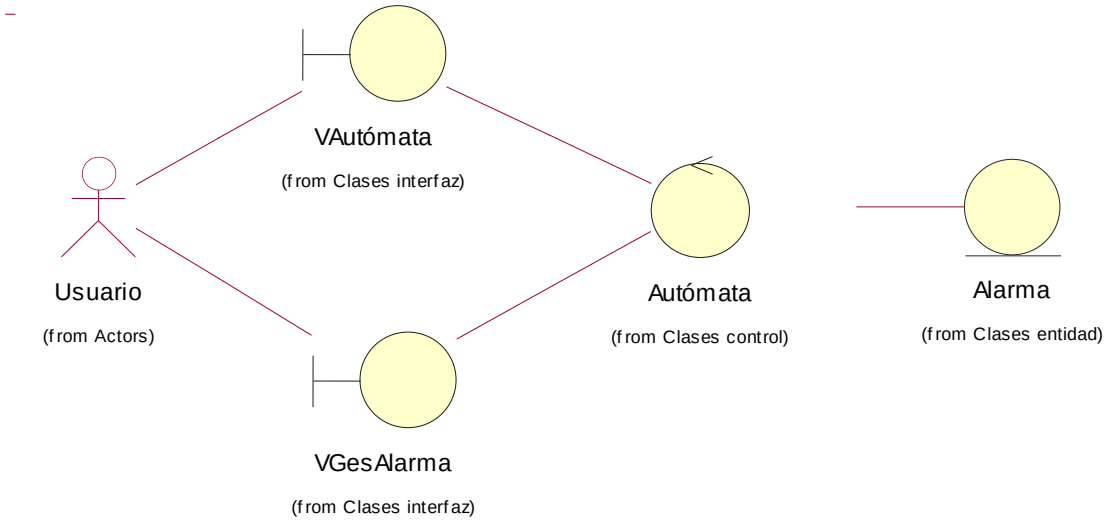

Figura 18: Diagrama de clases del análisis CU: Gestionar Alarmas

3.2 Diseño

Concibiendo el diseño como un refinamiento del análisis que tiene en cuenta los requisitos no funcionales y permite determinar cómo cumple el sistema sus objetivos; se procedió a la realización del mismo, obteniendo como resultado los artefactos necesarios para la implementación del sistema sin ambigüedades. En el diseño se modeló el sistema incluyendo la arquitectura, para que soporte los requisitos y las restricciones impuestas. Además fueron tomados en consideración los principios básicos del diseño, lo que permitió obtener un producto en el cual: no se reinventó nada, presenta uniformidad e integración, está estructurado para admitir cambios y se fue valorando su calidad durante el proceso.

3.2.1 Descripción de la arquitectura

Tomando en consideración que de acuerdo con el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE por sus siglas en Inglés) a partir del año 2000, se adopta como definición de Arquitectura de Software: la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución; podemos decir que es una vista estructural de alto nivel, que ocurre muy tempranamente en el ciclo de vida y que define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales.

La vista estructural de alto nivel del sistema domótico objeto de esta tesis, está compuesta por tres módulos independientes que se interrelacionan mediante interfaces bien definidas para lograr el funcionamiento de la aplicación y que constituyen una base sólida para su desarrollo coherente.

En la figura 19 se representan los componentes del sistema domótico y las relaciones entre ellos, puede observarse que existen 4 paquetes, tres de los cuales representan los módulos de comunicación, pizarra y automatización y el cuarto es una librería .jar encargada de la gestión y trabajo con la base de datos que se encuentra embebida en el sistema. Además se muestra el uso de una dll externa al sistema domótico, que permite la comunicación mediante el puerto paralelo.

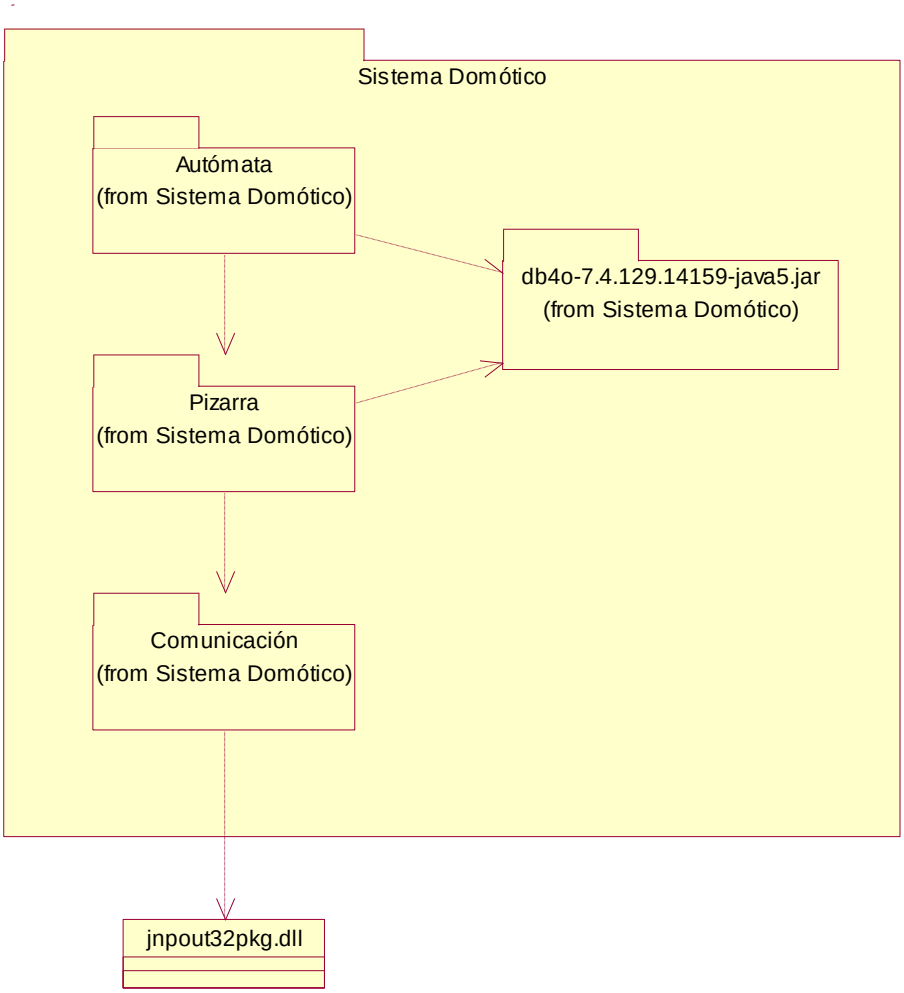

Figura 19: Diagrama de arquitectura del sistema

3.2.2 Patrones utilizados

Un patrón es una solución a un problema en un contexto dado, codifica conocimiento específico acumulado por la experiencia de desarrolladores en un dominio. Cada patrón describe un problema que ocurre frecuentemente en nuestro ambiente, y el núcleo de la solución a dicho problema, de modo tal que pueda ser utilizada reiteradamente .Es por ello que un patrón es una pareja de problema/solución representada por un nombre y aplicable a varios contextos; además, viene acompañado de una sugerencia sobre la manera de usarlo en situaciones nuevas. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos. A continuación se mencionan los patrones GRASP que serán utilizados para diseñar eficazmente el sistema domótico:

Nombre	**Experto.**
Problema	¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?
Solución	Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
Nombre	**Creador.**
Problema	¿Quién sería responsable de crear una nueva instancia de alguna clase?
Solución	Se asigna la responsabilidad de que una clase B cree un objeto de la clase A solamente cuando: B contiene a A, B es una agregación (o composición) de A, B almacena a A, B tiene los datos de inicialización de A (datos que requiere su constructor), o B usa a A.
Nombre	**Bajo Acoplamiento.**
Problema	¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?
Solución	Asignar una responsabilidad para mantener bajo acoplamiento. Debe haber pocas dependencias entre las clases.
Nombre	**Alta Cohesión.**
Problema	¿Cómo mantener la complejidad dentro de los límites manejables?
Solución	Asignar una responsabilidad de modo que la cohesión siga siendo alta. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.
Nombre	**Controlador**
Problema	¿Quién debería encargarse de atender un evento del sistema?
Solución	Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

Tabla 19: Patrones utilizados en el diseño del sistema.

Además de los anteriores, se utilizó el patrón *GoF Singleton* que garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a tal instancia.

3.2.3 Diagramas de interacción

Un diagrama de interacción está constituido por un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Los diagramas de interacción pueden ser de secuencia o de colaboración. Un diagrama de secuencia es un diagrama de interacción que destaca la disposición temporal de los mensajes.

Para cada caso de uso se han realizado los diagramas de interacción, específicamente diagramas de secuencia, donde se expone el flujo principal de información entre los objetos del diseño, con sus métodos y parámetros. A continuación se muestran los 5 diagramas de secuencia más significativos de los elaborados para los 6 casos de uso documentados para el sistema (figuras 20 a 24) y en el anexo 2 se presentan sus correspondientes diagramas de colaboración.

.

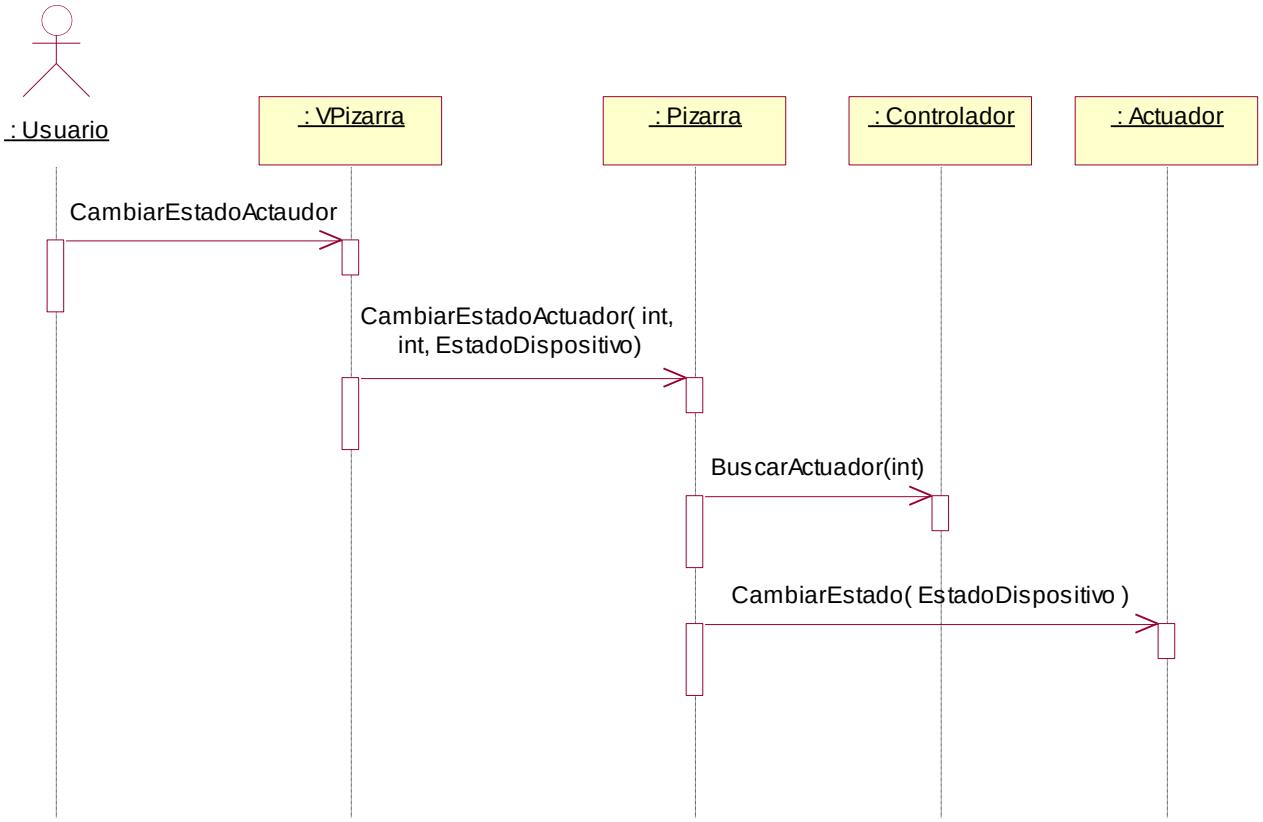

Figura 20: Diagrama de secuencia CU: Controlar Actuadores.

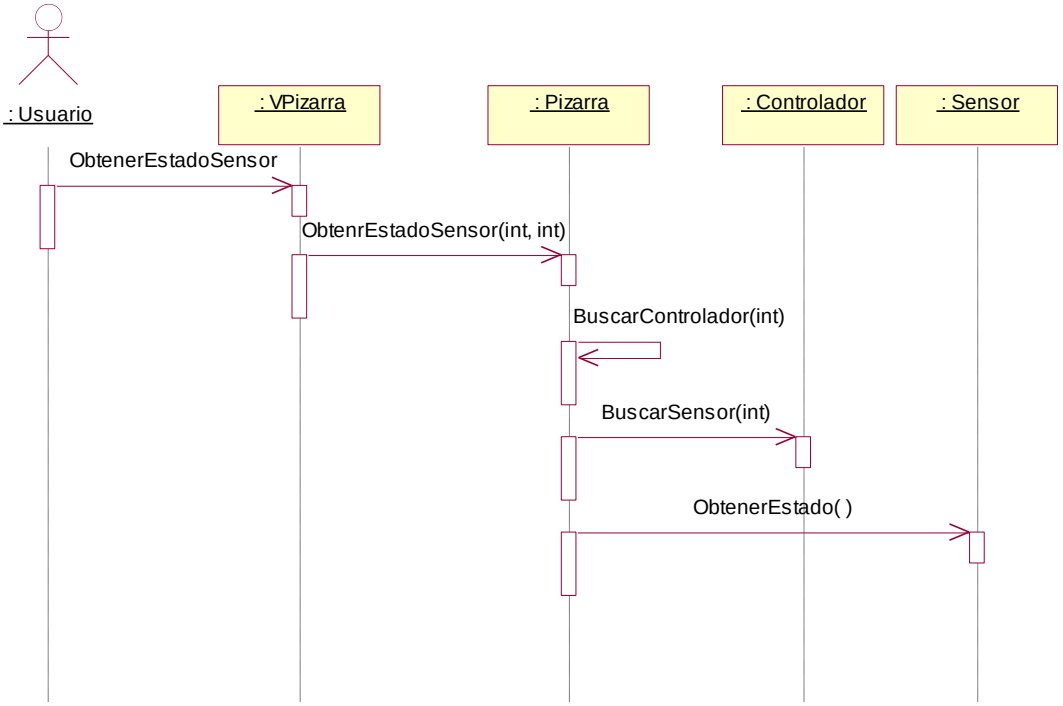

Figura 21: Diagrama de secuencia CU: Obtener Estados Sensores.

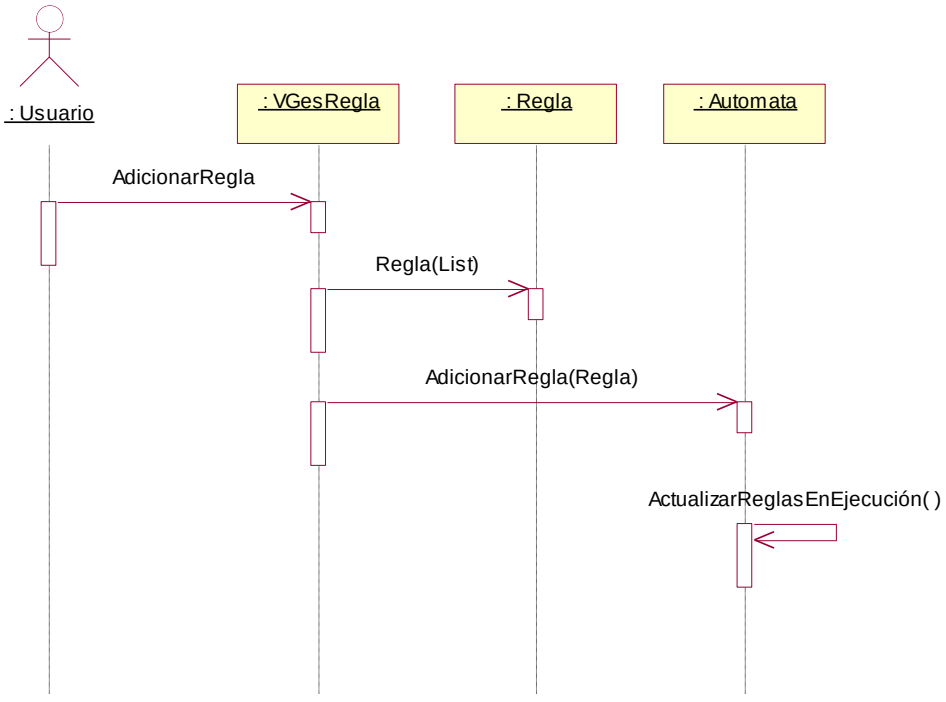

Figura 22: Diagrama de secuencia CU: Gestionar Reglas, sección 1.

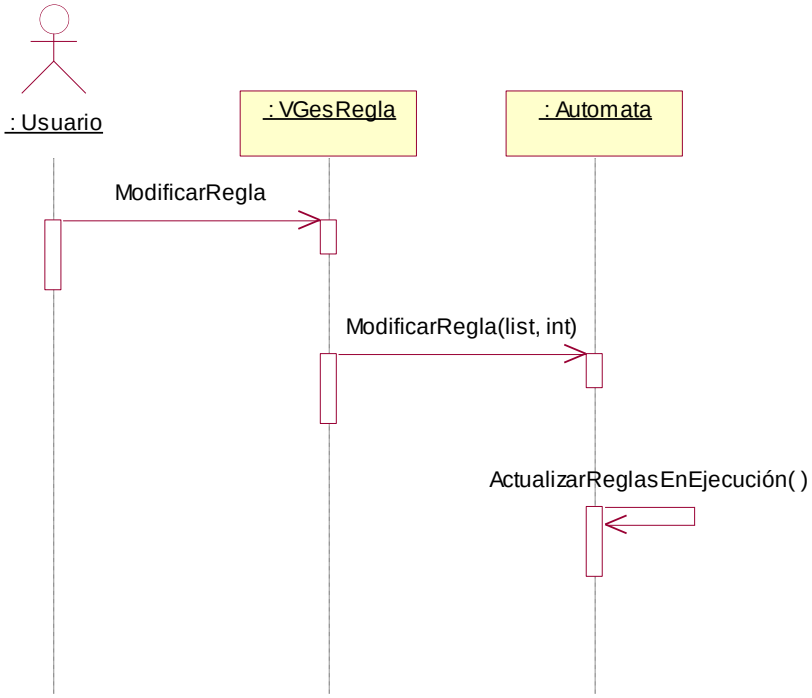

Figura 23: Diagrama de secuencia CU: Gestionar Reglas, sección 2.

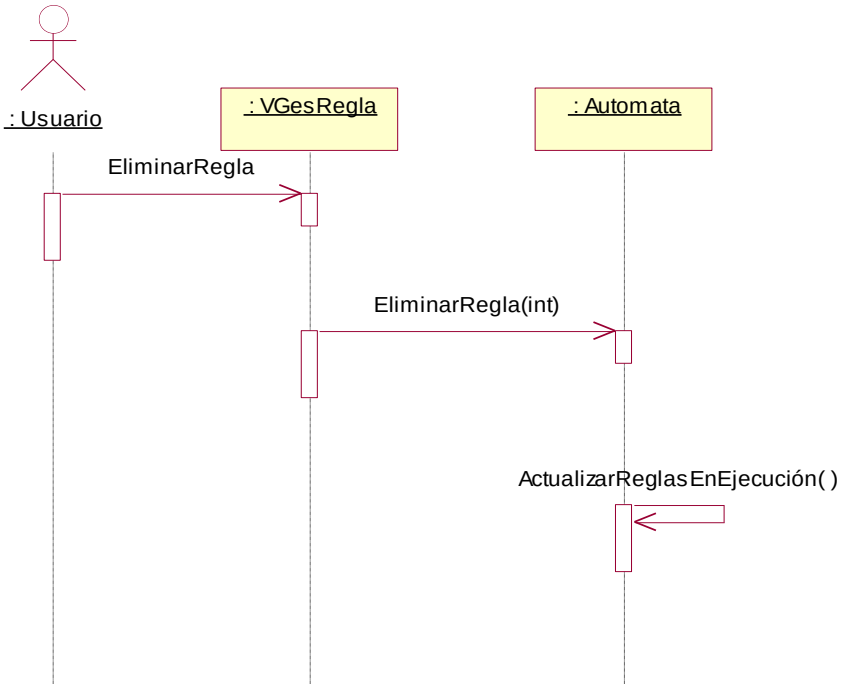

Figura 24: Diagrama de secuencia CU: Gestionar Reglas, sección 3.

3.2.4 Diagramas de clases del diseño

Los diagramas de clases del diseño muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Gráficamente, un diagrama de clases es una colección de nodos y arcos. Los subsistemas que contienen las clases de diseño a menudo participan en la realización de varios casos de uso. Estos diagramas se utilizan para modelar la vista de diseño estática de un sistema principalmente, e incluye modelar el vocabulario del sistema, modelar las colaboraciones y modelar esquemas.

Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa. En este trabajo se estructuraron por cada uno de los 3 módulos del sistema (Comunicación, Pizarra y Autómata) y se muestran en las figuras 25, 26 y 27, que aparecen a continuación.

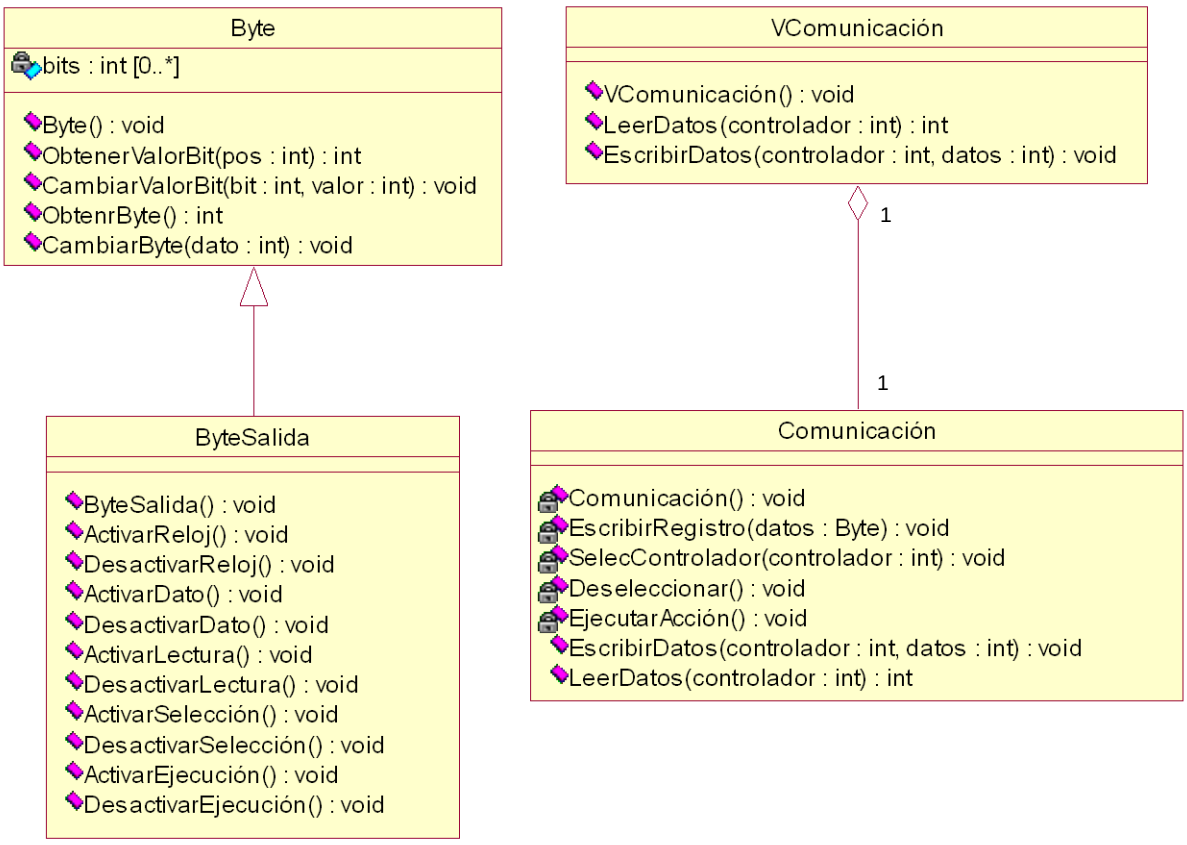

Figura 25: Diagrama de clases del diseño. Módulo Comunicación

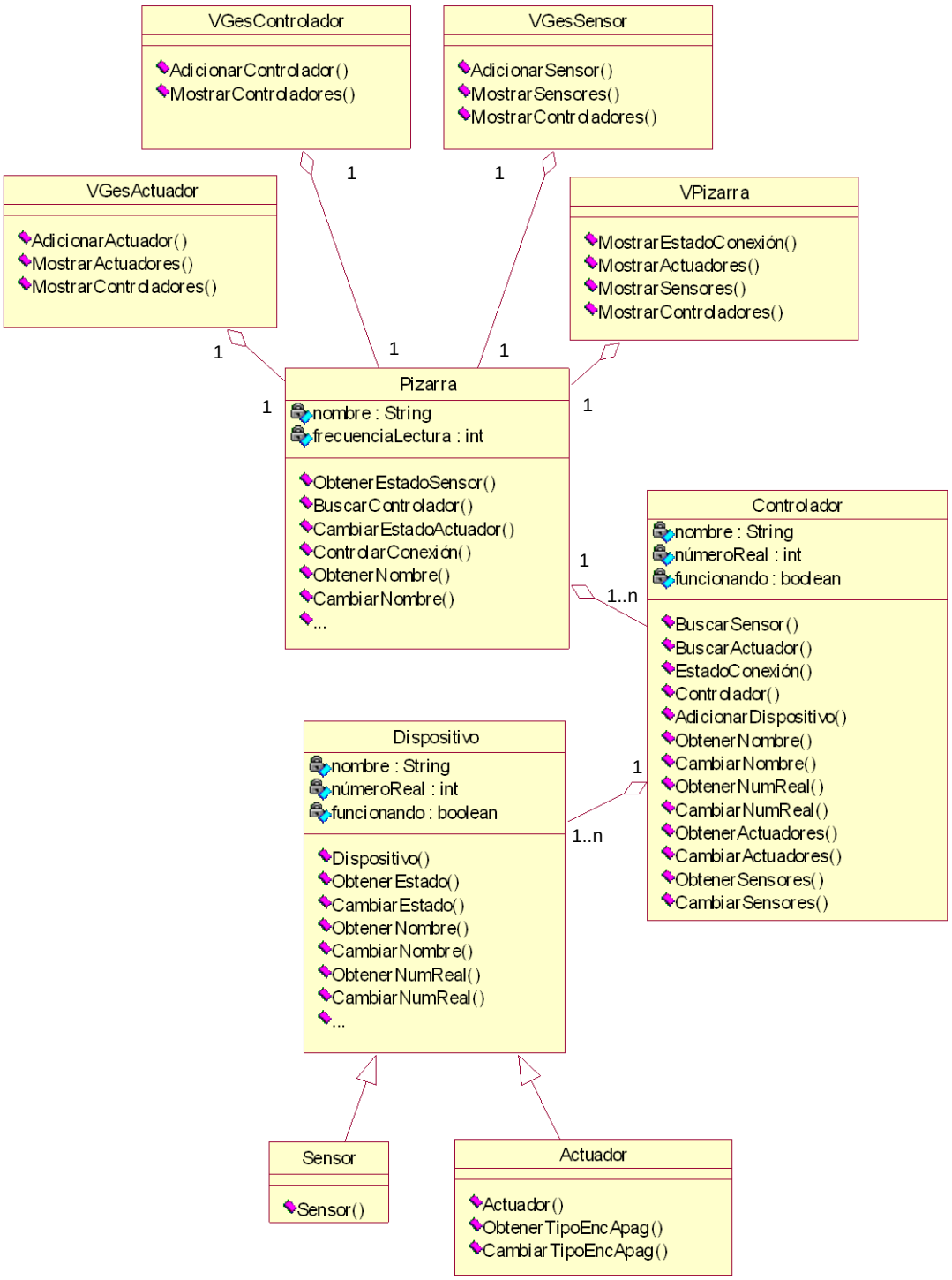

Figura 26: Diagrama de clases del diseño. Módulo Pizarra

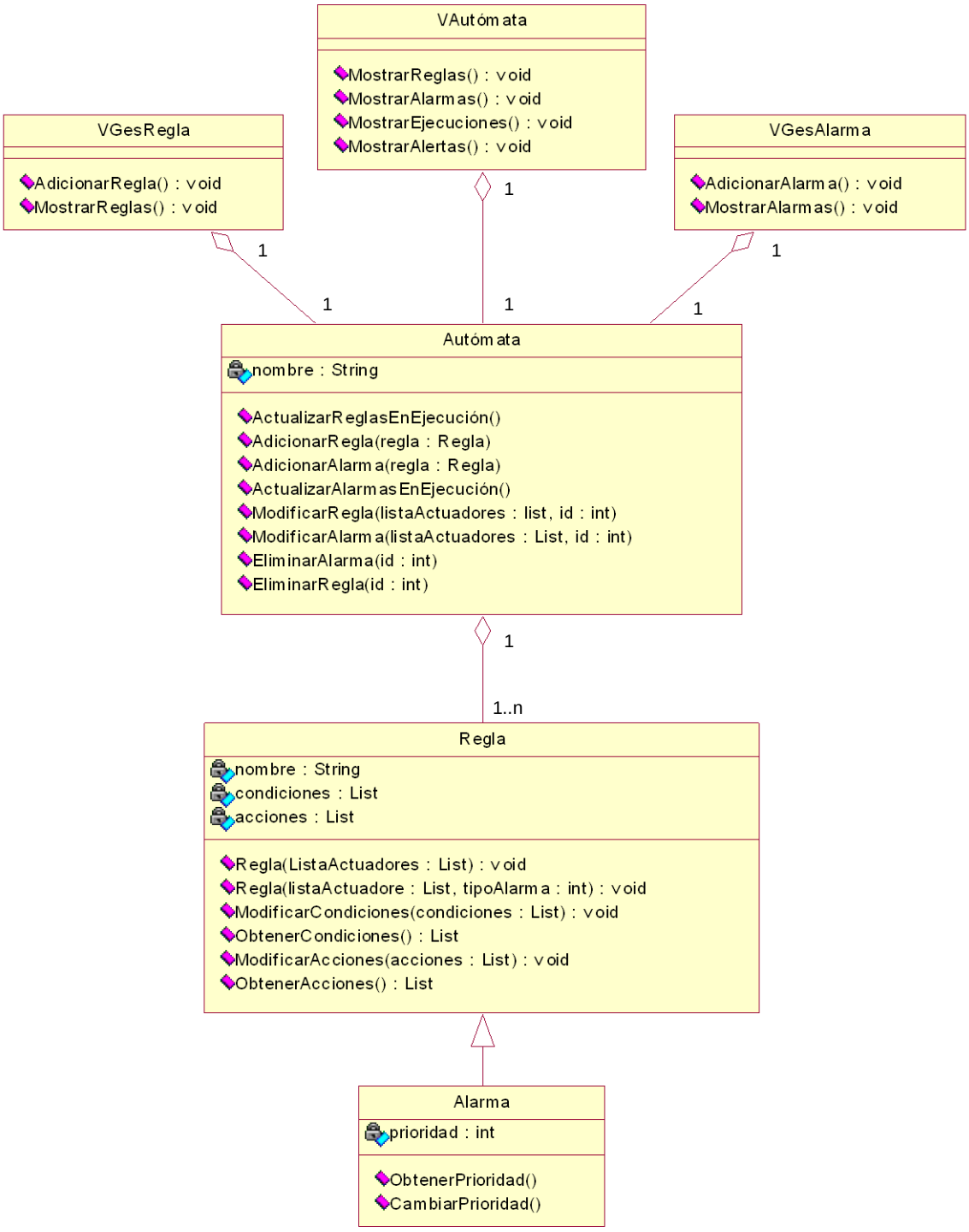

Figura 27: Diagrama de clases del diseño. Módulo Autómata.

3.2.5 Descripción de las clases del diseño

A continuación se describen las clases fundamentales del sistema domótico, organizadas por cada uno de los módulos (Comunicación, Pizarra y Automatización).

Nombre	Comunicación	
Tipo de clase	Controladora	
Atributos	**Tipo**	**Descripción**
puerto	PPort	Clase de librería de comunicación con puerto paralelo.
Métodos	**Descripción**	
Comunicación()	Constructor por defecto.	
EscribirDatos(tarjeta: int, datos: int) : void	Escribe una cadena de 8 bit de datos en serie, sincronizados con el pulso del reloj. Se le pasa el número del controlador y el dato a escribir.	
LeerDatos(tarjeta: int) : int	Lee una cadena de 8 bit de datos en serie, sincronizados con el pulso del reloj. Devuelve el dato en decimal.	

Tabla 20: Descripción de la clase del diseño “Comunicación”

Nombre	Byte	
Tipo de clase	Entidad	
Atributos	**Tipo**	**Descripción**
bits	int []	Arreglo de enteros.
Métodos	**Descripción**	
Byte()	Constructor por defecto.	
Byte(dato: int)	Constructor al que se le pasa un entero y construye el Byte correspondiente.	
ObtenerValorBit (pos: int): int	Devuelve el valor del bit correspondiente a la posición que se le pasa como parámetro dentro del Byte.	
CambiarValorBit(bit :int, valor : int) : void	Cambia el valor del bit correspondiente a la posición que se le pasa como parámetro dentro del Byte.	
ObtenerByte() : int	Devuelve el valor del Byte en decimal.	
CambiarByte(dato : int) : void	Cambia el valor del Byte pasándole un entero. El valor que recibe como parámetro es en decimal.	

Tabla 21: Descripción de la clase del diseño “Byte”

Nombre	ByteSalida	
Tipo de clase	Entidad	
Atributos	**Tipo**	
Métodos	**Descripción**	
ByteSalida()	Constructor. Devuelve un objeto de tipo ByteSalida utilizado para la comunicación con los controladores, el cual tiene los bits 2, 3 y 4 en un estado alto y permite variar fácilmente los estados de los otros bits participantes en la comunicación con los dispositivos.	

ActivarReloj(): void	Activa el bit del reloj (ponerlo a 0 ó a 1), activo a 1.
DesactivarReloj(): void	Desactiva el bit del reloj (ponerlo a 0 ó a 1), desactivado a 0.
ActivarDato(): void	Activa el bit de dato (ponerlo a 0 ó a 1), activo a 1.
DesactivarDato(): void	Desactiva el bit de dato (ponerlo a 0 ó a 1), desactivado a 0.
ActivarLectura(): void	Activa el bit de lectura (ponerlo a 0 ó a 1), activo a 0.
DesactivarLectura(): void	Desactiva el bit de lectura (ponerlo a 0 ó a 1), desactivado a 1.
ActivarSeleccion(): void	Activa el bit de selección (ponerlo a 0 ó a 1), activo a 0.
DesactivarSeleccion(): void	Desactiva el bit de selección (ponerlo a 0 ó a 1), desactivado a 1.
ActivarEjecucion(): void	Activa el bit de ejecución (ponerlo a 0 ó a 1), activo a 0.
DesactivarEjecucion(): void	Desactiva el bit de ejecución (ponerlo a 0 ó a 1), desactivado a 1.

Tabla 22: Descripción de la clase del diseño “ByteSalida”

Nombre	Pizarra	
Tipo de clase	Controladora	
Atributos	**Tipo**	**Descripción**
nombre	String	Nombre del objeto.
controladores	ArrayList<Controlador>	Lista de controladores que están instalados.
frecuenciaLectura	int	Frecuencia a la que se lee en los controladores.
comunicación	Comunicación	Clase Comunicación del módulo Comunicación.

Métodos	**Descripción**
Pizarra()	Constructor por defecto.
Pizarra(controladores :ArrayList<Controlador>)	Construye el objeto pasándole la lista de controladores.
ObtenerNombre(): String	Obtiene el nombre de la pizarra del sistema domótico.
CambiarNombre(nombre: String): void	Cambia el nombre de la pizarra del sistema domótico.
ObtenerFrecuenciaLectura(): int	Devuelve la frecuencia de lectura en los controladores.
CambiarFrecuenciaLectura(frecuenciaLectura: int): void	Cambia la frecuencia con que se lee en cada uno de los controladores.
ObtenerControladores(): ArrayList<Controlador>	Devuelve la lista completa de todos los controladores que tiene el sistema.
CambiarControladores(controladores: ArrayList<Controlador>): void	Cambia la lista de controladores que tiene el sistema.
LeerEstadosSensores(): void	Crea un hilo de ejecución que se encarga de la lectura en todas las tarjetas controladoras con un intervalo de tiempo que es decidido por el usuario con la frecuencia de lectura.
CambiarEstadoActuador(numControlador: int, numActuador: int, estado: EstadoDispositivo): boolean	Cambia el estado de un actuador instalado en el sistema.
BuscarControlador(numeroReal: int): Controlador	Busca un controlador en la lista de controladores del sistema. Si no es encontrado retorna -1.
AdicionarControlador(controlador: Controlador): void	Adiciona un controlador a la lista de controladores.

Tabla 23: Descripción de la clase del diseño “Pizarra”

Nombre	Controlador	
Tipo de clase	Entidad	
Atributos	**Tipo**	**Descripción**
nombre	String	Nombre del objeto.
númeroReal	int	Número con el cual es instalado en el sistema.
funcionando	boolean	Indica si está en funcionamiento o no.
actuadores	ArrayList<Actuador>	Lista de actuadores que están instalados.
sensores	ArrayList<Sensor>	Lista de sensores que están instalados.

Métodos	**Descripción**
Controlador()	Constructor por defecto.
Controlador(nombre: String, númeroReal: int, funcionando: boolean)	Construye el objeto pasándole el nombre, el número real y si está funcionando o no.
AñadirDispositivo (dispositivo: Dispositivo): void	Añade un dispositivo a la lista correspondiente ya sea actuador o sensor.
BuscarActuador (numeroReal: int): Actuador	Busca en la lista de actuadores un actuador dado el número real de instalación que tiene el mismo. Si no es encontrado retorna -1.
BuscarSensor (numeroReal: int): Sensor	Busca en la lista de sensores un sensor dado el número real de instalación que tiene el mismo. Si no es encontrado retorna -1.
ObtenerNombre(): String	Devuelve el nombre del controlador en cuestión.
CambiarNombre(nombre: String): void	Cambia el nombre del controlador.
ObtenerNúmeroReal(): int	Devuelve el número real de instalación que posee el controlador.
CambiarNúmeroReal(numeroReal: int): void	Cambia el número real de instalación que posee el controlador.
ObtenerActuadores(): ArrayList<Actuador>	Devuelve la lista completa de todos los actuadores con que cuenta el controlador en cuestión.
CambiarActuadores(actuadores: ArrayList<Actuador>): void	Cambia la lista completa de todos los actuadores con que cuenta el controlador.
ObtenerSensores(): ArrayList<Sensor>	Devuelve una lista completa de todos los sensores con que cuenta el controlador.
CambiarSensores(sensores: ArrayList<Sensor>): void	Cambia la lista de sensores con que cuenta el controlador.
Funcionando(): boolean	Devuelve si el controlador está funcionando o no.
CambiarFuncionando(funcionando: boolean): void	Este método define de forma lógica, si el controlador estará funcionando o no.

Tabla 24: Descripción de la clase del diseño “Controlador”

Nombre	Dispositivo	
Tipo de clase	Entidad	
Atributos	**Tipo**	**Descripción**
nombre	String	Nombre del objeto.
numeroReal	int	Número con el cual es instalado en el sistema.
estado	EstadoDispositivo	Estado del dispositivo (Enumerativo).
funcionando	boolean	Indica si está en funcionamiento o no.
Métodos		**Descripción**
Dispositivo(nombre: String, numeroReal: int)		Construye el objeto pasándole nombre y número de instalación.
Dispositivo(nombre: String, numeroReal: int, estado: EstadoDispositivo)		Construye el objeto pasándole el nombre, el número de instalación y el estado que tendrá en un inicio.
Dispositivo(nombre: String, numeroReal: int, estado: EstadoDispositivo, funcionando: boolean)		Construye el objeto pasándole nombre número de instalación, estado que tendrá en un inicio y si está funcionando o no.
ObtenerEstado(): EstadoDispositivo		Devuelve el estado del dispositivo, si está activado pasa a desactivado o viceversa.
CambiarEstado(estado: EstadoDispositivo): void		Cambia el estado del dispositivo.
ObtenerNombre(): String		Devuelve el nombre del dispositivo.
CambiarNombre(nombre: String): void		Cambia el nombre del dispositivo.
ObtenerNumeroReal(): int		Devuelve el número real de instalación del dispositivo.
CambiarNumeroReal(numeroReal: int): void		Cambia el número real de instalación del dispositivo.
Funcionando(): boolean		Devuelve si el controlador está funcionando o no.
CambiarFuncionando(funcionando: boolean): void		Este método define de forma lógica, si el dispositivo estará funcionando o no.

Tabla 25: Descripción de la clase del diseño “Dispositivo”.

Nombre	Actuador	
Tipo de clase	Entidad	
Atributos	**Tipo**	**Descripción**
tipoEncendidoApagado	TipoEncendidoApagado	Indica de qué forma es el encendido y apagado.
Métodos		**Descripción**
Actuador(nombre: String, numeroReal: int)		Construye el objeto pasándole nombre y número de instalación.
Actuador(nombre: String, numeroReal: int, tipoEncendidoApagado: TipoEncendidoApagado)		Construye el objeto pasándole nombre, número de instalación y la forma en que este actuador se encenderá y apagará.
ObtenerTipoEncendido(): TipoEncendidoApagado		Devuelve el tipo de encendido y apagado del actuador, puede ser pulso o constante.
CambiarTipoEncendido(tipoEncendido: TipoEncendidoApagado): void		Cambia el tipo de encendido y apagado del actuador en cuestión, si es de pulso puede cambiar a constante y viceversa.

Tabla 26: Descripción de la clase del diseño “Actuador”.

Nombre	Sensor	
Tipo de clase	Entidad	
Atributos		**Tipo**
Métodos		**Descripción**
Sensor(nombre: String, numeroReal: int)		Construye el objeto pasándole nombre y número de instalación.
Sensor(nombre: String, numeroReal: int, funcionando: boolean)		Construye el objeto pasándole el nombre, el número de instalación y si estará funcionando o no.

Tabla 27: Descripción de la clase del diseño “Sensor”.

Nombre	Autómata	
Tipo de clase	Controladora	
Atributos	**Tipo**	**Descripción**
nombre	String	Nombre del objeto.
Métodos		**Descripción**
Autómata()		Constructor por defecto.
AdicionarRegla(regla : Regla) :void		Adiciona una regla a la lista de reglas que tiene el sistema para lograr la automatización.
AdicionarAlarma(regla : Regla)		Adiciona una alarma a la lista de alarmas que tiene el sistema, ya sea de incidentes, averías u otras.
ActualizarReglasEnEjecución()		Actualiza las reglas que están en ejecución poniendo a funcionar aquellas que hayan tenido cambio o que se hayan incorporado nuevas al autómata.
ActualizarAlarmasEnEjecución()		Actualiza las alarmas que están en ejecución poniendo a funcionar aquellas que hayan tenido cambio o que se hayan incorporado nuevas al autómata.
ModificarRegla(listaActuadores : list, id : int)		Modifica una regla dado el id de la misma.
ModificarAlarma(listaActuadores : List, id : int)		Modifica una alarma dado el id de la misma.
EliminarAlarma(id : int)		Elimina una alarma dado el id de la misma.
EliminarRegla(id : int)		Elimina una regla dado el id de la misma.

Tabla 28: Descripción de la clase del diseño “Autómata”.

Nombre	Regla	
Tipo de clase	Entidad	
Atributos	**Tipo**	**Descripción**
nombre	String	Nombre del objeto.
condiciones	ArrayList	Grupo de condiciones que se tienen que cumplir.
acciones	ArrayList	Acciones que se ejecutan cuando se cumplen las condiciones.
Métodos		**Descripción**
Regla(ListaActuadores : List) : void		Construye el objeto pasándole una lista de actuadores.

ModificarCondiciones(condiciones : List) : void	Cambia la lista de condiciones que tiene la regla.
ObtenerCondiciones() : List	Devuelve la lista de condiciones que tiene la regla.
ModificarAcciones(acciones : List) : void	Cambia la lista de acciones que tiene la regla en cuestión.
ObtenerAcciones() : List	Devuelve la lista de acciones que tiene la regla en cuestión.

Tabla 29: Descripción de la clase del diseño “Regla”.

Nombre	Alarma	
Tipo de clase	Entidad	
Atributos	**Tipo**	**Descripción**
prioridad	int	Prioridad de la alarma.
Métodos		**Descripción**
Alarma(ListaActuadores : List) : void		Construye el objeto pasándole una lista de actuadores.
Alarma(listaActuadore : List, tipoAlarma : int) : void		Construye el objeto pasándole una lista de actuadores y el tipo de alarma.
ObtenerPrioridad() : int		Devuelve la prioridad de la alarma.
CambiarPrioridad(prioridad : int) : void		Cambia la prioridad de la alarma.

Tabla 30: Descripción de la clase del diseño “Alarma”.

3.3 Base de datos

Es común que aplicaciones Java trabajen con datos persistentes, lo que en la mayoría de los casos significa utilizar una base de datos relacional, en las que comúnmente la interfaz se complica por la “diferencia de impedancia” entre el modelo de objetos de dominio de la aplicación y el modelo relacional de la base de datos, que son diferentes y no siempre se acoplan de forma confortable.

Algunas soluciones a este problema, como *Hibernate y Java Data Objects*, están diseñadas para proporcionar al desarrollador la persistencia transparente. En cualquiera de estas soluciones, los objetos son mapeados a tablas en una Base de Datos Relacional, cuanto más complejo sea el modelo de objetos más difícil será el mapeo. La herencia y las relaciones muchos-a-muchos en particular, añaden complejidad ya que estas relaciones no se pueden representar directamente en el modelo relacional. Los árboles de herencia pueden mapearse a un conjunto de tablas de varias formas, la elección resulta de un balance entre la eficiencia de almacenamiento y la complejidad de las consultas, ya que se requiere una unión de tablas separadas para implementar relaciones muchos-a-muchos.

Almacenar objetos en una base de datos, que a su vez utiliza su propio modelo de objetos, ofrece otra solución. Durante los años 90 se desarrolló una gran variedad de Bases de Datos Orientadas a Objetos (OODBMS), pero dichas herramientas pueden ser complejas de configurar y pueden requerir

el uso de un lenguaje de definición de objetos. Los objetos se almacenan como objetos, pero no son nativos al lenguaje de la aplicación.

En algunas aplicaciones no es necesaria la sobrecarga de una potente DMBS industrial, y los requerimientos de almacenamiento de datos los proporciona mejor un motor de base de datos de objetos embebido. Es un buen momento para mirar a db4o, que tiene algunas características interesantes como:

- ✓ No hay diferencia de impedancia: los objetos se almacenan tal y como son.
- ✓ Manejo automático del esquema de base de datos.
- ✓ No hay que cambiar las clases para poder almacenarlas.
- ✓ Sin junturas en la unión con el lenguaje Java (o .NET).
- ✓ Uniones de datos automatizadas.
- ✓ Se instala añadiendo un único fichero de librería de 250Kb (jar para Java o DLL para .NET).
- ✓ Un único fichero de base de datos.
- ✓ Versionado del esquema automático.
- ✓ Consultas-por-ejemplo (*Query-By-Example*).
- ✓ S.O.D.A. (*Simple Object Database Access*), un API de consultas open source.

Una base de datos de objetos, pequeña y embebible ofrece una forma de persistencia de objetos, simple y compacta, db4o es una base de datos *Open Source* que soporta tanto Java como .NET. La simplicidad de instalación y utilización así como la ausencia de la diferencia de impedancia entre los modelos de objetos y de datos hacen que db4o sea útil en un amplio rango de aplicaciones.

3.4 Interfaz

Para la confección de la interfaz de usuario se siguieron las pautas que se presentan a continuación:

- ✓ Diseñar para una resolución de pantalla de 1024; 768, con calidad de color 32 bits.
- ✓ Debe sólo verse lo que el usuario puede usar en ese momento y no otras opciones.
- ✓ Solamente una acción a la vez.
- ✓ Para cambiar de acción el usuario debe decidir qué hacer con la que tiene en curso.
- ✓ Los controles que tengan estrecha relación están agrupados en un Panel.

La interfaz de usuario del sistema domótico está compuesta por varias ventanas y algunas de ellas se muestran en el anexo 3. La pantalla principal del módulo de mayor número de funcionalidades se divide en tres áreas, que se muestran en la figura 28 y son las siguientes: área de Controladores, área de Actuadores y área de Sensores

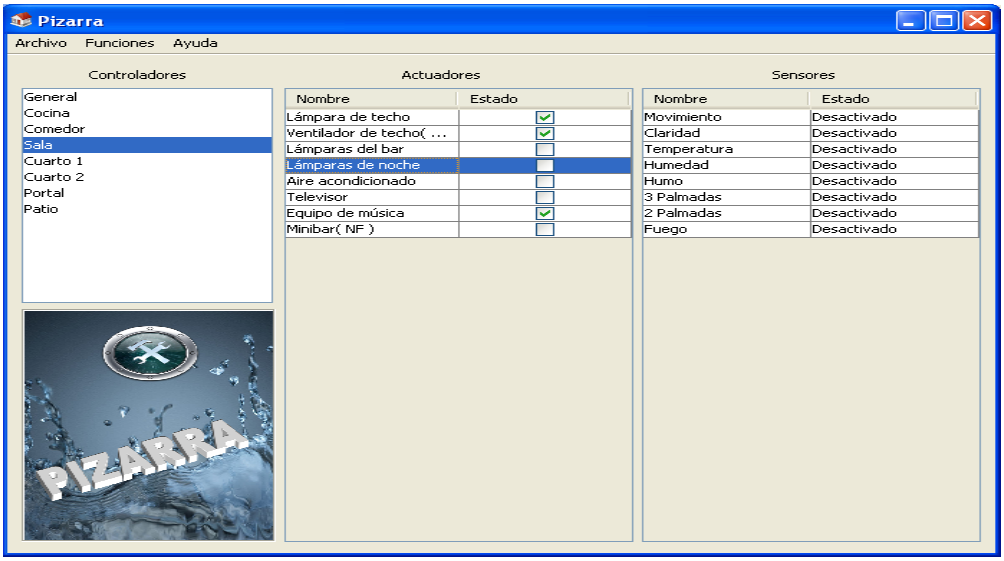

Figura 28: Interfaz del módulo Pizarra.

3.5 Tratamiento de errores

Para el tratamiento de errores se validan los datos que son introducidos por los usuarios al sistema, verificando el formato y la correspondencia con las características de cada uno de los dispositivos conectados al sistema domótico. En este proceso de validación se generan mensajes que informan al usuario oportunamente sobre los errores cometidos. Cada usuario tendrá acceso restringido según su rol y se le permitirá sólo ver las funcionalidades para las cuales obtenga permisos.

Por otro lado como el sistema domótico está implementado en Java se utilizan los mecanismos disponibles en este lenguaje para el tratamiento de excepciones. Una excepción es una condición que interrumpe el flujo normal de operaciones dentro de un programa, por ejemplo un error de programación inesperado como una división por cero o un posible error que ocurra en tiempo de ejecución, como un "disco lleno" mientras escribe un archivo. Cuando ocurre una excepción, el flujo normal de instrucciones se rompe. En concreto, se arroja la excepción y el flujo continúa en el punto en el que se capturó la excepción. Al arrojarse la excepción se permite la eliminación del código controlador de excepciones del flujo regular de operaciones. Si la condición anormal no se controla adecuadamente, pueden aparecer resultados incorrectos u otras condiciones anormales.

El constructor de lenguaje para controlar excepciones en un programa de Java es el bloque ***try-catch***. Básicamente, las declaraciones que podrían arrojar una excepción se encuentran dentro del bloque ***try***. Si se produce una excepción, se controla mediante cláusulas ***catch*** ejecutadas con éxito. Si hay varias cláusulas ***catch***, se ejecutará el bloque de código para el primer tipo de excepción que concuerde.

3.6 Seguridad

En la actualidad la seguridad en las aplicaciones informáticas adquiere una importancia vital. Desde su aparición Java ha ido incorporando elementos destinados a proporcionar un mayor control sobre la seguridad de los datos y programas. Los distintos JDK incorporan herramientas para añadir seguridad a las aplicaciones Java, entre ellas firmas digitales y transmisión segura de datos. Java permite establecer distintos niveles de seguridad, lo que posibilita una gran flexibilidad para asignar o denegar permisos.

Para garantizar la seguridad de la información del sistema domótico se crearon dos niveles, definidos como roles de usuario, que son usuario y administrador. Este último es el encargado de realizar los cambios y del buen funcionamiento del sistema, por tanto, posee control total del mismo. El resto de los usuarios sólo tendrá acceso a la información determinada por su rol. Para esto, se utilizan las herramientas que posee Java como *Java Authentication and Authorization Service* (JAAS), que se encuentran en el paquete *javax.security*.

Conclusiones del capítulo

Tras efectuar el análisis del sistema domótico se comprendieron sus requisitos de manera más precisa, analizándolos con mayor profundidad. Además, se consiguió preparar y simplificar las subsiguientes actividades del diseño. Respecto a los diagramas de clases del análisis que se obtuvieron, se concluye que pueden ser aplicados a varios diseños y cada una de sus clases de control, entidad e interfaz aísla futuros cambios al comportamiento e información que representan.

Durante el proceso de diseño de la arquitectura se encontró una forma para lograr desarrollar un sistema flexible a los cambios en los requisitos. Mediante los diagramas de secuencia se mostró la forma en que los objetos se comunican entre sí al transcurrir el tiempo, contribuyendo de manera importante a entender el comportamiento del sistema.

Se logró proponer una interfaz amigable y sencilla acorde con las exigencias, así como establecer medidas de seguridad que junto a un eficaz tratamiento de errores hacen fiable el futuro sistema.

Con esta fase de análisis y diseño se logró asignar eficientemente las responsabilidades a los componentes del software. Además, se crearon los artefactos necesarios como punto de partida para la implementación del sistema.

Capítulo 4 Implementación y prueba

En el presente capítulo se muestran los resultados alcanzados en los flujos de trabajo de implementación y pruebas, en los cuales partiendo de los artefactos obtenidos en el diseño se realizó el diagrama de despliegue que muestra los nodos necesarios para el funcionamiento del sistema. Además se realizó la implementación del sistema en términos de componentes, dando paso luego a la aplicación de las pruebas de caja negra en pos de garantizar la calidad del sistema.

4.1 Implementación.

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. Los diagramas de despliegue y componentes conforman lo que se conoce como un modelo de implementación al describir los componentes a construir, su organización y dependencia entre los nodos físicos en los que funcionará la aplicación.

Este flujo de trabajo está fuertemente determinado por el lenguaje de programación y alcanza su mayor intensidad en la fase de construcción, que es la que tiene como propósito dejar listo un producto software en su versión operativa inicial, a esta versión le incumbe tener la calidad requerida para su uso y cumplir con los requisitos determinados en el segundo capítulo.

Los objetivos de este flujo de trabajo logrados durante la investigación fueron:

- ✓ Definir la organización del sistema en términos de subsistemas de implementación organizados en capas.
- ✓ Implementar los elementos de diseño en términos de (ficheros fuentes, binarios, ejecutables y otros).
- ✓ Probar los componentes desarrollados independientemente como unidades.
- ✓ Integrar los resultados producidos en un sistema ejecutable.

4.1.1 Diagrama de despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados, es decir, se sitúa el software en el hardware que lo contiene, cada elemento de hardware constituye un nodo el cual es representado por un cubo. Un nodo es un elemento donde se ejecutan los componentes y representa su despliegue físico.

En la figura 29 se muestra el diagrama de despliegue del sistema domótico que cuenta con cuatro nodos uno de procesamiento (PC) y tres en representación de los tipos de dispositivos (Controladores,

Actuadores y Sensores). El componente PC es el encargado de ejecutar el sistema domótico que mediante la interfaz Paralelo se comunica con los controladores instalados en el sistema, los cuales a su vez tienen acoplados todos los dispositivos actuadores y sensores que se deseen. La conexión de los controladores con los actuadores y sensores se realiza con cable eléctrico de dos vías. Los controladores dominan a los actuadores suministrándoles o no energía eléctrica, mientras que los sensores funcionan como interruptores que pueden estar abiertos o cerrados.

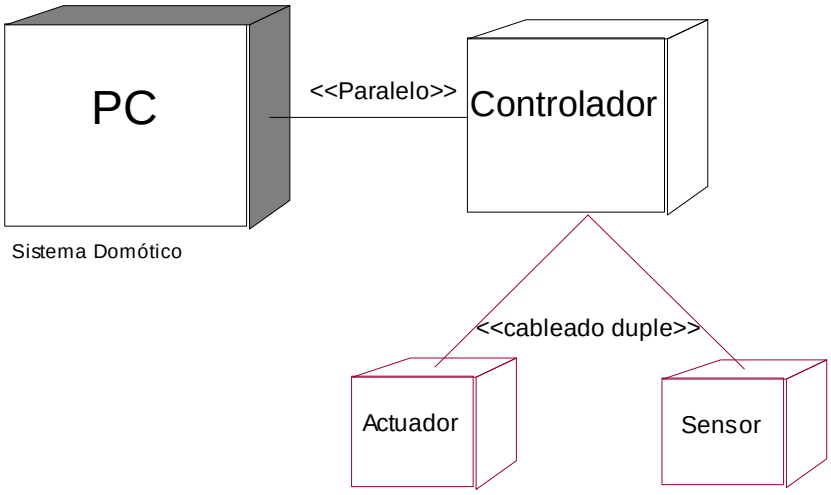

Figura 29: Diagrama de despliegue.

4.1.2 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas y mostrar las relaciones entre los elementos. Son grafos de componentes unidos a través de relaciones que pueden ser de compilación o de ejecución, y además se pueden representar las interfaces de esos componentes, interfaces se refiere a la descripción de las operaciones que se realizan en esos componentes y en esos subsistemas.

Es otra forma de representar una vista estática del sistema, que muestra la organización y dependencia que existe entre los componentes que se necesitan para ejecutar la aplicación, sean éstos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes.

El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando los subsistemas de implementación y sus dependencias a la hora de importar código.

En la figura 30 se muestra el diagrama de componentes de la aplicación que está compuesta por tres módulos: Comunicación, Pizarra y Automatización, cuyas funciones se describen brevemente a continuación.

- ✓ **Módulo de comunicación:** es el encargado de la escritura y la lectura de datos en los controladores.
- ✓ **Módulo Pizarra:** se encarga de gestionar los dispositivos del sistema domótico. Además permite interactuar con los dispositivos.
- ✓ **Módulo de Automatización:** permite gestionar reglas que garantizan la automatización de los dispositivos actuadores y sensores que se encuentren acoplados al sistema domótico.

Los módulos Pizarra y Automatización utilizan la librería db4o-7.4.191.1 4159-java5.jar para las operaciones relacionadas con las bases de datos embebida en el sistema que almacena la información en el archivo bdSalva.domo. Además el módulo Comunicación utiliza el archivo jnpount32pkg.dll para la comunicación a través del puerto paralelo.

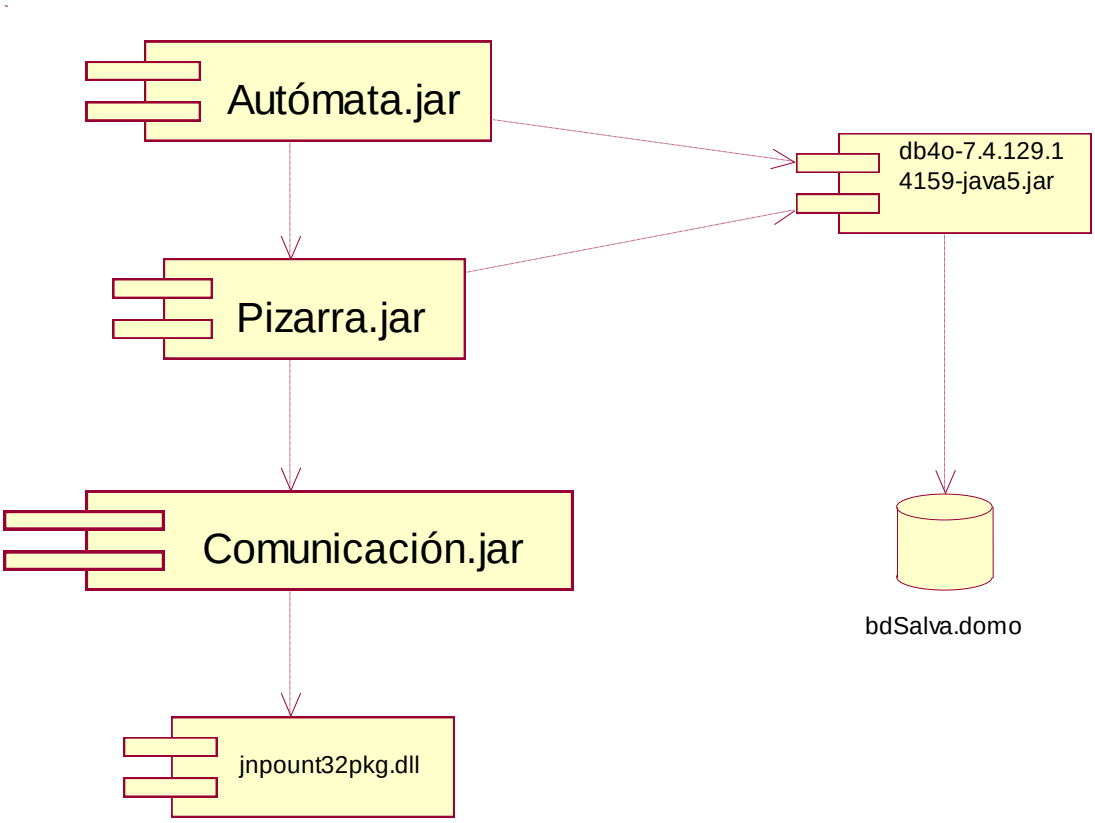

Figura 30: Diagrama de componentes.

4.2 Modelo de prueba

Los métodos de prueba de software tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con el menor tiempo y esfuerzo posibles, para ello existen dos métodos de prueba que son las pruebas de caja negra y pruebas de caja blanca, las cuales se caracterizan a continuación:

Pruebas de caja negra: Se enfocan en los requerimientos establecidos y en la funcionalidad del sistema, se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Estas pruebas no consideran la codificación dentro de sus parámetros a evaluar, es decir, no están basadas en el conocimiento del diseño interno del programa

Pruebas de caja blanca*:* Al contrario de las de caja negra, estas se basan en el conocimiento de la lógica interna del código del sistema, se realiza un minucioso examen de los detalles procedimentales, se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen si están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado.

El método de prueba seleccionado en la investigación fue el método de caja negra. Estas pruebas fueron realizadas una vez concluido el software y luego que todos los componentes de software y hardware fueron integrados. Las pruebas de sistema principalmente se centraron en verificar la interacción de los actores con el sistema.

Los datos de prueba se escogieron atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar la ejecución correcta de la aplicación y para escogerlos se siguieron los criterios que a continuación se relacionan:

- ✓ Valores fáciles: El programa se depuró con datos fáciles de comprobar.
- ✓ Valores típicos realistas: Se ensayó el programa con datos representativos de su futura aplicación.
- ✓ Valores extremos: Se utilizaron los valores límites para los rangos de la aplicación.
- ✓ Valores ilegales: Se entraron datos cuyos valores no corresponden con los requeridos por el sistema.

A continuación se muestran los principales casos de prueba realizados:

CP 1: Caso de uso Controlar Actuadores

Entrada	Resultados	Condiciones
El actor selecciona la opción correspondiente al control del estado de un actuador	El sistema cambia el estado del dispositivo actuador de acuerdo a la acción del actor y lo refleja en la interfaz gráfica.	El actor debe estar previamente autenticado y seleccionar un actuador que pueda ser cambiado de estado.
El actor selecciona la opción correspondiente al control del estado de un actuador que no puede ser cambiado.	El sistema envía un mensaje al actor informando que el estado de ese dispositivo actuador no puede ser cambiado en ese momento.	El actor debe estar previamente autenticado y seleccionar un actuador que no pueda ser cambiado de estado.

Tabla 31: Caso de prueba 1

CP 2: Caso de uso Obtener Estados Sensores

Entrada	Resultados	Condiciones
El actor selecciona la opción correspondiente al control del estado de un sensor.	El sistema muestra el estado del sensor seleccionado por el actor, reflejando información como hora de activación y tiempo de activado.	El actor debe estar previamente autenticado y seleccionar un sensor que pueda ser visualizado
El actor selecciona la opción correspondiente al control del estado de un sensor que no está funcionando.	El sistema envía un mensaje al actor informando que el sensor seleccionado no está funcionando.	El actor debe estar previamente autenticado y seleccionar un sensor que no está funcionando

Tabla 32: Caso de prueba 2

CP 3: Caso de uso Gestionar Reglas

Sección “Adicionar Regla”

Entrada	Resultados	Condiciones
El actor construye la regla ordenando correctamente los controles y dispositivos necesarios.	El sistema guarda la regla y la pone en ejecución.	El actor debe estar previamente autenticado y construir la regla correctamente
El actor construye la regla utilizando controles o dispositivos no válidos.	El sistema envía un mensaje al actor informando que existe un error en la construcción de la regla.	El actor debe estar previamente autenticado y utilizar controles o dispositivos no válidos.
El actor construye la regla dejando campos vacíos.	El sistema envía un mensaje al actor informando que faltan argumentos para la construcción de la regla.	El actor debe estar previamente autenticado y dejar campos vacíos.

Tabla 33: Caso de prueba 3, primera sección.

Sección “Modificar Regla”

Entrada	Resultados	Condiciones
El actor selecciona la regla que desea modificar y ordena correctamente los controles y dispositivos.	El sistema guarda la regla y la pone en ejecución.	El actor debe estar previamente autenticado y construir la regla correctamente
El actor selecciona la regla que desea modificar y lo hace utilizando controles o dispositivos no válidos.	El sistema envía un mensaje al actor informando que existe un error en la modificación de la regla y mantiene en ejecución la anterior.	El actor debe estar previamente autenticado y utilizar controles o dispositivos no válidos.
El actor selecciona la regla que desea modificar y lo hace dejando campos vacíos.	El sistema envía un mensaje al actor informando que faltan argumentos para la modificación de la regla y mantiene en ejecución la anterior.	El actor debe estar previamente autenticado y dejar campos vacíos.

Tabla 34: Caso de prueba 3, segunda sección.

Sección “Eliminar Regla”

Entrada	Resultados	Condiciones
El actor selecciona la regla que desea eliminar.	El sistema elimina la regla quitando la misma de la interfaz y de la lista de reglas en ejecución. Luego notifica al actor que la regla fue eliminada	El actor debe estar previamente autenticado y seleccionar la opción de eliminar regla.

Tabla 35: Caso de prueba 3, tercera sección.

CP 4: Caso de uso Gestionar Alarmas

Sección “Adicionar Alarma”

Entrada	Resultados	Condiciones
El actor construye la alarma ordenando correctamente los controles y dispositivos necesarios.	El sistema guarda la alarma y la pone en ejecución.	El actor debe estar previamente autenticado y construir la alarma correctamente
El actor construye la alarma utilizando controles o dispositivos no válidos.	El sistema envía un mensaje al actor informando que existe un error en la construcción de la alarma.	El actor debe estar previamente autenticado y utilizar controles o dispositivos no válidos.
El actor construye la alarma dejando campos vacíos.	El sistema envía un mensaje al actor informando que faltan argumentos para la construcción de la alarma.	El actor debe estar previamente autenticado y dejar campos vacíos.

Tabla 36: Caso de prueba 4, primera sección.

Sección “Modificar Alarma”

Entrada	Resultados	Condiciones
El actor selecciona la alarma que desea modificar y ordena correctamente los controles y dispositivos necesarios.	El sistema guarda la alarma y la pone en ejecución.	El actor debe estar previamente autenticado y construir la alarma correctamente
El actor selecciona la alarma que desea modificar y lo hace utilizando controles o dispositivos no válidos.	El sistema envía un mensaje al actor informando que existe un error en la modificación de la alarma y mantiene en ejecución la anterior.	El actor debe estar previamente autenticado y utilizar controles o dispositivos no válidos.
El actor selecciona la alarma que desea modificar y lo hace dejando campos vacíos.	El sistema envía un mensaje al actor informando que faltan argumentos para la modificación de la alarma y mantiene en ejecución la anterior.	El actor debe estar previamente autenticado y dejar campos vacíos.

Tabla 37: Caso de prueba 4, segunda sección.

Sección “Eliminar Alarma”

Entrada	Resultados	Condiciones
El actor selecciona la alarma que desea eliminar.	El sistema elimina la alarma quitando la misma de la interfaz y de la lista de alarmas en ejecución. Luego notifica al actor que la alarma fue eliminada	El actor debe estar previamente autenticado y seleccionar la opción de eliminar alarma.

Tabla 38: Caso de prueba 4, tercera sección.

Las pruebas del sistema estuvieron sujetas al carácter iterativo incremental de la metodología utilizada, que hace que el software se construya en fases y con las iteraciones necesarias en cada caso. Es por ello que los 4 casos de prueba definidos fueron aplicados al sistema iterativamente de acuerdo con sus características propias, resultando de este proceso aspectos como los siguientes:

- ✓ Detección de errores de ejecución, que limitaban el cumplimiento total de determinados requisitos funcionales o no funcionales del sistema.
- ✓ Recomendaciones para el perfeccionamiento de las funcionalidades y la facilidad de uso de la aplicación.

El caso de prueba que mayores dificultades arrojó fue el tercero, es decir, el referido al caso de uso Gestionar Reglas; dada la complejidad de dicho caso de uso y las funcionalidades que este genera.

Los errores detectados y las recomendaciones derivadas de la aplicación de los casos de prueba fueron corregidos e implementados respectivamente en la medida en que fueron apareciendo.

Durante el proceso de pruebas fueron evaluadas también las interfaces de usuario, lo cual permitió la corrección de aspectos como:

- ✓ Algunos errores ortográficos.
- ✓ El ordenamiento de algunos componentes visuales, para lograr una comunicación más intuitiva.
- ✓ Existencia limitada de mensajes para orientar al usuario sobre la funcionalidad de los componentes.

Todo lo anteriormente expresado permitió ir depurando el sistema hasta obtener la versión actual.

Conclusiones del capítulo

Durante el proceso de implementación se desarrolló la aplicación, fueron implementadas las clases obtenidas en el diseño y se obtuvieron los diagramas de componentes y de despliegue que permitieron describir los módulos a construir y el modo en que se realiza el despliegue de la aplicación respectivamente.

La ejecución de la fase de prueba de aseguramiento de la calidad del software se concretó mediante el método de caja negra, para lo cual se hizo una rigurosa selección de los datos, partiendo de los criterios establecidos y se efectuaron todos los casos de prueba necesarios en función de los casos de uso definidos; todo lo cual permitió demostrar que la aplicación cuenta con las características y funcionalidades para las que fue concebida.

CONCLUSIONES

El desarrollo y culminación de esta investigación ha permitido arribar a las conclusiones siguientes:

- ✓ El estudio realizado sobre el uso de la informática en el control doméstico automatizado, arrojó que la mayoría de los sistemas utilizan tecnologías costosas o no adecuadas a las condiciones socio-económicas cubanas, además permitió identificar rasgos que caracterizan la domótica y aspectos esenciales para elaborar y fundamentar la propuesta de solución al problema planteado.
- ✓ El uso de la metodología, herramientas y tecnologías seleccionadas fue factible y mediante el paso por todas las fases del ciclo de vida se desarrolló la aplicación, quedando esta ampliamente documentada con los artefactos generados mediante la metodología RUP, lo cual permitirá posibles ajustes o mejoras futuras.
- ✓ El sistema obtenido se caracteriza por ser: seguro, portable, de alto rendimiento y posee una interfaz sencilla que facilita la comunicación con el usuario, además las pruebas realizadas permitieron comprobar que con su utilización se logra el control de dispositivos y procesos domésticos de forma automatizada.

RECOMENDACIONES

- ✓ Perfeccionar la aplicación añadiendo nuevas funcionalidades a los módulos existentes, de modo que se amplíen las posibilidades de automatización del sistema.
- ✓ Añadir un módulo de inteligencia para que la automatización no se limite al trabajo con reglas, sino que incluya la utilización de algoritmos de inteligencia artificial.

BIBLIOGRAFÍA

¿Qué es NetBeans? [En línea] 2007. [Citado el: 02 de Febrero de 2009.] http://www.netbeans.org/index_es.html

AGUILERA, A. G., PULIDO, FRANCISCO, JAVIER. Seguridad y hogar digital, estrellas de SIMO 2005. Madrid, IDG Communications, 2005, nº p. 22-26.

Autores, Colectivo de. 2007. Introducción al proceso de desarrollo de software [Digital] Ciudad de La Habana: Universidad de las Ciencias Informáticas, 2007.

Bases de datos Orientadas a Objetos DB4O. [En línea] 09 de Diciembre de 2005. [Citado el: 30 de Enero de 2009.] www.mhproject.org/media/blogs/mhpenlaces/Interno/Presentaciones/.../BD.%20Orientada%20a%20Objetos%20-%20db4objects.ppt.

BATISTA CHILLÓN, Y. Y THOMPSON MARTÍNEZ, R.. 2008. Diagnóstico de vulnerabilidades de PC Vía Web. [Digital] Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

BOOCH, G. 1991. Object-Oriented Design with Applications. The Benjamin/Cummings Publishing Company. 1991.

BOOCH, G., JAMES JACOBSON, I. y RUMBAUGH, J. UML Manual de referencia. Addison Wesley. 1997.

BOUDY GONZÁLEZ, S., LEYVA FONSECA M. Desarrollo de una aplicación para el control inteligente de un robot manipulador. Tesis en opción al título de Ingeniero. Universidad de Ciencias Informáticas, Ciudad de La Habana. 2009.

BROWN, W. 1998. AntiPatterns: refactoring software, architectures, and projects in crisis. 1998.

BUSCHMANN, F. 1996. Pattern-Oriented Software Architecture. 1996.

Casadomo Soluciones SL, Servicios de una casa inteligente, 2006, http://www.casadomo.com

CASTRO RUZ, F. Discurso. Conferencia de Naciones Unidas sobre Medio Ambiente y Desarrollo. Río de Janeiro, Brasil, 1992.

CEDEÑO POZO, A. Módulos de adquisición y análisis para la interacción con dispositivos de campo en un SCADA. Tesis en opción al título de Ingeniero. Universidad de Ciencias Informáticas, Ciudad de La Habana. 2009.

COMINO LUCENDO, D. Cuatro Centros de ocio para el hogar digital. 2006, 66-72 p.

---. Disfruta de un hogar digital. Madrid, IDG Communications, 2005, nº p. 68-72.

DAVIS, A. 1995. Principles of Software Development. [Digital] 1995. del Toro Ríos, José Carlos y GONZÁLEZ BRITO, H.. 2008. Documento Visión. Proyecto ERP-Cuba. [Digital] La Habana: Universidad de las Ciencias Informáticas, 2008.

ESTRADA RODRÍGUEZ, J., GUERRERO CHACÓN, Y. Desarrollo de una librería de funciones para el control de un robot manipulador. Tesis en opción al título de Ingeniero. Universidad de Ciencias Informáticas, Ciudad de La Habana. 2008.

FLANAGAN, D. Java en pocas palabras. Editado por: México, M.-H. I. 1998, Disponible en: http://bibliodoc.uci.cu/pdf/reg01329.pdf. 610.

FRANKEL, D. S. Model Driven Architecture Applying MDA to Enterprise Computing. Editado por: Indianapolis, W. P. 2003, Disponible en: http://bibliodoc.uci.cu/pdf/0-471-31920-1.pdf. 354.

FUENTS VIÑAS, I. GUZMAN RODRÍGUEZ, E. Módulo de Visitas para el sistema de Control de Acceso. Tesis en opción al título de Ingeniero. Universidad de Ciencias Informáticas, Ciudad de La Habana. 2009.

GARLAN, M. 1996. Software Architecture: Prespectives on an emerging discipline. 1996.

Gestión, Centro de Soluciones de. 2008. Modelo de desarrollo orientado a componentes. [Digital] La Habana: Universidad de las Ciencias Informáticas, 2008.

GRIMÁN, A., PÉREZ, M., MENDOZA, L. Estudio de la influencia de mecanismos arquitectónicos en la calidad del software. Caracas, Universidad Simón Bolívar, 2005, Disponible en: http://bibliodoc.uci.cu/pdf/evaluacion.pdf. 7.

HERNÁNDEZ ORALLO, E.. 2002. El Lenguaje unificado de Modelado (UML). [En línea] 17 de enero de 2010. http://www.acta.es/articulos_mf/26067.pdf.

JACOBSON, I., BOOCH, G. y RUMBAUGH, J. El Proceso Unificado de Desarrollo de Software. 1a. ed. Madrid: Addison Wesley, 2000. 464 p. ISBN: 8478290362

KUCHANA, P. Software architecture design patterns in Java. Editado por: Boca Raton, A. P. 2004, Disponible en: http://bibliodoc.uci.cu/pdf/0-8493-2142-5.pdf. 476.

LARMAN, C. UML y Patrones. Introducción al análisis y diseño orientado a objetos. 1a. ed. México: Prentice Hall, 1999. 536 p.ISBN: 9701702611

Linux journal. Embedding the db4o object-oriented database, Universidad de La Rioja, ISSN 1075-3583, Nº 142, 2006

LÓPEZ, A. Java, la programación del futuro. Editado por: Buenos Aires, M. E. 1997, Disponible en: http://bibliodoc.uci.cu/pdf/reg00542.pdf. 287.

MARTÍN, E., MARCELO, JUAN F. Ordenadores y domótica: hogares inteligentes Madrid, IDG Communications, 2003, nº p. 218-225.

MILLAN TEJEDOR, R. Domótica: edificios inteligentes. Barcelona, Anaya Multimedia S.A., 2004. 17

MOLDES, T. Java 2. J2SE 1.4 (Guía Práctica para usuarios). Editado por: Madrid, A. M. S. A. 2003, Disponible en: http://bibliodoc.uci.cu/pdf/reg02485.pdf. 352.

Myers, Glen. 1979. The Art of Software Testing. 1979. ISBN-10: 0471078786.

PRESSMAN, R. S. Ingeniería de Software. Un enfoque práctico. 5. La Habana, 2005. p.289

REYES PADILLA, G. Domótica, 2004.http://www.monografias.com/trabajos35/domotica/domotica.shtml

RODRÍGUEZ PELÁEZ, Y. Sistema para la informatización del proceso del Control de Estancias en Hoteles para la Dirección de Migración y Fronteras de la República Bolivariana de Venezuela Tesis en opción al título de Ingeniero. Universidad de Ciencias Informáticas, Ciudad de La Habana. 2009.

RUBINOS CARVAJAL, A. Subsistema de Seguridad para el SCADA Nacional Guardián del ALBA. Tesis en opción al título de Ingeniero. Universidad de Ciencias Informáticas, Ciudad de La Habana. 2009.

SCHMULLER, J. Aprendiendo UML en 24 Horas. 2000. 448 p.

SEDGEWICK, R. Algorithms in Java. Editado por: Boston, A.-W. 2003, Disponible en: http://bibliodoc.uci.cu/pdf/reg03749.pdf. 721.

STEPHEN, R. D. Aprenda Java ya. Editado por: Madrid, M.-H. 1997. 342

TEXEIRA LÓPEZ, M. Jini – Sistemas Distribuidos en Java. http://ciberia.ya.com/pxai/ma.html, 2002

TRIANA GONZÁLEZ, R., Domótica Soluciones Integrales SL. Introducción y tecnologías de la Domótica, 2008, http://www.domotica.net

WAITE, M. Data Structures and Algorithms in Java. 1998, nº Disponible en: http://bibliodoc.uci.cu/pdf/1571690956.pdf. 526.

WENDY BOGGS, M. B. UML whit Rational Rose 2002. 2001.

WIKIPEDIA (2008). *World Wide Web*. [fecha de consulta: 10-diciembre-2009]. Disponible en: http://es.wikipedia.org/wiki/World_Wide_Web

ZUKOWSKI, J. Java2 J2SE 1.4. Anaya Multimedia S. A., 2003.

---. Libro blanco de Java. http://java.sun.com/people/jag/OriginalJavaWhitepaper.pdf. 2001.

---. Programación. Java 2 J2SE 1.4 V-I-II-III. Editado por: Varela, F. 2007. 1008

GLOSARIO DE TÉRMINOS

Actuadores: Los actuadores son los dispositivos capaces de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema (apagar/encender, subir/bajar, apertura/cierre, etc.).

Aplicación informática: programas informáticos diseñados como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajos.

Arquitectura: Conjunto de elementos estructurales significativos acerca de la organización de un sistema software, la selección de los elementos estructurales que representan el sistema, y las interfaces entre ellos, junto con su comportamiento, tal y como se especifica en las colaboraciones entre esos elementos, la composición de estos elementos estructurales y de comportamiento en subsistemas progresivamente mayores, y al estilo arquitectónico que guía esta organización: estos elementos y sus interfaces, sus colaboraciones y su composición.

Bus: Es el medio de transmisión que transporta la información entre los distintos dispositivos y puede ser por un cableado propio, por la redes de otros sistemas (red eléctrica, red telefónica, red de datos) o de forma inalámbrica.

Caso de uso: es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

Controlador: dispositivo que gestiona el sistema según la programación y la información que tiene o que recibe. Puede haber uno o varios controladores distribuidos por el sistema.

Domótica: se refiere a la **automatización** y **control** (encendido / apagado, apertura / cierre y regulación) de aparatos y sistemas de instalaciones eléctricas y electrotécnicas (iluminación, climatización, persianas y toldos, puertas y ventanas motorizados, etc.) de forma centralizada y/o remota.

GoF: *Gang of four*, grupo compuesto por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides, publicaron el libro *Design Patterns* en el que se recogían 23 patrones diseño comunes.

Inteligencia Artificial: Rama de la Informática que desarrolla procesos que imitan la inteligencia de los seres vivos. La principal aplicación de esta ciencia es la creación de máquinas para la automatización de tareas que requieran un comportamiento inteligente.

Interfaz: son los dispositivos (pantallas, móviles, Internet, teclados) y los formatos (binario, audio, gráficos) en que se muestra la información del sistema para los usuarios y donde los mismos pueden interactuar con el sistema.

Latencia: Suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Módulo: En programación, un módulo es un software que agrupa un conjunto de subprogramas y estructuras de datos. Los módulos son unidades que pueden ser compiladas por separado y los hace reusables y permite que múltiples programadores trabajen en diferentes módulos en forma simultánea, produciendo ahorro en los tiempos de desarrollo. Los módulos promueven la modularidad y el encapsulamiento, pudiendo generar programas complejos de fácil comprensión. Puede tomarse como sinónimo de subrutina o de unidad de software, aunque este último es más abarcador.

Requisito Funcional: Requisito que especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas. Requisito que especifica comportamiento de entrada/salida de un sistema.

Requisito no Funcional: Requisito que especifica propiedades del sistema, como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, extensibilidad o fiabilidad. Requisito que especifica restricciones físicas sobre un requisito funcional.

Requisito: Condición o capacidad, necesidad o deseo que debe cumplir un sistema.

Sensores: Los sensores son los dispositivos que monitorean el entorno captando información y transmitiéndola al sistema, por ejemplo sensores de movimiento, agua, gas, humo, temperatura, viento, humedad, lluvia, iluminación, etc.

Tecnología: Es un concepto amplio que abarca un conjunto de técnicas, conocimientos y procesos, que sirven para el diseño y construcción de objetos para satisfacer necesidades humanas. La palabra tecnología proviene del griego tekne (técnica, oficio) y logos (ciencia, conocimiento). La tecnología puede referirse a objetos que usa la humanidad (como máquinas, utensilios, hardware), pero también abarca sistemas, métodos de organización y técnicas.