ESCUELA POLITECNICA DEL EJÉRCITO

DPTO. DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMATICA

DISEÑO Y DESARROLLO DE UN PROTOTIPO DE CONTROL MEDIANTE SMS PARA CASAS INTELIGENTES

Previa a la obtención del Título de:
INGENIERO EN SISTEMAS E INFORMÁTICA

DIRECTOR: INGENIERO DIEGO MARCILLO
CODIRECTOR: INGENIERO JAIME ANDRANGO

POR: PAÚL FERNANDO INCA REA

SANGOLQUI, 10 DE MAYO DEL 2012

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el Sr. PAÚL FERNANDO INCA REA como requerimiento parcial a la obtención del título de INGENIERO EN SISTEMAS E INFORMÁTICA.

10 de Mayo de 2012

Ing. Diego Marcillo Parra

DEDICATORIA

A mis padres queridos Lida y Benjamín, quienes con una vida de entrega y sacrificio han sabido educarme bajo la guía del amor y el respeto.

Paúl Fernando Inca Rea

AGRADECIMIENTOS

A mi Dios, por bendecirme con una familia amorosa, comprensiva y emprendedora.

A mis padres, quienes son los pilares fundamentales de mi vida, ellos quienes siempre me han apoyado en todo y han sabido guiarme con su ejemplo.

A mis hermanas, mis cómplices de toda la vida.

A mis amigos, aquellos doteros, estudiosos, farreros, bieleros todos aquellos amigos de verdad con los que he compartido momentos grandiosos en la vida.

Quiero rendir un agradecimiento especial a mis ingenieros de la universidad, quienes fueron mis amigos y profesores, aquellos que en su respectivo momento supieron educarme tanto en el campo científico, cultural y humano.

Paúl Fernando Inca Rea

INDICE DE CONTENIDOS

Contenido

LISTADO DE TABLAS

LISTADO DE FIGURAS

RESUMEN

La evolución del mercado inmobiliario ha incorporado nuevas normativas y reglamentos, así como nuevas exigencias lo que provoca que las viviendas actuales tengan un ambiente tecnológico e interactivo con el usuario. Tomando en cuenta lo antes mencionado se pensó en dar un aporte para la integración de la tecnología con la arquitectura en los hogares ecuatorianos, para lo cual se desarrolló el presente proyecto.

El proyecto se enfocó en el control de la vivienda, como es la manipulación de equipos y alertas por parte de los habitantes del hogar. Para el desarrollo del mismo se utilizó la metodología XP, el lenguaje de programación JAVA y como entorno integrado de desarrollo Netbeans.

El proyecto está conformado por varios módulos, los cuales han sido desarrollados de acuerdo al análisis de requerimientos, además cuenta una aplicación gráfica la cual se instala en el celular para poder enviar órdenes a la vivienda de manera remota, la comunicación se realiza mediante mensajes de texto que son enviados desde el celular al servidor sin que el mensaje sea visible para el usuario.

CAPÍTULO 1

INTRODUCCIÓN

1.1. Antecedentes

La evolución del mercado inmobiliario con nuevas normativas, reglamentos y una competencia en aumento demanda cada vez más y mejores instalaciones y equipamientos en las viviendas de nueva construcción. A la vez, los compradores exigen viviendas más confortables, seguras, con mejores comunicaciones y adaptadas tecnológicamente al momento actual.

La adaptación a las nuevas tecnologías que se encuentran en el hogar digital, proporciona a los futuros inquilinos funciones y servicios que facilitan la gestión de la vivienda, aumentan la seguridad, incrementan el confort, mejoran las telecomunicaciones, ahorran energía y tiempo a la hora de controlar su hogar.

En Ecuador, el sector de la construcción está desarrollando sus proyectos dentro de los esquemas tecnológicos más confiables. Ahora estos inmuebles son adaptables y armonizan con el ambiente.

La intención del sector inmobiliario es construir viviendas que cuenten con nuevos beneficios para sus habitantes. Entre las ventajas que brinda la domótica está la reducción de costos de operación y mantenimiento, esto permite que las nuevas viviendas tengan una plusvalía más alta en comparación con los diseños actuales

1.2. Planteamiento del problema

Actualmente son pocas las personas que conocen acerca de los progresos relacionados con la tecnología y la arquitectura, por ende desconoce los beneficios que prestan los hogares digitales. Uno de estos beneficios sería el poder tener un control externo sobre la vivienda, lo cual permitiría una mejor administración de la misma además de proporcionar seguridad y confort.

La falta de dispositivos y servicios para que los propietarios de las viviendas puedan tener control sobre las mismas, permite el planteamiento del desarrollo de nuevos productos para tal evento.

Tomando en cuenta lo antes mencionado se pensó en dar un aporte para la integración de la tecnología con la arquitectura en los hogares ecuatorianos, mediante el desarrollo de un sistema de control vía SMS para los hogares.

1.3. Justificación e importancia

El proyecto permite utilizar la tecnología en favor de las necesidades cotidianas, en el tema de la automatización de viviendas, así como aportar soluciones de control.

Dada la convivencia de la sociedad ecuatoriana con el servicio de mensajería instantánea, se ha pensado en aprovechar tal uso y poder permitir que los dueños de los hogares puedan controlar sus viviendas mediante SMS[1].

Los servicios a controlar podrían ser los siguientes:

- Control de encendido de luces.
- Estados de puertas/ventanas (abierta-cerrado).
- Abertura de Puertas
- Recordatorios Internos (Información de actos)

El sistema se desarrollará en el lenguaje de programación Java, lo cual permite una mejor compatibilidad a la hora de instalar el sistema en los equipos del usuario.

[1] El mensaje SMS consiste de una cadena alfanumérica de hasta 160 caracteres de 7 bits, cuyo encapsulado corresponde a una serie de parámetros, los cuales se emplean para enviar y recibir mensajes de texto normales, pero estos también permiten incluir otro tipo de contenidos, como ser darle un formato especial a los mensajes y encadenar varios mensajes de texto para ganar en longitud. El sistema fue originariamente diseñado como parte estándar de la telefonía móvil digital GSM

1.4. Objetivos

1.4.1. Objetivo general

Realizar el diseño y desarrollo de un prototipo de control mediante SMS para casas inteligentes utilizando herramientas open source.

1.4.2. Objetivos específicos

- Realizar el levantamiento y análisis de requerimientos para desarrollar cada uno de las etapas en el proyecto.

- Desarrollar una aplicación de escritorio utilizando herramientas y editores open source para la administración y envío de mensajes SMS.

- Desarrollar el software de interfaz gráfica para ser ejecutada en el dispositivo celular del usuario que permitirá el control de la vivienda.

- Realizar pruebas de implementación y ajustes para alcanzar un correcto funcionamiento.

1.5. Alcance

El alcance del proyecto dispone de las siguientes características:

- El sistema de control de casa inteligente será capaz de activar o desactivar las funciones específicas controladas vía remota mediante un módulo SMS hardware.
- Desde cualquier teléfono celular se enviará un mensaje al módulo GSM[2]. Se recibe el mensaje y este es enviado al computador central que se comunica con el hardware de control para apagar y prender luces, ó abrir y cerrar puertas.
- Proporcionar una aplicación en J2ME para facilitar su uso mediante el celular.

[2] Sistema global para comunicaciones móviles (GSM) es un estándar mundialmente aceptado para la comunicación celular digital. GSM es el nombre de un grupo de normalización establecido en 1982 para crear un teléfono móvil estándar común europeo que formular las especificaciones de un sistema paneuropeo de radio móvil celular sistema operativo a 900 MHz. Se estima que muchos países fuera de Europa se unirán a la asociación GSM

- Adicionalmente desde la computadora se envían mensajes de alerta de SMS del usuario para avisar la llegada de alguien a la casa.
- En su primera etapa se diseñara el software tomando como guía una maqueta para luego realizar el desarrollo del software más el módulo GSM con la conexión ENFORA.
- Desarrollar un sistema de información, basado en una base de datos open source, que realice las tareas de monitoreo y almacenamiento de los eventos que se generan en los accesos, para brindar seguridad a la casa inteligente.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Herramientas de desarrollo

2.1.1. Netbeans 6.9

2.1.1.1. Definición

Netbeans es un IDE multilenguaje completo y modular que tiene soporte para Java SE, Java EE, J2ME, además consta de una gran cantidad de módulos[3] de terceros (plugins).

Tiene un entorno de desarrollo integrado (IDE) para desarrollar bajo la plataforma JAVA, además de ser gratis y open source, consta de una gran comunidad de usuarios y desarrolladores los cuales siempre aportan con nuevas ideas y ayudas online.

"NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos"[4]

2.1.1.2. Características

Netbeans es una base para desarrollar aplicaciones complejas con un enfoque modular y pensando en características como la extensibilidad y la escalabilidad.
El IDE de Netbeans es una muestra del tipo de aplicaciones que se pueden desarrollar utilizando la Plataforma, ya que el mismo está construido sobre ella.

A continuación se resaltan algunas de las características de Netbeans:

- Framework para la creación de interfaces de usuario.
- El editor de datos de Netbeans IDE.
- Interfaz de usuario para la personalización de la aplicación.

[3] Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las API(Aplication Programming Interface) de NetBeans y un archivo especial (manifest file) que lo identifica como módulo
[4] Referencias tomadas de http://es.wikipedia.org/wiki/NetBeans

- Framework para la creación de asistentes (Wizards).
- Sistema de datos que permite obtener información de diferentes orígenes de datos (FTP, CVS, Base de Datos).
- Internacionalización.
- Ayudas del sistema.
- Ayudas contextuales del sistema.
- Organización de la aplicación basada en estándares y patrones estructurales y de diseño.

Rendimiento optimo en tiempo de ejecución y optimización de recursos.

2.1.1.3. Ventajas

Las ventajas de usar Netbeans son muchas a continuación se mencionan algunas:

- Tener el respaldo de una empresa tan grande y seria como es Oracle.
- Poder usar las licencias open source para realizar mejoras futuras.
- Tener un respaldo online por parte de otros programadores.
- Tener un IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles).

Modularidad. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones.

2.1.2. MySql

2.1.2.1. Definición

MySQL es uno de los mejores sistemas de administración de base de datos[5] actualmente, teniendo como respaldo ser Open Source, y contar con mejoras constantes de parte de desarrolladores a nivel mundial.

[5] http://www.scribd.com/doc/13059949/Base-de-Datos.- Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

"MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009, desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C[6].

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código."[7]
El tener diseñado una base de datos no es suficiente para el manejo de la información para lo cual se utiliza MySQL Server para agregar, acceder a y procesar datos guardados en un servidor.
"El sistema de base de datos operacional MySQL es hoy en día uno de los más importantes en lo que hace al diseño y programación de base de datos de tipo relacional. Cuenta con millones de aplicaciones y aparece en el mundo informático como una de las más utilizadas por usuarios del medio. El programa MySQL se usa como servidor a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo"[8]

MySQL se basa en la administración relacional de bases de datos, el cual archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo lo cual permite velocidad y flexibilidad.
Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

[6] ANSI C es un estándar publicado por el Instituto Nacional Estadounidense de Estándares (ANSI), para el lenguaje de programación C. Se recomienda a los desarrolladores de software en C que cumplan con los requisitos descritos en el documento para facilitar así la portabilidad del código. Tomado de http://es.wikipedia.org/wiki/ANSI_C
[7] Tomado de http://es.wikipedia.org/wiki/MySQL
[8] Referencia tomada de http://www.definicionabc.com/tecnologia/mysql.php

2.1.2.2. Características

A continuación se mencionan algunas de las características de MySql:

- Interioridades y portabilidad.
- Escrito en C y en C++.
- Probado con un amplio rango de compiladores diferentes.
- Funciona en diferentes plataformas.
- Usa GNU Automake, Autoconf, y Libtool para portabilidad.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente multiple CPUs si están disponibles.
- Proporciona sistemas de almacenamientos transaccionales y no transaccionales.
- Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.
- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.
- Soporta diferente tipos de datos en sus columnas como: enteros con/sin signo de 1, 2, 3, 4, y 8 bytes de longitud, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, y tipos espaciales OpenGIS[9].

[9] Una geometría simple es definida por la especificación OpenGIS como una entidad que consta de atributos espaciales y no espaciales. Los atributos espaciales son valores geométricos, y las

- Soporte completo para operadores y funciones en las cláusulas de consultas SELECT y WHERE.
- Soporte completo para las cláusulas SQL GROUP BY y ORDER BY. Soporte de funciones de agrupación (COUNT(), COUNT(DISTINCT), AVG(), STD(), SUM(), MAX(), MIN(), y GROUP_CONCAT()).
- Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN cumpliendo estándares de sintaxis SQL y ODBC.
- Soporte para alias en tablas y columnas como lo requiere el estándar SQL.
- DELETE, INSERT, REPLACE, y UPDATE devuelven el número de filas que han cambiado. Es posible devolver el número de filas que serían afectadas usando un flag al conectar con el servidor.
- El comando específico de MySQL SHOW puede usarse para obtener información acerca de la base de datos, el motor de base de datos, tablas e índices. El comando EXPLAIN puede usarse para determinar cómo el optimizador resuelve una consulta.
- Los nombres de funciones no colisionan con los nombres de tabla o columna. Por ejemplo, ABS es un nombre válido de columna. La única restricción es que para una llamada a una función, no se permiten espacios entre el nombre de función y el '(' a continuación. Puede mezclar tablas de distintas bases de datos en la misma consulta.
- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verficación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.
- Soporte a grandes bases de datos. Usamos MySQL Server con bases de datos que contienen 50 millones de registros. También conocemos a usuarios que usan MySQL Server con 60.000 tablas y cerca de 5.000.000.000.000 de registros.

Se permiten hasta 64 índices por tabla. Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes.Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

---

geométricas simple están basadas en geométricas 2D con interpolación lineal entre vértices. Tomado de : Especificaciones OpenGIS

2.1.2.3. Ventajas

A continuación se indican algunas ventajas de MySql, por el cual es usado para el desarrollo de proyectos:

- Capacidad para funcionar con varios sistemas operativos.
- Fácil de escalar en los productos básicos de hardware.
- MySQL Proxy.
- Múltiples motores de almacenamiento.
- Bloqueo a nivel de fila.
- Servicios de carga de datos, de alta velocidad.
- Copia de seguridad en línea con el punto en el tiempo de recuperación.
- Reinicio automático / recuperaciones.
- Servidor forzado de integridad referencial.
- Controladores (ODBC, JDBC,. NET, PHP, etc).
- Puntos de vista actualizable.
- Soporte de Oracle.

2.1.3. Lenguaje de desarrollo

2.1.3.1. JAVA

2.1.3.2. Definición

"Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode[10], aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es

[10] Tomado de http://es.wikipedia.org/wiki/Bytecode El bytecode es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina

normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible."[11]

"Java es un lenguaje de programación y la primera plataforma informática creada por Sun Microsystems en 1995. Es la tecnología subyacente que permite el uso de programas punteros, como herramientas, juegos y aplicaciones de negocios. Java se ejecuta en más de 850 millones de ordenadores personales de todo el mundo y en miles de millones de dispositivos, como dispositivos móviles y aparatos de televisión."[12]

2.1.3.3. Características

En relación a Java se puede señalar algunas características a la hora del desarrollo del proyecto como son:

- Es de licencia gratuita.
- Es de programación orientado a objetos.
- Permite un fácil desarrollo en aplicaciones móviles.
- Soporta múltiples sistemas operativos.
- Es robusto pues permite buscar errores en tiempo real como es a la hora de compilar y ejecutar el programa.
- Permite el polimorfismo.

Arquitectura neutral al ser Java parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará.

2.1.4. SLQ

2.1.4.1. Definición

"El lenguaje de consulta estructurado o SQL (por sus siglas en inglés structured query language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar de una

[11] Tomado de http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29
[12] Tomado de http://www.java.com/es/download/faq/whatis_java.xml

forma sencilla información de interés de una base de datos, así como también hacer cambios sobre ella."[13]

SQL consta de varias partes las cuales serán mencionadas a continuación:

- Lenguaje de definición de datos (DDL): Proporciona órdenes para definir esquemas de relación, eliminar relaciones, crear índices y modificar esquemas de relación.[14]

Lenguaje de manipulación de datos interactivos (DML): incluye un leguaje de consultas que permite rescatar datos de las relaciones. También incluye órdenes para insertar, suprimir y modificar registros.

2.2. Plataforma de desarrollo

2.2.1. Hardware

Los elementos de hardware que se utilizó son los siguientes:

1. Para el desarrollo del proyecto se utilizara un computador que dispone la empresa Softeratronic , que posee las siguientes características:

   - Procesador: Intel Core 2 Duo de 2.2.
   - Memoria: 4 GB.
   - Tamaño en disco: 250 GB.

2. Como segundo elemento a usar es un teléfono celular Sony Ericsson K-850i, propiedad de Paul Inca realizador de la tesis, a continuación se mencionan algunas características:

Redes

- GSM /GPRS/EDGE 850/900/1800/1900.
- UMTS /HSDPA 850/1900/2100.

[13] Tomado de http://es.wikipedia.org/wiki/SQL
[14] Tomado de http://www.monografias.com/trabajos11/prosq/prosq.shtml

Java

- Añade aplicaciones y juegos extra a tu teléfono[15].

3. Se requiere también el uso de un modem usb para conexión de internet sea este de Movistar , para el caso usaremos el siguiente modem:

Huawei E1756c

Con las siguientes características: [16]

- HSUPA/HSDPA/UMTS 2100/1900/850 MHz.
- EDGE/GPRS/GSM 1900/1800/900/800 MHz.
- HSPA equalizer.
- SMS.
- Plug and Play.
- USB.

2.2.2. Software

Para el desarrollo del proyecto se utilizó:

1. Herramientas de desarrollo
   - Netbeans.
   - MySql.

2. Lenguaje de Desarrollo
   - Java.
   - Sql.

.

[15] Tomado de: http://www.sonyericsson.com/cws/products/mobilephones/overview/k850i?lc=es&cc=es#view=features_specifications

[16] Tomado de: http://tienda.movistar.com.ec/prod.php?pid=202114

2.3. Tecnología de dispositivos móviles

2.3.1. Celulares concepto

"Dispositivo electrónico que permite realizar múltiples operaciones de forma inalámbrica en cualquier lugar donde tenga señal. Entre las múltiples operaciones se incluyen la realización de llamadas telefónicas, navegación por internet, envío de mensajes de texto (SMS), captura de fotos y sonidos, reloj, agenda, realización de pagos, etc."[17]

2.3.2. Generaciones de celulares

2.3.2.1. G-0 Generación 0

Su aparición se dió en la época de la segunda guerra mundial, la compañía Motorola lanzó el Handie Talkie H12-16, el cual permitía la comunicación a distancia entre las tropas, un dispositivo basado en la transmisión mediante ondas de radio que, a pesar de trabajar por aquel entonces con un espectro de 550 MHz aproximadamente, supuso una revolución de enormes proporciones.

Esta tecnología fue aprovechada a partir de los años 50 y 60 para crear una gran variedad de aparatos de radio y de comunicación a distancia (los tradicionales Walkie-Talkies), utilizados sobre todo por servicios públicos tales como taxis, ambulancias o bomberos.

Aunque realmente estos dispositivos no pueden ser considerados como teléfonos móviles, la implementación de los primeros supuso el comienzo de la evolución hacia los dispositivos que se conoce en la actualidad.

Los primeros estándares más utilizados, en los que fundamentó esta generación fueron:

• Estándar PTT (Push To Talk): Pulsar para Hablar.

• Estándar IMTS (Improved Mobile Telefone System): el Sistema de Telefonía Móvil Mejorado.

[17] Tomado de: http://www.alegsa.com.ar/Dic/celular.php

2.3.2.2. 1-G Móviles de primera generación

Surge a partir de 1973 y con un tamaño y peso inmanejable, los móviles de primera generación funcionaban de manera analógica, es decir que la transmisión y recepción de datos se apoyaba sobre un conjunto de ondas de radio que cambiaban de modo continuo.

El hecho de que fueran analógicos traía consigo una serie de inconvenientes, tales como que solo podían ser utilizados para la transmisión de voz o su baja seguridad, la cual hacia posible a una persona escuchar llamadas ajenas con un simple sintonizador de radio o, incluso hacer uso de las frecuencias cargando el importe de las llamadas a otras personas.

A pesar de todo, esta fue la primera generación considerada realmente como de teléfonos móviles.

Estándares más utilizados:

• NMT: Nordic Mobile Telephone

• AMPS: Advaced Mobile Phone System

2.3.2.3. 2-G Segunda generación

Al contrario de lo que pasa en otras generaciones, la denominada “segunda generación” no es un estándar concreto, sino que marca el paso de la telefonía analógica a la digital, que permitió, mediante la introducción de una serie de protocolos, la mejora del manejo de llamadas, más enlaces simultáneos en el mismo ancho de banda y la integración de otros servicios adicionales al de la voz, de entre los que destaca el Servicio de Mensajes Cortos (Short Message Service).

Estándares más utilizados:

• GSM: Global System for Mobile Communications - Sistema Global para Comunicaciones Móviles.

• CDMA: Code Division Multiple Access - Acceso Múltiple por División de Código.

• GPRS: General Packet Radio Service - Servicio General de Radio por paquetes.

2.3.2.4. 3-G Tercera generación

El año 2001 fue un año revolucionario en el ámbito de la telefonía móvil ya que supuso la aparición de los primeros celulares que incorporaban pantalla LCD a color, hecho que abría un inmenso abanico de posibilidades en cuanto a adaptación de nuevas funciones se refiere.

Así, pronto el usuario pudo asistir al nacimiento de dispositivos que se creían como mínimo futuristas tales como móviles con cámara fotográfica digital, posibilidad de grabar videos y mandarlos con un sistema de mensajería instantánea evolucionado, juegos 3d, sonido Mp3 o poder mantener conversaciones por videoconferencia gracias a una tasa de transferencia de datos más que aceptable.

Todo este conjunto de nuevos servicios integrados en el terminal junto con un nuevo estándar dieron lugar a la denominada hoy en día "tercera generación de móviles" o móviles 3G.

Estándares más utilizados:

• UMTS: Universal Mobile Telecommunications System - Servicios Universales de Comunicaciones Móviles.

2.3.2.5. 4-G Cuarta generación

La 4G está basada completamente en el protocolo IP, siendo un sistema de sistemas y una red de redes, que se alcanza gracias a la convergencia entre las redes de cables e inalámbricas.

Esta tecnología podrá ser usada por modems inalámbricos, celulares inteligentes y otros dispositivos móviles. La principal diferencia con las generaciones predecesoras será la capacidad para proveer velocidades de acceso mayores de 100 Mbps en movimiento y 1 Gbps en reposo, manteniendo una calidad de servicio (QoS) de punta a punta de alta seguridad que permitirá ofrecer servicios de cualquier clase en cualquier momento, en cualquier lugar, con el mínimo coste posible.

2.3.3. Red de telefonía celular

La red de telefonía celular está compuesta por dos grandes elementos las cuales son:

- Estaciones base.- son las encargadas de transmitir y recibir la señal.
- Centrales de conmutación.- son las que permiten la conexión entre dos terminales concretos.

Funciones del equipo de conmutación:

- Identificar al abonado solicitante.
- Analizar la información de selección
- De acuerdo a esta información, seleccionar la vía o canal a utilizar.
- Iniciar la central subsiguiente.
- Transferirle la información de selección.
- Investigar el estado libre/ ocupado del abonado solicitante.
- Informar al abonado A/B lo que le corresponde.
- Establecer /liberar el enlace.
- Supervisar la conexión.
- Y liberar los caminos establecidos cuando la comunicación haya finalizado.

2.3.3.1. Tipos de redes

Al señalar los tipos de redes utilizadas para la conexión de la telefonía celular nos enfocaremos en las dos siguientes:

- **Red analógica (TMA)**: La comunicación se produce mediante el envió de la información sobre la señal analógica[18].

Esta clase de redes operaba en la banda de frecuencias de los 450 Mhz en un principio y posteriormente en los 900 MHz, este tipo de red quedo en desuso a partir del 31 de diciembre del 2003 en España pero aun se tiene este servicio en Estados Unidos.

[18] Una señal analógica es un tipo de señal generada por algún tipo de fenómeno electromagnético y que es representable por una función matemática continua en la que es variable su amplitud y periodo (representando un dato de información) en función del tiempo.
Tomado de http://es.wikipedia.org/wiki/Se%C3%B1al_anal%C3%B3gica

- **Red digital:** En esta red la comunicación se realiza mediante señales digitales, lo que permite optimizar tanto el aprovechamiento de las bandas de radiofrecuencia como la calidad de transmisión. Su exponente más significativo en el ámbito público es el estandar GSM y su tercera generación, UMTS[19]. Funciona en las bandas de 850/900 y 1800/1900 MHz. Aparte de los ya mencionados tenemos también otro estándar digital, presente en América y Asia, denominado CDMA[20].

En el ámbito privado y de servicios de emergencias como policía, bomberos y servicios de ambulancias se utilizan los estóndares Tetrapol y Terrestrial Trunked Radio (TETRA) en diferentes bandas de frecuencia.

2.3.3.2. Estándares

2.3.3.2.1. G-0 Generación 0

Estandar PTT (Push to talk): como su nombre lo indica "Pulsar para hablar", se trata de un estándar que posibilitaba la transmisión y recepción de voz utiliando el mismo ancho de banda. Para permitir esto PTT discriminaba entre transmisión o recepción pulsando un botón, el cual era pulsado para enviar la voz y al soltar recibia la voz.

[19] Sistema Universal de Telecomunicaciones Móviles (Universal Mobile Telecommunications System - UMTS) es una de las tecnologías usadas por los móviles de tercera generación (3G, también llamado W-CDMA), sucesora de GSM, debido a que la tecnología GSM propiamente dicha no podía seguir un camino evolutivo para llegar a brindar servicios considerados de Tercera Generación.
Sus tres grandes características son las capacidades multimedia, una velocidad de acceso a Internet elevada, la cual también le permite transmitir audio y video en tiempo real; y una transmisión de voz con calidad equiparable a la de las redes fijas. Además, dispone de una variedad de servicios muy extensa
Tomado de http://es.wikipedia.org/wiki/Universal_Mobile_Telecommunications_System

[20] La multiplexación por división de código, acceso múltiple por división de código o CDMA (del inglés Code Division Multiple Access) es un término genérico para varios métodos de multiplexación o control de acceso al medio basados en la tecnología de espectro expandido.
Tomado de
http://es.wikipedia.org/wiki/Acceso_m%C3%BAltiple_por_divisi%C3%B3n_de_c%C3%B3digo

Improved Mobile Telephone System (Sistema de Telefonia Movil Mejorado): es un sistema de comunicación móvil analógico que fue implementado en los años 60, con muy poco éxito.

IMTS usaba un transmisor de muy alta potencia, para lo cual estos nuevos sistemas se tenían que implantar muy lejos unos de otros para evitar interferencias lo cual provoco su fracaso.

A diferencia de los PTT este nuevo estándar no emitía y recibía en la misma banda de frecuencias, por lo cual eliminaba la necesidad de pulsar un botón para alterar la direccionalidad de la comunicación. IMTS puso a disposición de los clientes 23 canales espaciados entre 150 y 450 MHz.

2.3.3.2.2. G-1 Generación 1

NMT (Nordic Mobile Telephone): Las celdas de las redes NMT son de igual o mayor tamaño que las de GSM: de 2 a 30 km, en vez de cinco.

Cuanto menor la celda, más usuarios pueden ser atendidos, lo que hace que algunas celdas sean voluntariamente pequeñas en zonas densamente pobladas.

NMT es un sistema full-dúplex, por lo que es posible transmitir y recibir al mismo tiempo.
Las versiones para automóvil usan potencias de 15 watts (NMT-450) o bien 6 watts (NMT-900); los teléfonos portátiles son de hasta 1 watt.

NMT no tenía cifrado de las comunicaciones, lo que era una desventaja; cualquier persona equipada de un scanner podía escuchar las conversaciones de los clientes.

2.3.3.2.3. G-2 Generación 2

GSM (Global System for Mobile Communications): Quizás se trate del protocolo más característico de la 2G, ya que además se trata de un estándar desarrollado por y para todas las regiones del mundo.

Su funcionamiento se sustenta sobre una compleja base de canales lógicos que permiten tanto la transmisión de voz como de datos.

El rango de frecuencias utilizado varía, debido sobre todo al país del que estemos hablando, dando lugar a distintos tipos de protocolos GSM:

- GSM-1800: Sistema celular GSM que funciona en la banda de frecuencias 1800 MHz. Utilizado principalmente en zonas urbanas de Europa.
- GSM-1900: Sistema celular GSM que funciona en la banda de frecuencias 1900 MHz. Utilizado principalmente en zonas urbanas de Estados
- GSM-900: red celular digital que opera en el rango de 900 MHz.

El GSM, se puede dedicar tanto a voz como a datos. Una llamada de voz utiliza un codificador GSM específico a velocidad total de 13Kbits/s, posteriormente se desarrolló un códec a velocidad mitad de 6,5 kbits/s que permitirá duplicar la capacidad de los canales TCH, se denomina FR (Full Rate) y HR (Half Rate).

2.3.3.2.4. G-3 Generación 3

Acceso Múltiple por División de Código de Banda Ancha, que no es más que una interfaz de herencia militar para UMTS, que se caracteriza por la utilización de una banda más ancha que su hermano pequeño CDMA, lo que supone una serie de ventajas adicionales tales como:

- Velocidades de transmisión mejoradas (hasta 2 Mbps).
- Menos interferencias y, por tanto, una voz de calidad mayor.
- Cobertura a nivel mundial ya sea de modo terrestre o a través de satélite, dando como resultado una comunicación sin fisuras aún estando en movimiento.
- Posibilidad de acceso múltiple y de trabajar con dos antenas simultáneamente.
- Un mundo multimedia a disposición del usuario.
- Mecanismos de seguridad ampliamente mejorados.

Sistema de transmisión:
En la telefonía fija o convencional, los usuarios se conectan en su central a través de una red fija común, compuesta por alambres de cobre. En la red móvil el medio de conexión es el aire, a través de ondas electromagnéticas.

En la actualidad, los móviles han evolucionado de tal forma que no solo disponen de puertos físicos para comunicarse con otros dispositivos, sino que también poseen algún tipo de sistema de comunicación inalámbrica (WAP, Bluetooth, etc.) que permite la transmisión de datos con cualquier tipo de dispositivo (computadores, PDAs, otros celulares).

Clusters:
El término cluster se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora.

Hoy en día desempeñan un papel importante en la solución de problemas de las ciencias, las ingenierías y del comercio moderno.

La tecnología de clusters ha evolucionado en apoyo de actividades que van desde aplicaciones de supercomputo y software de misiones críticas, servidores web y comercio electrónico, hasta bases de datos de alto rendimiento, entre otros usos.

El cómputo con clusters surge como resultado de la convergencia de varias tendencias actuales que incluyen la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran.

Simplemente, un cluster es un grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que los comunes de escritorio.

Los clusters son usualmente empleados para mejorar el rendimiento y/o la disponibilidad por encima de la que es provista por un solo computador típicamente siendo más económico que computadores individuales de rapidez y disponibilidad comparables.

Componentes de un cluster:

En general, un cluster necesita de varios componentes de software y hardware para poder funcionar, como son:

- Nodos.
- Almacenamiento.
- Sistemas operativos.
- Conexiones de red.
- Middleware.
- Protocolos de comunicación y servicios.
- Aplicaciones.

Ambientes de programación paralela.

2.4. Sociedad de la información

El término “Sociedad de la Información” ha sido incorporado, con relativa insistencia, en los años recientes, a la literatura política, académica y mediática contemporáneas. Periodistas, políticos, cibernautas, académicos e investiga- dores suelen evocar tan ambiguo concepto para referirse al tipo de sociedades deseables a las cuales habrá de conducirnos a la globalización.

La Cumbre Mundial sobre la Sociedad de la Información (CMSI)[7], organizada por el sistema de la ONU bajo el auspicio de Kofi Annan, la Unión Internacional de Telecomunicaciones (UIT), y otras agencias de la ONU interesadas en la materia, planea adoptar una declaración que incorpore un conjunto de principios y reglas de conducta destinados a crear a nivel mundial una Sociedad de la Información más inclusiva y equilibrada; un plan de acción y una declaración de principios que formulen propuestas

operativas y medidas concretas para que todos los actores se beneficien mas equitativamente de las oportunidades que concederá la Sociedad de la Información en el futuro.

Las organizaciones cívicas internacionales que más han participado a nivel mundial en la discusión y construcción del futuro que debe adoptar la sociedad de la información a través de la (CMSI), poseen una visión diferente a la manejada por los organismos económicos y políticos internacionales tradicionales que han creado esencialmente un proyecto de expansión de las empresas como negocios eficientes y no en el mejoramiento generalizado de las condiciones de vida de los seres humanos. Por ello, la sociedad civil ha proclamado la otra versión de lo que debe ser la Sociedad de la Información y de la Comunicación y fundamenta su concepción, en las siguientes bases: concepción de la sociedad de la información, principios guías.

2.4.1. Visión sobre las sociedades de la información

La sociedad civil entiende a las sociedades de la información y de la comunicación como realidades basadas en los derechos humanos y en el desarrollo humano duradero. Los sucesos que definen las sociedades de información y comunicación deben basarse en principios de justicia económica, política y social y deben perseguir objetivos de desarrollo humano duradero, además del apoyo a la democracia, la participación, el fortalecimiento y la igualdad de géneros.

2.4.2. La ley de Moore y la visión de Weiser

Los constantes avances en microelectrónica se han convertido en algo común: la ley de Moore, formulada en los años sesenta por Gordon Moore, afirma que la capacidad de computación disponible en un microchip se multiplica por dos aproximadamente cada 18 meses y, de hecho, esto ha resultado ser un pronóstico extraordinariamente exacto del desarrollo del chip desde entonces. Se puede observar también un crecimiento exponencial comparable en otras áreas de la técnica, como por ejemplo en la capacidad de almacenamiento y el ancho de banda para la comunicación. Visto de otra forma, los precios para la funcionalidad microelectrónica con la misma capacidad de computación están bajando gradualmente de forma radical.

Esta tendencia que no cesa producirá una profusión de computadores muy pequeños en un futuro no demasiado lejano, lo que anuncia un cambio de paradigma en las aplicaciones informáticas: se montarán procesadores, dispositivos de memoria y sensores para formar una amplia gama de "aparatos electrónicos de información" baratos, que estarán conectados sin cables a Internet y serán construidos de forma personalizada para realizar tareas específicas. Estos componentes microelectrónicos se podrán incrustar además en casi cualquier tipo de objeto cotidiano, lo que le añadirá "sensibilidad" (smartness), por ejemplo modificando su comportamiento dependiendo del contexto en que se encuentre del objeto. Al final, el procesamiento de la información y las capacidades de comunicación quedarán integrados en objetos que, por lo menos a prime- ra vista, no parecerán de ningún modo aparatos eléctricos, de esta forma las capacidades de la computación se volverán ubicuas.

El término "computación ubicua" (ubiquitous computing), que denota esta visión, fue acuñado hace más de diez años por Mark Weiser, un investigador del Palo Alto Research Center de XEROX. Weiser ve la tecnología solamente como un medio para un fin y como algo que debería quedar en segundo plano para permitir al usuario concentrarse completamente en la tarea que está realizando. En este sentido, considerar el computador personal como herramienta universal para la tecnología de la información sería un enfoque equivocado, ya que su complejidad absorbería demasiado la atención del usuario.

Según Weiser, el computador como dispositivo dedicado debería desaparecer, mientras que al mismo tiempo debería poner a disposición de todo lo que nos rodea sus capacidades de procesamiento de la información.

Weiser ve el término "computación ubicua" en un sentido más académico e idealista como una visión de tecnología discreta, centrada en la persona, mientras que la industria ha acuñado por eso el término "computación persuasiva", o ampliamente difundida "pervasive computing" con un enfoque ligeramente diferente: Aunque su visión siga siendo todavía integrar el procesamiento de la información en objetos cotidianos de forma casi invisible, su objetivo principal es utilizar tales objetos en un futuro próximo en el ámbito del comercio electrónico y para técnicas de negocios basados en la Web.

Esta variante pragmática de computación ubicua está empezando ya a tomar forma: El presidente de IBM Lou Gerstner describía una vez su visión de la ”era post-PC” como “mil millones de personas interactuando con un millón de negocios electrónicos a través de un billón de dispositivos inteligentes interconectados”.

2.5. Metodología XP

2.5.1. Concepto

XP es una metodología de desarrollo ágil basada en una serie de valores y una docena de prácticas las cuales propician un aumento en la productividad a la hora de generar software.

XP se centra en las relaciones interpersonales el cual es la clave de su éxito teniendo muy en cuenta el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Usa la realimentación continua entre el cliente y el equipo de desarrollo, dando una comunicación fluida entre todos los participantes, lo que es aprovechado para tener simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

XP permite controlar los problemas de riesgo en los proyectos.
XP requiere un variado equipo de desarrollo.
XP permite la participación de pequeños grupos de programadores.
XP permite la capacidad de hacer pruebas

La meta real de XP es entregar el software requerido a tiempo, pudiendo haberlo ajustado a todo cambio en su realización.

2.5.2. Características

Las características generales de XP es deliberadamente una metodología “liviana” que pasa por alto la utilización de elaborados casos de uso, la exhaustiva definición de requerimientos y la producción de una extensa documentación.

Todo lo anterior puede parecer caótico según el enfoque tradicional de la ingeniería de software, aunque no hay que olvidar que XP tiene asociado un ciclo de vida y es considerado a su vez un proceso.

La tendencia de entregar software en lapsos cada vez menores de tiempo y con exigencias de costos reducidos y altos estándares de calidad, hace que XP sea una opción a considerar.

2.5.3. Valores de XP

2.5.3.1. Comunicación

Es muy importante entender cuáles son las ventajas de este medio. Cuando dos (o más) personas se comunican directamente pueden no solo consumir las palabras formuladas por la otra persona, sino que también aprecian los gestos, miradas, etc. que hace su compañero. Sin embargo, en una conversación mediante el correo electrónico, hay muchos factores que hacen de esta una comunicación, por así decirlo, mucho menos efectiva.

2.5.3.2. Coraje

El coraje es un valor muy importante dentro de la programación extrema. Un miembro de un equipo de desarrollo extremo debe de tener el coraje de exponer sus dudas, miedos, experiencias sin "embellecer" éstas de ninguna de las maneras. Esto es muy importante ya que un equipo de desarrollo extremo se basa en la confianza para con sus miembros.

2.5.3.3. Simplicidad

Dado que no se puede predecir cómo va a ser en el futuro el software que se está desarrollando; un equipo de programación extrema intenta mantener el software lo más sencillo posible. Esto quiere decir que no se va a invertir ningún esfuerzo en hacer un desarrollo que en un futuro pueda llegar a tener valor. En el XP frases como "...en un futuro vamos a necesitar..." o "Haz un sistema genérico de..." no tienen ningún sentido ya que no aportan ningún valor en el momento.

2.5.3.4. Retroalimentación

La agilidad se define (entre otras cosas) por la capacidad de respuesta ante los cambios que se van haciendo necesarios a lo largo del camino. No se puede dirigir adecuadamente un proceso si no se dispone de realimentación permanente sobre su progreso, la misma que puede provenir del cliente, de los programadores, de herramientas automáticas, etc.

2.5.4. Fases de la metodología

2.5.4.1. Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.5.4.2. Planificación de la entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una

fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

2.5.4.3. Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

2.5.4.4. Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

2.5.4.5. Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

2.5.4.6. Finalización del proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

2.5.5. Doce practicas de XP

La programación extrema, se fundamenta en doce prácticas, que son las actividades que el equipo de un proyecto lleva a cabo cada día, tienen su origen en destrezas bien conocidas en la ingeniería del software, y por tanto, no pueden resultar extrañas. Sin embargo, lo que caracteriza a este conjunto es la cohesión de todos los elementos, y que cada conocimiento ha sido llevado al extremo. La conceptualización de estas doce prácticas vistas desde el enfoque de Carlos Sánchez González se mencionan a continuación.

2.5.5.1. Planificación incremental

Esta práctica busca dividir la funcionalidad de un proyecto en pequeños fragmentos auto-contenidos, cada uno de los cuales se denomina historia de usuario (user story). El cliente y el equipo del proyecto dialogan para decidir cuáles son las más importantes (siempre se hacen las historias más importantes primero) y estimar cuánto puede tardar el equipo en completar cada historia.

El enfoque utilizado para llevar a cabo la planificación es eminentemente práctico. Gran parte de la eficacia de este modelo de planificación deriva de una división clara de responsabilidades, que tiene en cuenta las necesidades del negocio en todo momento.

2.5.5.2. Entregas frecuentes

Se trata de publicar una nueva versión en cuanto sea posible aportar algún nuevo valor al cliente, siguiendo la política de XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas adaptaciones de las aplicaciones con frecuencia. Éstas deben ser tan pequeñas como sea posible, aunque deben añadir suficiente importancia como para que resulten valiosas, de esta forma, se maximiza la retroalimentación y se controla el proyecto con mayor facilidad.

2.5.5.3. Diseño simple

El sistema debe ser el más simple posible y debe cumplir las especificaciones (pruebas de aceptación). En un entorno donde los requisitos del cliente y sus prioridades cambian continuamente, no tiene sentido realizar un diseño exhaustivo. XP define un "diseño tan simple como sea posible" como aquél que:

- Pasa todos los test.
- No contiene código duplicado.
- Deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código.
- Contiene el menor número posible de clases y métodos.

La mejor forma de obtener una idea de los futuros requisitos de un sistema es proporcionar cuanto antes un prototipo al cliente y obtener retroalimentación.

2.5.5.4. Pruebas automáticas

La ejecución automatizada de test es un elemento clave de la XP y la manera de asegurarla con cierta confianza es escribiendo pruebas automáticas con las que pueda comprobar el código en cualquier momento y sin esfuerzo. Las pruebas no pueden dejarse para el final, sino que deben escribirse al mismo tiempo que el código, o incluso antes. El objetivo de los test no

es corregir errores, sino prevenirlos. Elaborarlos exige pensar por adelantado cuáles son los problemas más graves que se pueden presentar, y cuáles son los puntos dudosos. Esto evita muchos inconvenientes y dudas, en lugar de dejar que aparezcan "sobre la marcha".

2.5.5.5. Integración continua

Si la integración es una fase crucial, en la que pueden aparecer errores, es relevante realizarla permanentemente, de esta forma se minimiza el riesgo de una composición errónea. Para poder hacerlo, es imprescindible que el proceso de unificación esté automatizado y pueda verificarse mediante pruebas. La existencia de esta fase de modo separado, tiene efectos laterales indeseables como por ejemplo, si se empieza a hacer codificación individualista, en la que todo el mundo modifica código, hace que se acumulen defectos; cuando lo óptimo es cargar todas las versiones que se realizan en el equipo y se centralicen en un solo lugar.

2.5.5.6. Refactorización

Uno de los objetivos de la XP es mantener la curva de costes tan plana como sea posible, por lo que existen una serie de mecanismos destinados a mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. La refactorización es un proceso disciplinado por el cual se modifica el diseño de un módulo sin afectar a su comportamiento externo. No sólo sirve para mantener el código legible y sencillo, también se utiliza cuando resulta conveniente modificar código existente para hacer más fácil implementar nuevas funcionalidades; haciendo posible compatibilizar el diseño simple con la flexibilidad.

2.5.5.7. Programación por parejas

En la programación por parejas, dos programadores comparten un único ordenador y colaboran para escribir el código o las pruebas. De esta forma, se estimula la comunicación y la transmisión de conocimiento, previamente se detectan los errores y se produce código de alta calidad. Esta práctica proporciona un mecanismo de seguridad enormemente valioso ya que dos personas son responsables del código en cada momento, haciendo menos probable que se caiga en la tentación de dejar de escribir test o se deje de lado tareas por descuido o negligencia. Además contribuye a que se disperse el conocimiento por todo el equipo.

2.5.5.8. Propiedad colectiva del código

La XP aboga por la propiedad colectiva del código. En otras palabras, todo el mundo tiene autoridad para hacer cambios a cualquier código, y es responsable de ellos. Esto permite no tener que estar esperando a otros cuando todo lo que hace falta es algún pequeño cambio. Esta práctica permite que funcionen bien los equipos dinámicos, cuya composición puede variar durante el proyecto.

2.5.5.9. Semana de 40 horas

La programación extrema lleva a un modo de trabajo en que el equipo está siempre al 100%. Una semana de 40 horas en las que se dedica la mayor parte del tiempo a tareas que suponen un avance puede dar mucho de sí, y hace innecesario recurrir a sobreesfuerzos; además los programadores cansados y mentalmente fatigados, se equivocan más. Si las semanas de más de 40 horas son la norma, algo no funciona bien en el proyecto o en la empresa, además, el sobreesfuerzo continuado pronto lleva a un rendimiento menor y a un deterioro de la moral de todo el equipo.

2.5.5.10. Cliente en el equipo

Algunos de los problemas más graves en el desarrollo son los que se originan cuando el equipo de desarrollo toma decisiones de negocio críticas. Esto no debería ocurrir, pero a la hora de la verdad con frecuencia no se obtiene retroalimentación del cliente con la fluidez necesaria, el resultado es que se ha de optar por detener el avance de los proyectos.

Para lograr una retroalimentación ágil, el cliente debe formar parte del equipo; de esta forma, puede ayudar a los programadores a escribir las pruebas de aceptación debido a su inmersión, y realizar conjuntamente los test que verifican si la funcionalidad de la aplicación es la correcta y deseada, obteniendo un feedback absolutamente realista del estado del proyecto.

2.5.5.11. Uso de metáforas

La comunicación fluida es uno de los valores más importantes de la programación extrema, el hecho de incorporar al equipo una persona que represente los intereses del negocio y otras

prácticas son valiosas entre otras cosas porque potencian enormemente la comunicación. Para conseguir que la comunicación sea fluida es imprescindible, utilizar el vocabulario del negocio. También es fundamental huir de definiciones abstractas.

2.5.5.12. Estándares de codificación

XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios.

Para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificarlo, es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP.

CAPÍTULO 3

PLANIFICACIÓN Y ANÁLISIS DE REQUERIMIENTOS

3.1. Faces de planificación

3.1.1. Introducción

Para el desarrollo de un proyecto sin diferencia de importancia o tamaño, siempre como primer paso es el de recabar información, para tener una idea clara sobre los diferentes elementos que van actuar y poder llegar a concluir de manera exitosa nuestro objetivo que será solucionar el problema por el cual se planteo el proyecto.

3.1.2. Apreciación global

El Desarrollo de un de un prototipo de control mediante SMS para casas inteligentes, está formado por una serie de procesos que necesitan estar analizados para tener una visión de las funciones que se encuentran en cada uno de los módulos propuestos, para lo cual en el presente capítulo se razonará el flujo de información, los acontecimientos de cada actividad contemplada, los requisitos funcionales y finalmente los requisitos no funcionales, empleando las historias de usuario, que muestran claramente las necesidades de cada usuario.

3.2. Descripción global

3.2.1. Perspectiva del producto

El sistema a desarrollar contempla el manejo de información de los diferentes usuarios de una casa inteligente, permitiendo un control eficaz sobre los servicios internos de la misma.

El sistema tiene que soportar la comunicación exterior de manera remota siendo el canal de comunicación mensajes de texto.

En la figura 3.1 se describe la idea primordial del sistema.

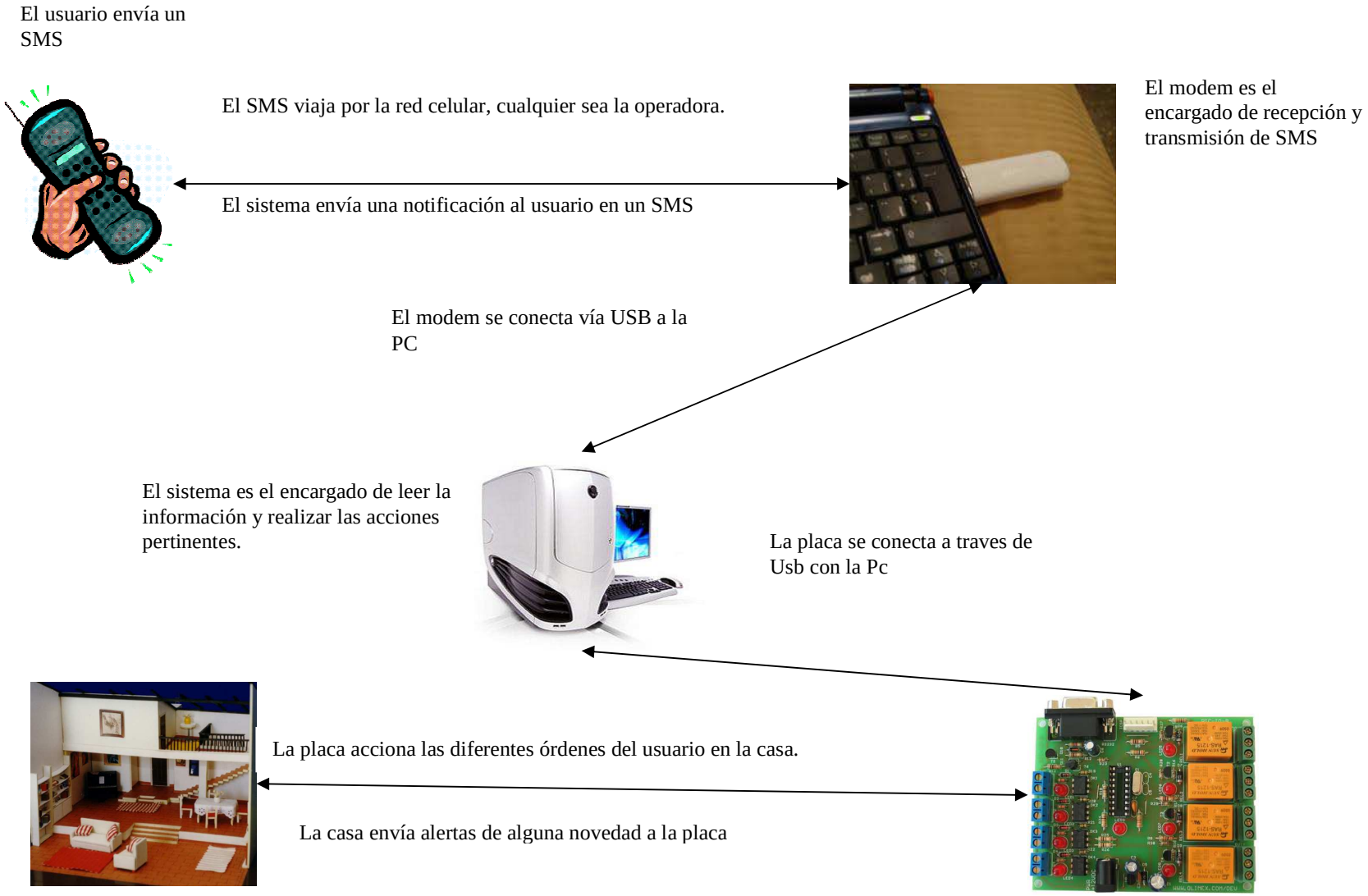

Figura 3. 1: Descripción gráfica del sistema.

3.2.2. Función del producto

Las funciones que realizará el sistema se pueden apreciar en la tabla 3.1:

Tabla 3. 1 Funciones solicitadas por el cliente.

Actividad	Función	Entidad involucrada
El usuario solicita un servicio	Ingresar y/o consultar información del usuario	Administrador
El usuario reciba alertas	Informar sobre activación de alertas	Administrador Usuario
Poder ingresar usuarios	Crear nuevos usuarios y asignar privilegios	Administrador
Tener reportes de eventos	Generar reportes de control	Base de datos Usuario
Poder bloquear usuario	Manejar los permisos de cada usuario	Administrador
Agregar nuevas alarmas a la casa	Ingresar nuevas alarmas	Administrador
Abrir puertas vía remota	Manejar dispositivos	Administrador Base de datos

3.3. Casos de uso

3.3.1. Identificación de actores

Los actores que intervienen dentro de la aplicación son: administrador, cliente, SMS, dispositivos y base de datos, se esquematiza en la figura 3.2

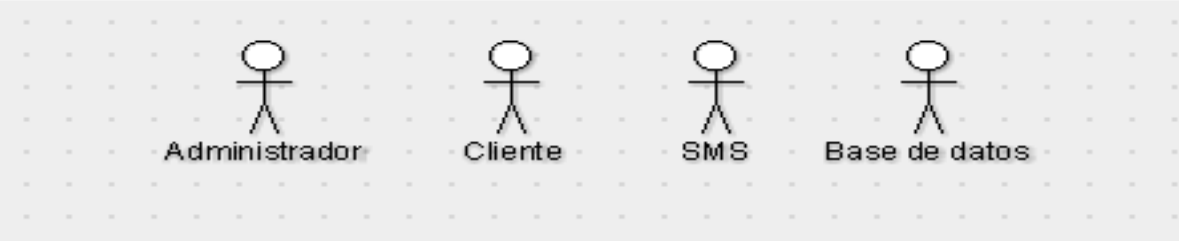

Figura 3. 2: Actores del sistema.

3.3.2. Caso de uso del proceso de control

El administrador, cliente, SMS, dispositivos y base de datos, son los principales actores que intervienen el sistema de control de la casa inteligente como representa la figura 3.3

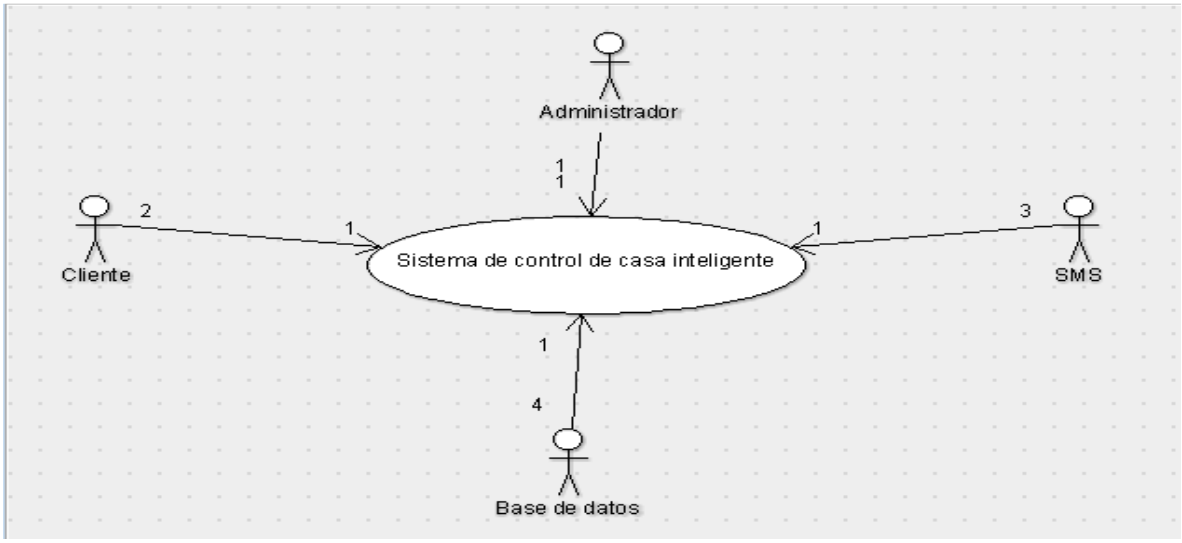

Figura 3. 3: Caso de uso proceso de control.

3.3.3. Caso de uso por actores

Todos los actores principales del sistema de control tienen privilegios de acceso, cada uno de ellos cumplen con dichos permisos, como se puede observar a continuación en la figura 3.4

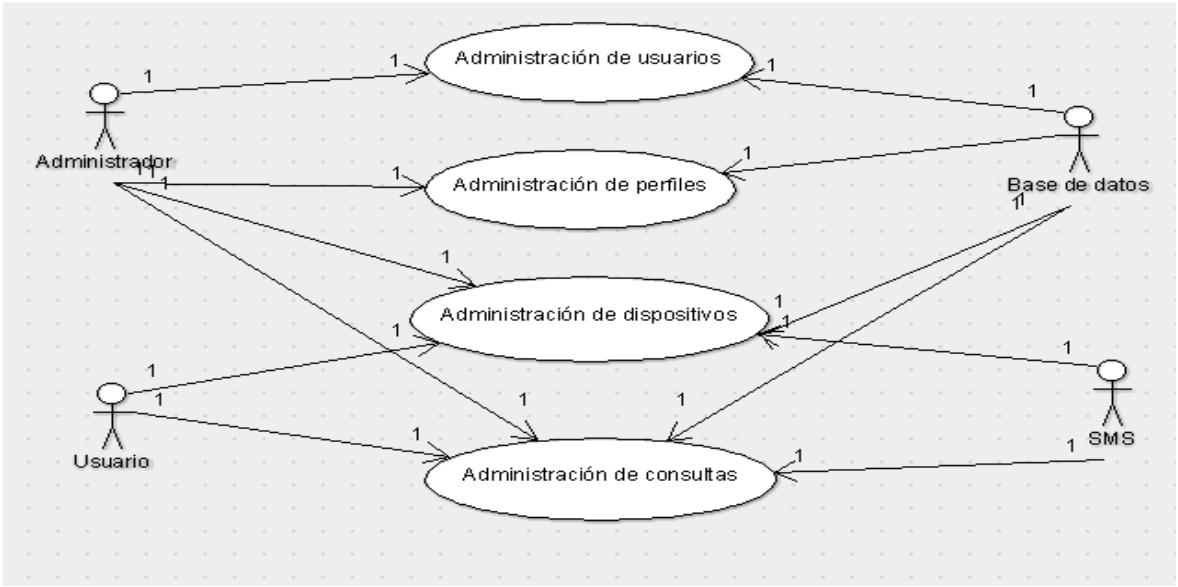

Figura 3. 4: Caso de uso por actores.

3.3.4. Caso de uso general

La figura 3.5, muestra el funcionamiento general del sistema, cada uno de los actores cumplen con diferentes cargos y permisos, los cuales permiten un control eficaz del sistema.

Con el diagrama de uso general tenemos una idea de lo que el sistema va a realizar y como los distintos actores manejan o intervienen en el sistema.

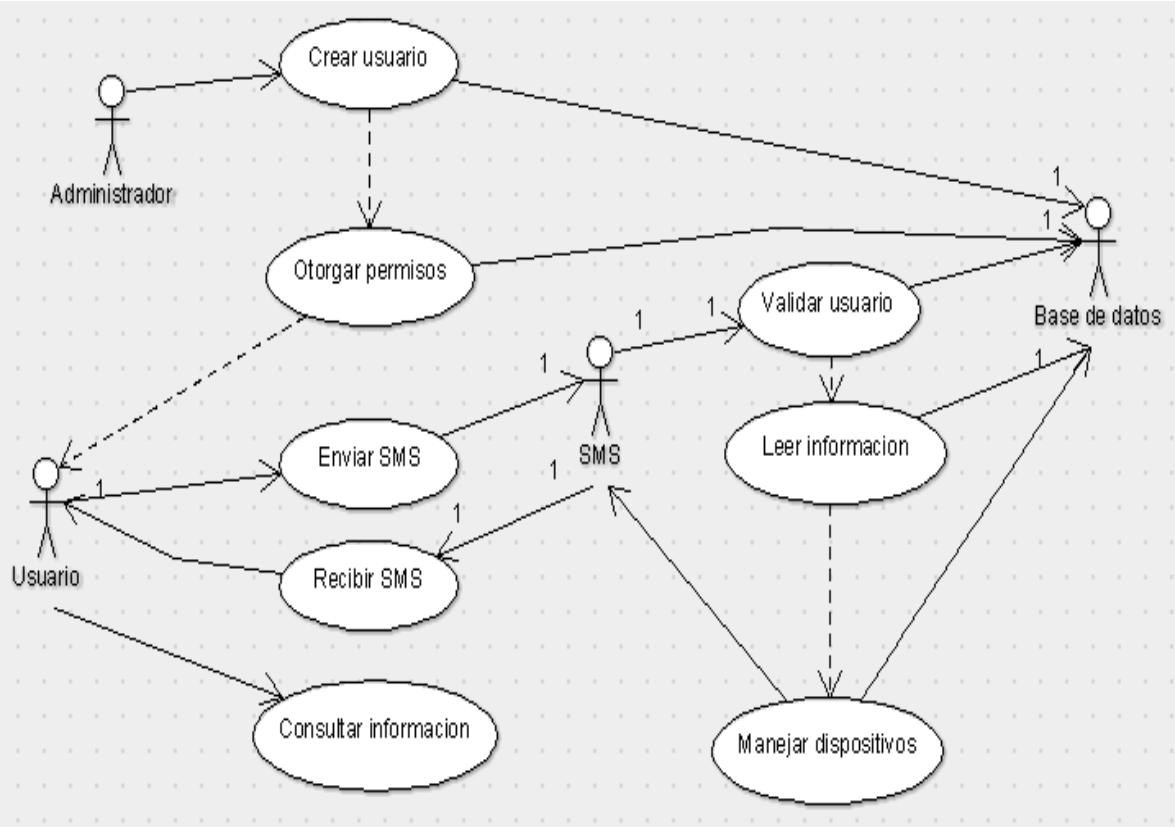

Figura 3. 5: Caso de uso general

3.3.5. Casos de uso por módulos

3.3.5.1. Módulo usuarios

En el módulo de usuarios se manejara distintos procesos como son ingreso de nuevos usuarios, validación de permisos, asignación de contraseñas y demás, tal como se muestra en la figura 3.6.

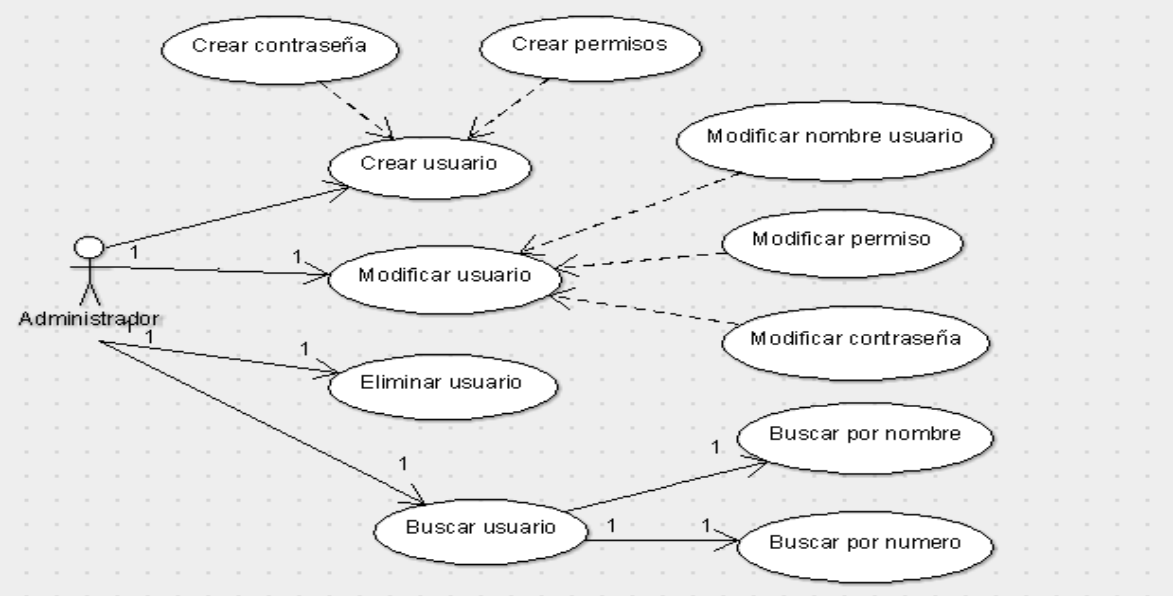

Figura 3. 6: Casos de uso módulo usuarios

3.3.5.2. Módulo administración de perfiles

En la figura 3.7 se puede observar los diferentes casos de uso que tiene el administrador con relación al sistema.

El módulo de administración de perfiles es el encargado del manejo de accesos que tienen los distintos usuarios en relación al sistema.

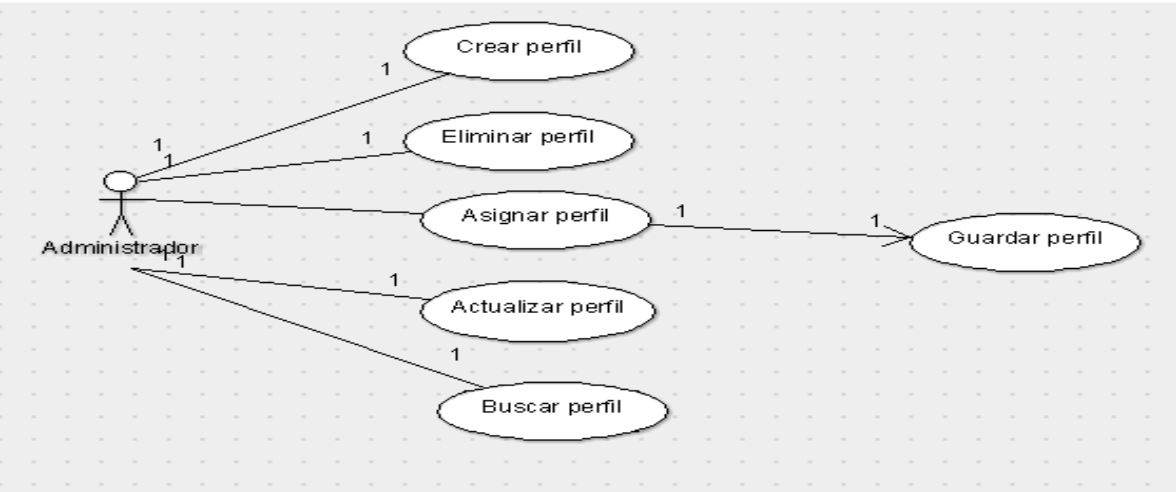

Figura 3. 7 Casos de uso de módulo administración de perfiles.

3.3.5.3. Módulo administración de dispositivos

En la figura 3.8 se puede observar la relación que el usuario tiene con los dispositivos para acceder a la base de datos y de esa manera tener acceso al manejo del sistema.

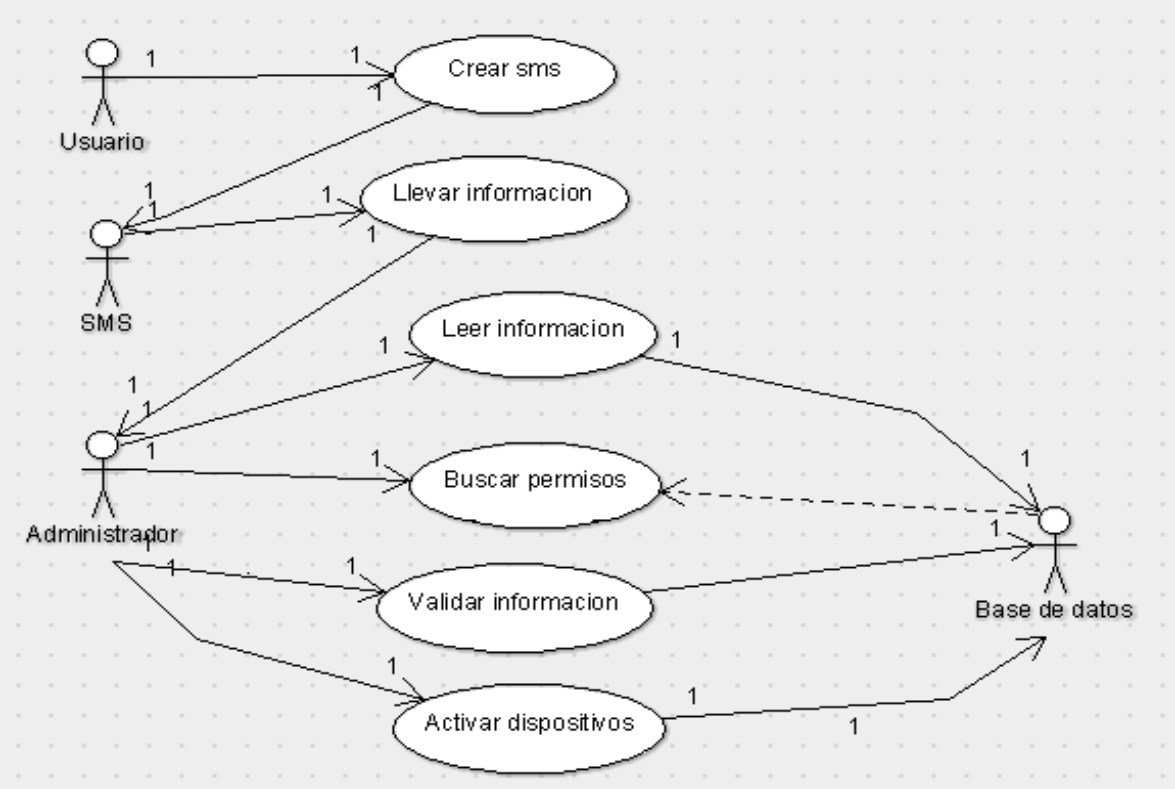

Figura 3. 8 Casos de uso de administración de dispositivos.

3.3.5.4. Módulo administración de consultas

El módulo de administración de consultas maneja información guardada en la base de datos, la cual es la encargada de registrar las distintas actividades desarrolladas por los usuarios del sistema.

Las consultas se realizarán por parte de los usuarios que tengan permiso para acceder a dicha información, las consultas tendrán distintos parámetros de búsqueda como es usuario, intervalos de fechas, equipos.

La figura 3.9 muestra a los usuarios que intervienen en el módulo.

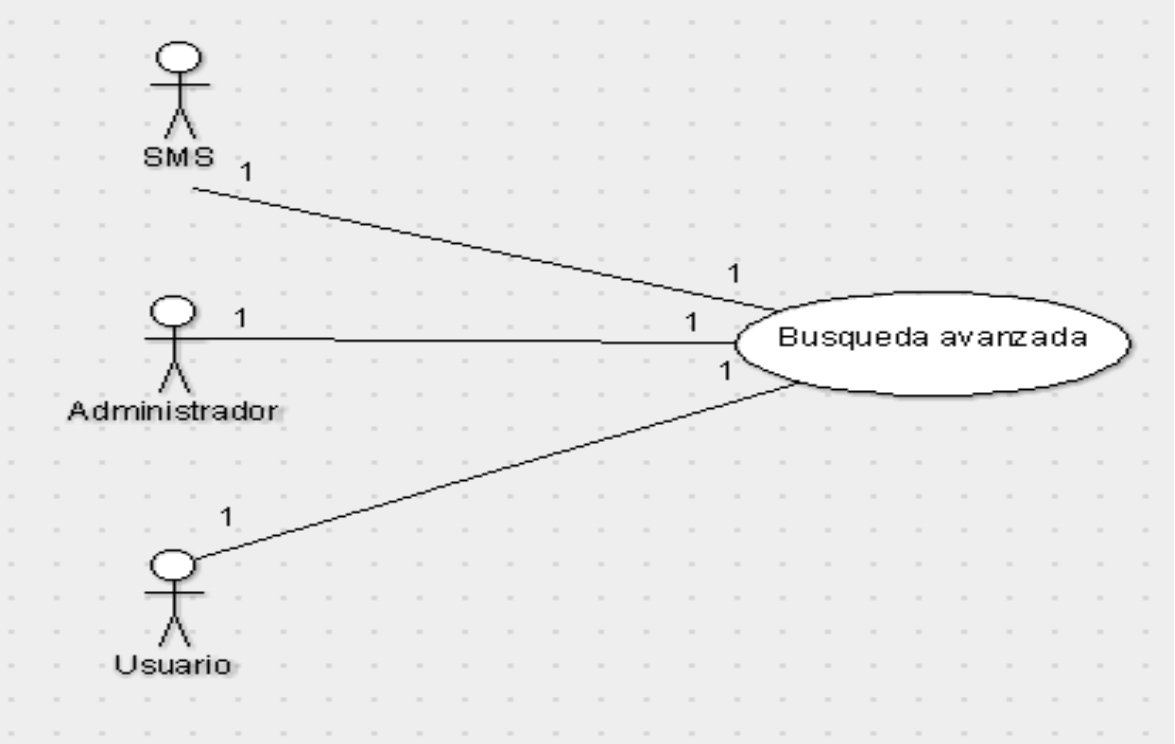

Figura 3. 9: Casos de uso de administración de consultas.

3.4. Características del usuario

Para poder manejar adecuadamente el sistema el usuario del mismo tendrá como requerimientos las siguientes condiciones:

- Conocimientos básicos de computación.
- Conocimiento del módulo que van a operar.
- Conocimientos básicos de uso celular.
- Ser organizados.

3.5. Requisitos específicos

Los requisitos específicos del software es la parte medular para poder desarrollar un sistema, pues debe poseer un nivel de detalle suficiente que permiten a los desarrolladores, satisfacer los requerimientos a cabalidad.

Los requisitos están inmersos en las historias de usuario, las mismas que permiten aclarar todas las necesidades del usuario a la hora de manejar determinado proceso.

Esta es la parte esencial de la Especificación de Requisitos del Software, por lo que, cada necesidad se ha especificado teniendo en cuenta, los parámetros principales de los procesos y actividades inmersas.

3.5.1. Requisitos funcionales

Los requisitos funcionales se representan a través de las historias de usuario que muestran las distintas necesidades a la hora de tener acceso a la casa de manera remota.

Tabla 3. 2 Historia de Usuario I - Administrar usuarios y perfiles.

Historia N.- 01	Administrar usuarios y perfiles
Objetivo: **Llevar un control de los usuarios en el sistema.**	
Descripción: **El administrador asigna un perfil de usuario a cada miembro activo de la casa, este perfil ha sido catalogado dependiendo de la actividad que desempeña cada miembro familiar, es por ello que cada perfil tiene permisos para el ingreso a ciertas pantallas y a otras tiene restricción para evitar que haya manipulación de la información.** **Esta administración de perfiles permite que puede ingresar solo personal calificado a las diferentes áreas, el sistema permite ingresar: el identificador del usuario, nombre del usuario, la clave, numero de celular y el perfil de usuario.** **La contraseña o clave se cifra en la base de datos luego de la primera vez que el trabajador ingrese al sistema con su nombre de usuario y su contraseña.**	

Tabla 3. 3 Historia de usuario III - Registro de dispositivos

Historia N.- 02	Registro de dispositivos
Objetivo: **Ingresar la información relevante de una nueva alarma o equipo electrónico a la casa.**	
Descripción: **La prestación de un determinado servicio requiere de una alarma o equipo que no ha sido considerado antes, motivo por el cual se requiere la compra del mismo y la integración a la base de datos con la que cuenta la casa inteligente, el administrador, ingresa la información pertinente tal como nombre, código y ubicación.**	

Tabla 3. 4 Historia de usuario III - Administración de SMS

Historia N.- 03	Administración de SMS
Objetivo: **Permitir la interacción entre un mensaje de celular con el sistema y llevar un registro.**	
Descripción: **Mediante un mensaje de texto se tiene control remoto del sistema de manera práctica y compacta, de la misma manera el sistema lleva un registro de los mensajes recibidos los cuales son validados de acuerdo a los registros de la base de datos, permitiendo las acciones solicitadas por el usuario.** **En el celular se tendrá una aplicación de fácil uso en la cual se seleccionara que acción se va a realizar en la casa.**	

Tabla 3. 5 Historia de usuario IV - Generar reportes de control

Historia N.- 04	Generar reportes de control
Objetivo: **Desarrollar reportes que permitan controlar todos los procesos de los usuario.**	
Descripción: **Los usuarios solicitan al sistema la generación de un determinado reporte que le permita llevar un mejor control de los procesos que han ocurrido en la casa.** **Entre los reportes podemos mencionar, los usuarios que accedieron al sistema, detalle de dispositivos, alarmas activadas; entre otros reportes que solicitarán diferentes parámetros de búsqueda además del rango de fechas entre las que se considera la información que se desea mostrar.**	

3.5.2. Requisitos no funcionales

"Una buena definición de requisito no funcional es la dada por Thayer [4]: es un requisito software que describe no lo que el software hará, sino como lo hará, como por ejemplo, requisitos de rendimiento. Los requisitos no funcionales son difíciles de verificar/testear, y por ello son evaluados subjetivamente…[21]

[21] Dorfman, M. and Thayer, R., "Standards, Guidelines and Examples on System and Software

Dentro de los requisitos no funcionales que se debe considerar en el presente proyecto tenemos:

- **Calidad:** el sistema deberá tener una atención a los usuarios de manera rápida, a la vez que permita al usuario una administración fácil, y con una interfaz sencilla y amigable que permita el manejo intuitivo del sistema por parte del usuario.

- **Disponibilidad:** el sistema deberá permitir que varios usuarios interactúen con el sistema y a su vez manipulen de manera remota los dispositivos de la casa.

- **Documentación:** como en todo sistema la documentación es parte importante para la futura manipulación del sistema, por lo tanto se deberá tener los respectivos manuales de usuario así como también el manual de configuración.

- **Eficiencia:** el sistema trabaja con un tiempo de respuesta optimizado arrojando información confiable sobre las consultas que se elaboren.

- **Integridad:** La información ingresada por parte del usuario de este sistema posee integridad de datos y lógica.

  La integridad lógica se refiere a las relaciones que existen entre las reglas de negocio y la base de datos, las cuáles ayudan a la ejecución correcta de los procesos.

- **Seguridad:** el sistema cuenta con los niveles de seguridad óptimos que garanticen el manejo de perfiles y controles de acceso.

3.5.3. Restricción del diseño

A continuación se mencionan las restricciones de diseño que tendrá el sistema:

Requirements Engineering", IEEE Computer Society Press, 1990.

- La hora del sistema no puede ser modificada a conveniencia de ningún usuario.

- Los colores que maneje la interfaz representara los colores de la empresa Soft. Teratronic.

- Los códigos de identificación, es decir la cédula son validados.

- Ningún usuario tendrá acceso a los módulos que no le competen según su perfil.
- Todas las pantallas son amigables para el fácil manejo del sistema.

- Todas las pantallas donde se almacene la información tienen las opciones de eliminación, actualización e inserción de información.

3.5.4. Atributos del sistema

El sistema cuenta con el acceso a una base de datos previamente inicializados y normalizados con todos los parámetros relacionados a la prestación de servicios.

Cada uno de los datos que se ingresarán al sistema, serán validados para evitar cualquier tipo de error.

El sistema cuenta con el respectivo manual de instalación y manuales de usuario que oportunamente se dará a conocer al momento de impartir la capacitación a quienes operarán y administrarán el sistema.

3.6. Planificación de entrega

Siguiendo la metodología XP, se establece un plan de entrega que sirva de referencia al equipo de desarrollo, para tener conocimiento del cumplimiento, avance y límites en cada etapa del proyecto.

El cliente es quien define de acuerdo a sus prioridades, el orden en el que se desarrollará cada historia de usuario, dependiendo de los cambios o mejoras que se manifiesten en cada presentación de avances, se realiza una nueva iteración.

Para la entrega de cada historia de usuario se tiene en cuenta:

- El tiempo de desarrollo.
- El número estimado de iteraciones posibles para cada historia de usuario.

Para sacar un tiempo estimado de desarrollo, se toma en cuenta la rapidez del equipo para entregar las historias de usuario, utilizando como referente el tiempo estimado tal como se indica en la tabla 3.6.

Tabla 3. 6 Planificación de entrega

MÓDULO	Historia de usuario y/o actividades a desarrollar	Tiempo Estimado		
		Semanas (estimadas)	Días (estimados)	Horas (estimadas)
PASOS PREVIOS	Instalación de las herramientas que permitirán la creación del ambiente de desarrollo	1	5	40
Módulo – Administración perfiles	Ingresar información de perfiles	0,8	4	32
	Ingresar permisos del perfil	0,6	3	24
	Generar lista de perfiles	1,4	7	56
Módulo – administración Usuarios	Registrar nuevo usuario	0,8	4	32
	Registrar tipo de perfil	0,8	4	32
	Asignar celular	1	5	40
	Generar reportes de control	0,8	4	32
Módulo – administración dispositivos	Registrar nuevo dispositivo	1,8	9	72
	Registrar estado	1,2	6	48
	Registrar movimientos	1,2	6	48
	Generar reporte de movimientos	1	5	40
ENTREGA Y EVALUACIÓN	Generación de encuesta para conocer el grado de aceptación del sistema	0,6	3	24

DEL SISTEMA	Retroalimentación y capacitación del manejo del sistema	0,6	3	24
	Generación de manuales de usuario	0,8	4	32
	Configuraciones en el servidor para el buen funcionamiento del sistema.	0,6	3	24
	Generación del acta de entrega-recepción del sistema y documentación	0,2	1	8
	TOTALES	**15,2**	**76**	**608**

CAPÍTULO 4
ARQUITECTURA

Teniendo en cuenta la necesidad del usuario y con el respectivo análisis se llego a la conclusión que la arquitectura cliente servidor de 3 capas es la que mejor se acopla al proyecto.

4.1. Cliente-Servidor de tres capas

La arquitectura cliente servidor consiste en la realización de peticiones de uno o varios clientes a un programa alojado en un servidor que acepta dicha solicitud y envía una respuesta. Para tener una visualización de lo expuesto referirse la figura 4.1

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.[22]

Vale mencionar que si el sistema a realizar es multicapa, este se descompone en diferentes soluciones, lo que permite de ser necesario la ejecución de las distintas respuestas en diferentes computadoras aumentando así el grado de distribución del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa, pudiendo acceder al servidor varios clientes desde distintas maquinas en tiempo real. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, entre otros.

[22] Información tomada de http://es.wikipedia.org/wiki/Cliente-servidor

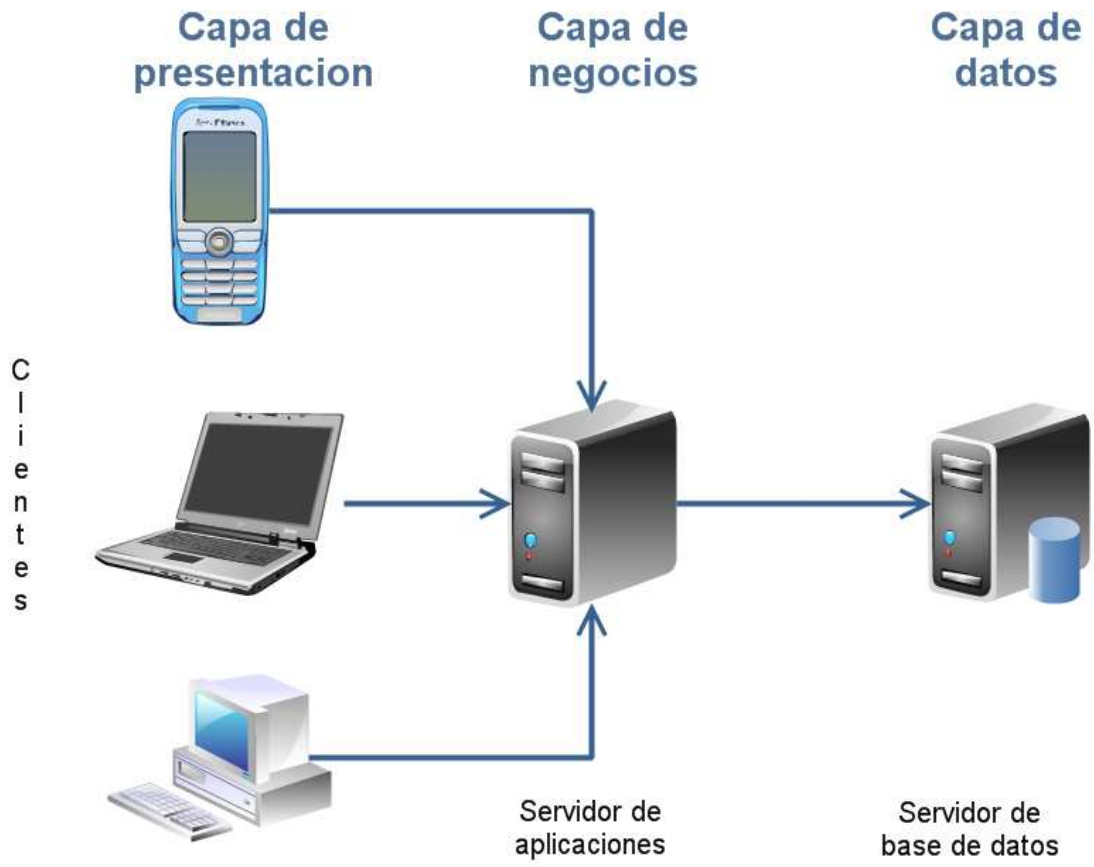

Figura 4. 1: Arquitectura de 3 capas

4.1.1. Capas y niveles

4.1.1.1. Capa de presentación

Es la que ve el usuario o también denominada "capa de usuario", presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso realizando un filtrado previo para comprobar que no hay errores de formato.

Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser amigable, entendible y fácil de usar para el usuario.

4.1.1.2. Capa de negocio

Es donde residen los programas que se ejecutan, recibe las peticiones del usuario y envía las respuestas tras el proceso. Se denomina capa de negocio e incluso de lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse.

Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos tareas como almacenamiento o recuperación de datos

4.1.1.3. Capa de datos

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador, si bien lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación, que son los clientes de la arquitectura cliente/servidor. Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores.

Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio. Si, por el contrario, fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares; el término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación/ Lógica de Negocio/ Datos.

En cambio, el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en un solo ordenador (presentación + lógica + datos). Se dice que la arquitectura de la solución es de tres capas y un nivel.

- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en dos ordenadores (presentación + lógica, lógica + datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles.

- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en tres ordenadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y tres niveles.

4.2. MVC (Modelo Vista Controlador)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Para un mejor entendimiento es prudente resaltar las características de los elementos del patrón.

4.2.1. Modelo

Accede a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si se encienden la alarma del hogar, enviar una alerta al celular del usuario principal".

Lleva un registro de las vistas y controladores del sistema.
Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

4.2.2. Vista

- Recibe datos del modelo y lo muestra al usuario.

- Tienen un registro de su controlador asociado, porque normalmente lo instancia.

- Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes.

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos: en MVC corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio de capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo ya que ambos cumplen ciclos de vida muy distintos entre sí.

4.2.3. Controlador

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas

peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_estado_equipo(nueva_accion_estado)".

4.3. Módulos del sistema

4.3.1. Administración de usuarios y perfiles

El módulo de administración de usuarios es el encargado de dar los diferentes permisos a los usuarios, como también de permitir el ingreso de nuevos administradores del sistema.

En el módulo se ingresaran información relevante del usuario, que luego serán manejados por los otros módulos del sistema para tener un sistema integrado e integro. A la vez se tiene un seguimiento de las diferentes acciones que se realicen en el módulo.

4.3.2. Administración de dispositivos

El módulo de administración de dispositivos, permite el ingreso de nuevos dispositivos del hogar, a la vez de dar mantenimiento de los ya instalados.

Con este módulo se logra armonizar la relación entre el hogar y el sistema domótica, teniendo un registro de las acciones efectuadas por parte del usuario.

4.3.3. Administración de reportes

El módulo de administración de reportes, integra la salida de los procesos generados en los módulos anteriores que abarca la relación hogar-sistema.
La información procesada se reporta a los usuarios dependiendo del tipo de permiso que tenga, permitiendo la toma de decisiones en un futuro.

4.4. Estándares de programación JAVA

4.4.1. Introducción

Para un correcto desarrollo en la programación se debe tener en cuenta estándares o convenciones de programación empleados en el desarrollo de software sobre la plataforma

Java. Este modelo de programación está basado en los estándares recomendados por Sun Microsystems, los cuales han sido difundidos y aceptados ampliamente por toda la comunidad Java, y que han terminado por consolidarse como un modelo estándar de programación de facto.

La utilización de normas es primordial en la estructura de un sistema, como se menciona en los siguientes puntos:

- Facilitan el mantenimiento de una aplicación. Dicho mantenimiento constituye el 80% del coste del ciclo de vida de la aplicación.

- Permite que cualquier programador entienda y pueda mantener la aplicación. En muy raras ocasiones una misma aplicación es mantenida por su autor original.

- Los estándares de programación mejoran la legibilidad del código, al mismo tiempo que permiten su compresión rápida.

4.4.2. Organización de ficheros

Las clases en Java se agrupan en paquetes. Estos paquetes se deben organizar de manera jerárquica, de forma que todo código desarrollado para el “Sistema de domótica” tendrá que estar incluido dentro del paquete "com.domotica".

Dentro del paquete principal las clases se organizarán en subpaquetes en función del área, organismo o sección al que pertenezca el código desarrollado. Por ejemplo, si estamos desarrollando la interfaz de usuario las clases de dicho servicio se incluirían en el paquete "com.domotica.iu" o similar.

Un fichero consta de secciones que deben estar separadas por líneas en blanco y comentarios opcionales que identifiquen cada sección.

Deben evitarse los ficheros de gran tamaño que contengan más de 1000 líneas. En ocasiones, este tamaño excesivo provoca que la clase no encapsule un comportamiento claramente definido, albergando una gran cantidad de métodos que realizan tareas funcional o conceptualmente heterogéneas.

4.4.2.1. Fichero fuente

Cada fichero fuente Java debe contener una única clase o interfaz pública. El nombre del fichero tiene que coincidir con el nombre de la clase. Cuando existan varias clases privadas asociadas funcionalmente a una clase pública, podrán colocarse en el mismo fichero fuente que la clase pública. La clase pública debe estar situada en primer lugar dentro del fichero fuente.

En todo fichero fuente Java distinguimos las siguientes secciones:

- Comentarios de inicio.
- Sentencia de paquete.
- Sentencias de importación.
- Declaraciones de clases e interfaces.

4.4.2.2. Sentencias de paquetes

La primera línea no comentada de un fichero fuente debe ser la sentencia de paquete, que indica el paquete al que pertenecen las clases incluidas en el fichero fuente. Por ejemplo:
package javax.crypto;

4.4.2.3. Sentencias de importación

Tras la declaración del paquete se incluirán las sentencias de importación de los paquetes necesarios. Esta importación de paquetes obligatorios seguirá el siguiente orden:

- Paquetes del JDK de java.

- Paquetes de utilidades no pertenecientes al JDK de Java, de frameworks de desarrollo o de proyectos opensource tales como apache, hibernate, springframework, etc.

- Paquetes desarrollados para el “Sistema de domótica”.

- Paquetes de la aplicación.

Se recomienda minimizar en la medida de lo posible el uso de importaciones del tipo "package.*", pues dificultan la comprensión de las dependencias existentes entre las clases utilizadas por la aplicación. En caso contrario, se recomienda utilizar comentarios de línea tras la importación.

4.4.2.4. Declaración de clases e interfaces

En la tabla 4.1 se describe los elementos que componen la declaración de una clase o interfaz, así como el orden en el que deben estar situados

Tabla 4. 1 Declaración de una clase

Elementos de declaración de una clase / interfaz	**Descripción**
Comentario de documentación de la clase/interfaz /** ... */	Permite describir la clase/interfaz desarrollada. Necesario para generar la documentación de la api mediante javadoc.
Sentencia class / interface	
Comentario de implementación de la clase/interfaz, si es necesario /* ... */	Este comentario incluye cualquier información que no pueda incluirse en el comentario de documentación de la clase/interfaz.
Variables de clase (estáticas)	En primer lugar las variables de clase públicas (public), después las protegidas (protected), posteriormente las de nivel de paquete (sin modificador), y por último las privadas (private).
Variables de instancia	Primero las públicas (public), después las protegidas

	(protected), luego las de nivel de paquete (sin modificador), y finalmente las privadas (private).
Constructores	
Métodos	Deben agruparse por funcionalidad en lugar de agruparse por ámbito o accesibilidad. Por ejemplo, un método privado puede estar situado entre dos métodos públicos. El objetivo es desarrollar código fácil de leer y comprender.

4.4.3. Sangría

Como norma general se establecen 4 caracteres como unidad de sangría. Los entornos de desarrollo integrado (IDE) más populares, tales como Eclipse o NetBeans, incluyen facilidades para formatear código Java.

4.4.4. Declaraciones

4.4.4.1. Declaraciones por línea

Se recomienda el uso de una declaración por línea, promoviendo así el uso de comentarios. Ejemplo:

```
int idUsuario;   // Identificador del usuario.
String[] funciones; // Funciones de la unidad.
```

4.4.4.2. Inicialización

Toda variable local tendrá que ser inicializada en el momento de su declaración, salvo que su valor inicial dependa de algún valor que tenga que ser calculado previamente. Ejemplo:

```
int idUsurario = 1;
String[] funciones = { "Administración", "Intervención", "Gestión" };
```

4.4.4.3. Paquetes

Se escribirán siempre en letras minúsculas para evitar que entren en conflicto con los nombres de clases o interfaces. El prefijo del paquete siempre corresponderá a un nombre de dominio de primer nivel, tal como: es, eu, org, com, net, etc.

El resto de componentes del paquete se nombrarán de acuerdo a las normas internas de organización de la empresa: departamento, proyecto, máquina, sección, organismo, área, etc.

Generalmente se suele utilizar el nombre de dominio de Internet en orden inverso. Cuando dicho nombre contenga un carácter "-", este se sustituirá por el carácter "_".

Ejemplos:
java.util.ArrayList
java.util.Date
java.util.Properties

javax.servlet.http.HttpServletRequest
javax.servlet.http.HttpServletResponse

4.4.4.4. Clases e interfaces

Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúsculas. Si el nombre es compuesto, cada palabra componente deberá comenzar con maýusculas.

Los nombres serán simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa (URL, HTTP, etc.).

Las interfaces se nombrarán siguiendo los mismos criterios que los indicados para las clases. Como norma general toda interfaz se nombrará con el prefijo "I" para diferenciarla de la clase que la implementa (que tendrá el mismo nombre sin el prefijo "I"). Ejemplo:

class Ciudadano
class OrganigramaDAO
class AgendaService
class IAgendaService

4.4.4.5. Métodos

Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras cada una de ellas tendrá la primera letra en mayúsculas. Ejemplo:

public void insertarUnidad(Unidad unidad);
public void eliminarEquipo(Equipo equipo);
public void actualizarUsuario(Usuario usuario)

4.4.4.6. Variables

Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas.

Las variables nunca podrán comenzar con el carácter "_" o "$". Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales. Ejemplo:
Unidad unidad;
Equipo equipo;
Tramite tramite;

4.4.4.7. Constantes

Todos los nombres de constantes tendrán que escribirse en mayúsculas. Cuando los nombres de constantes sean compuestos las palabras se separarán entre sí mediante el carácter de subrayado "_". Ejemplo:

int LONGITUD_MAXIMA;
int LONGITUD_MINIMA;

4.4.5. Modelo de datos

El uso de estándares en el modelado y diseño de base de datos, permite el mantenimiento y fácil interpretación del modelo, además de prolongar la vida útil del software y asegura la calidad del mismo.

Algunas ventajas que se tienen al usar estándares son:

- Asegurar la legibilidad del modelo de datos, inclusive para personas que no están relacionadas con el ambiente informático, en etapas de análisis y diseño.

- Facilitar la portabilidad entre motores de bases de datos, plataformas y aplicaciones.

- Facilitar la tarea de los programadores en el desarrollo de los sistemas.

Es por esto que la codificación de las tablas de las bases de datos a desarrollar debe cumplir ciertos requisitos, detallados en el presente documento. Estos requisitos pueden aplicarse a cualquier motor de bases de datos.

4.4.5.1. Reglas generales

Los nombres de las tablas, en su gran mayoría, se escriben en mayúscula, siguiendo el siguiente estándar:

- X: representa el módulo.
- Y: tipo de objeto.
- ZZZ: Nombre corto de la tabla.
- AAAAA: Nombre largo de la tabla.

Ejemplo:

EESEC_SECTOR

En la figura 4.2 se puede visualizar de mejor manera el uso de las reglas en el ejemplo anterior.

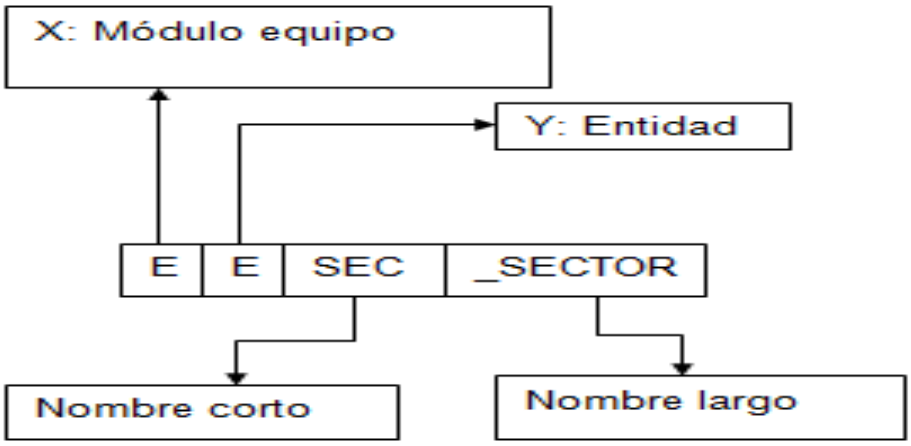

Figura 4. 2 Modelo de tablas en la base de datos.

- Únicamente se utilizarán caracteres alfabéticos, salvo que por la naturaleza del nombre se necesiten dígitos numéricos. Se prohíbe el uso de caracteres de puntuación o símbolos. Ejemplo:

  localidadesCiudades01.

- Las letras acentuadas se reemplazarán con las equivalentes no acentuadas, y en lugar de la letra eñe (ñ) se utilizará (ni).

  Ejemplos: anioEquipo,reporteAnio.

- El nombre elegido debe ser lo más descriptivo posible, evitando términos ambiguos o que se presten a distintas interpretaciones.

  Ejemplo: tiposUsuarios => categoriasUsuarios.

- El nombre no debe abreviarse, salvo que por necesidad específica deban especificarse más de una palabra en el mismo.

  Ejemplo: ido => idOrganismo, freg => fechaRegistro

Agregar comentarios a las bases de datos y los campos, sobre todo a los booleanos.

4.4.5.2. Tablas

- Toda tabla debe poseer uno o más campos clave.

- Toda relación entre tablas debe implementarse mediante constraints (claves foráneas) con integridad referencial, de acuerdo al motor de base de datos utilizado.

- La integridad referencial deberá actualizar en cascada en todos los casos, y restringir el borrado salvo para las entidades débiles.

- Los campos clave deben ubicarse al inicio de la definición de la tabla (deben ser los primeros).

- El nombre del campo clave debe estar compuesto por "id" + nombre de la tabla en singular (para claves no compuestas). Dependiendo de la naturaleza de la entidad, el nombre de la tabla a usar es el de la misma tabla, o el de la relacionada.

  Ejemplos: tabla localidades => idLocalidad.

- Las claves compuestas sólo deben utilizarse en casos específicos, por ejemplo, tablas de relación o entidades débiles. Si una tabla X con clave compuesta necesita ser referenciada desde otra tabla Y, deberá generarse un campo clave en X al inicio de la misma como "idX", y generar un índice único en los campos que la identificaban.

4.4.5.3. Procedimientos y funciones

- Los procedimientos deben contener un nombre apropiado que detalle a breves rasgos su funcionalidad, no se permitirá procedimientos con letras y/o números.

- En ciertos procedimientos considerados especiales por el equipo de desarrollo, se colocarán los prefijos sp<<NombreProcedimiento>> ó PROC<<NombreProcedimiento>> de modo que se distingan de los procedimientos comunes del repositorio.

- Las funciones deben contener un prefijo GET<<NombreFuncion>> ó get<<NombreFuncion>> de modo que se distingan de otros tipos de objetos del repositorio.

4.4.6. Diseño conceptual

El esquema conceptual es la abstracción de hechos reales de los cuales se emite un concepto o es posible hacer una idea de ello, la cual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que ésta maneja.

Para poder realizar la abstracción de un tema en un área específica, a nivel informático, es necesario tener los requerimientos formulados por los usuarios con respecto a este.

Estos requerimientos contienen el conjunto de hechos y reglas que dan pauta a la creación del esquema conceptual donde por medio de este se podrá realizar una descripción de alto nivel de la futura base de datos.

Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema. La más popular es la notación del modelo entidad-relación.

En esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este esquema se le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren la semántica

(significado) de los datos de la empresa: encuentran entidades, atributos y relaciones. El objetivo es comprender:

- La perspectiva que cada usuario tiene de los datos.
- La naturaleza de los datos, independientemente de su representación física.
- El uso de los datos a través de las áreas de aplicación.

4.4.6.1. Modelo conceptual

En la figura 4.3 se puede apreciar el modelo conceptual del sistema, mediante este modelo se tiene idea de los distintos atributos y eventos que manejaran las distintas clases del sistema.

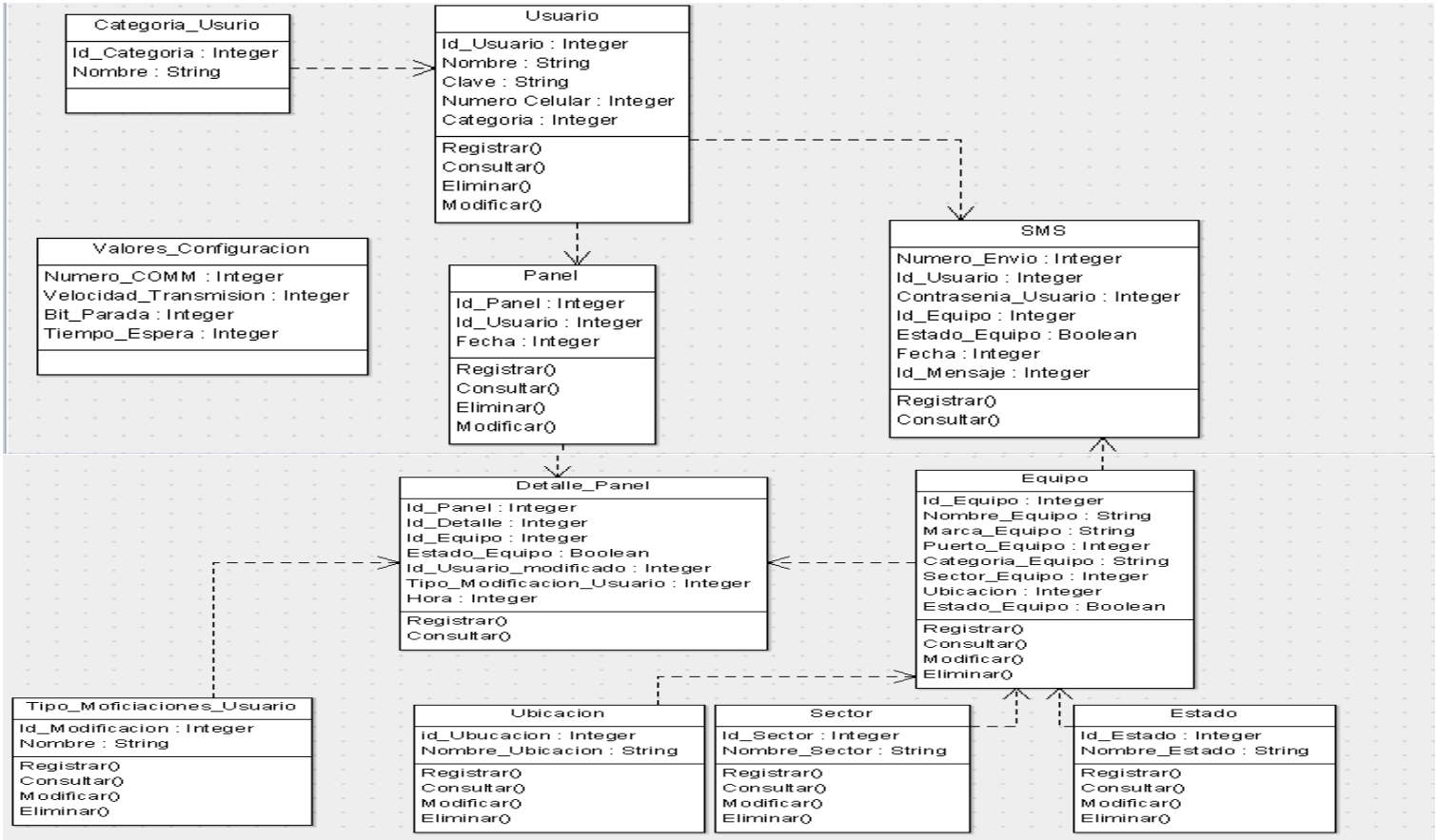

Figura 4. 3: Mapa conceptual del sistema.

4.4.7. Diseño físico

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas.

- Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.

- Diseñar el modelo de seguridad del sistema.

4.4.7.1. Modelo físico

La grafica 4.4 nos muestra el resultado del análisis de las distintas relaciones que tendrán los actores del sistema en la base de datos

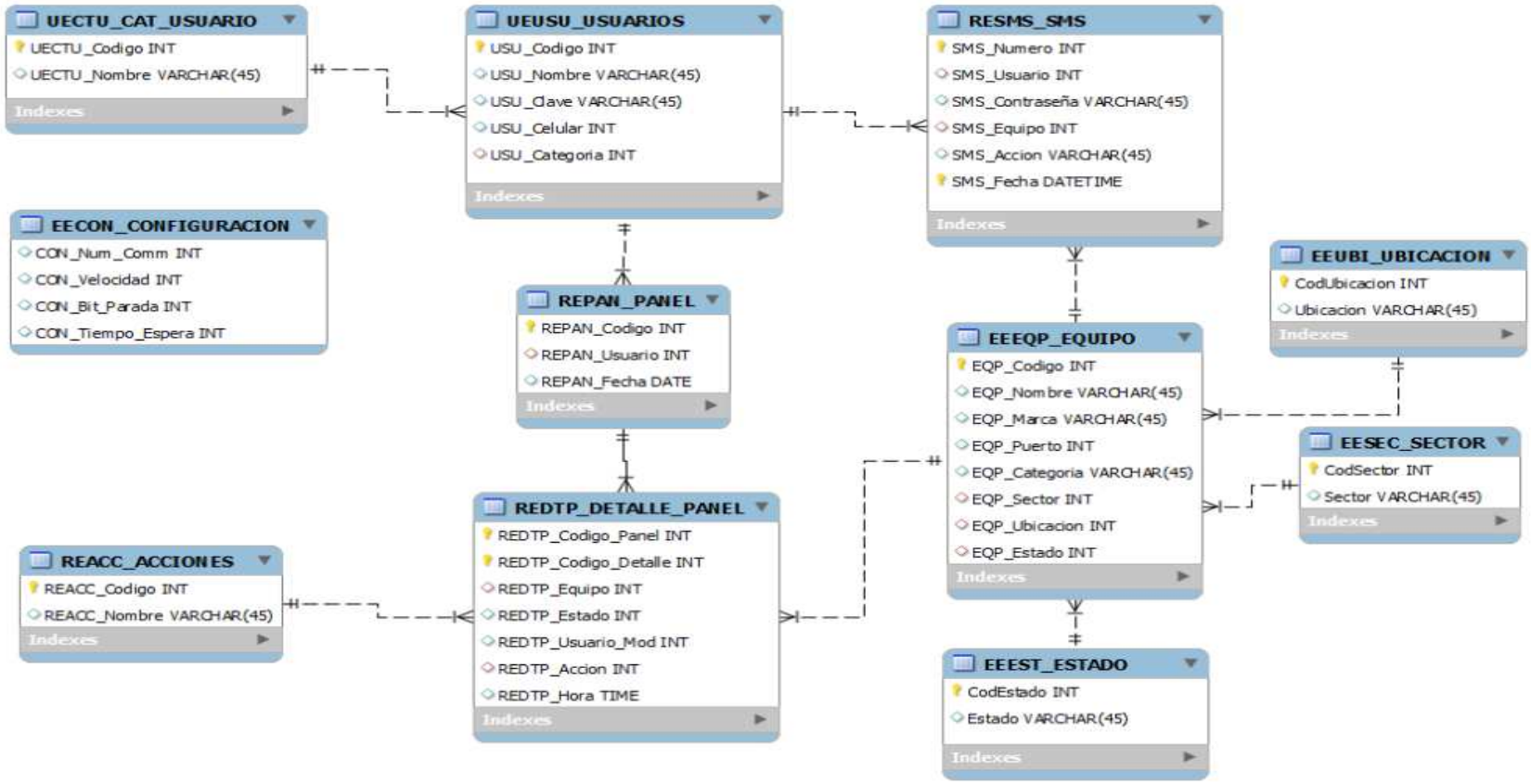

Figura 4. 4: Modelo físico del sistema

4.4.8. Tarjetas CRC

Usar tarjetas CRC, "Clase, Responsabilidad, Colaboración", en el diseño, ayuda a la describir de mejor manera las clases a implementar en el proyecto. Para el formato de las tarjetas CRC se ha estandarizado el siguiente esquema: El nombre de Clase en la parte superior, las responsabilidades (qué debe hacer) a la izquierda y los colaboradores (clases asistentes) a la derecha.

Para el desarrollo de las tarjetas CRC se tomo en cuenta las historias de usuario, y se enfocó la reutilización de clases genéricas que evitan la ambigüedad y el exceso de líneas de código innecesarias. Consecuentemente las tarjetas CRC propuestas son:

Tabla 4. 2 Tarjeta CRC Gestión de base de datos

Clase: ConexionBDD	
Responsabilidades	**Colaboradores**
- StrUsuario() - StrClave() - StrSql() - LeerConexionString() - ValidaUsuario() - EjecutarSqlNonQuery() - EjecutarSQLContraBaseDs() - EjecutarSQL()	Ninguna.

Tabla 4. 3 Tarjeta CRC Administración general

Clase: Madre	
Responsabilidades	**Colaboradores**
- LLenaParametrosModem() - LlenaCategoriaUsuarios() - EjecutaSQLContraBaseDs()	- Clase:ConexionBDD

- LeeConexion() - EjecutarSQL() - ListarUsuarios() - ConsultaEquiposNombre() - ListarEquipos() - ActualizarEquipos() - GrabarUsuarios() - ValidaUsuario() - NuevaCategoria()	

Tabla 4. 4 Tarjeta CRC- Panel general del sistema

Clase: Panel	
Responsabilidades	**Colaboradores**
- TraerValor() - ListarPerfiles() - PerfilesAcceso() - MeuSectores() - MenuEquipos() - ListarUsuarios() - ListarEquipos() - GrabarManejoEquipo()	- Clase: Madre - Clase: ConexionBDD

Tabla 4. 5 Tarjeta CRC- Gestión de usuario, acceso y perfiles

Clase: Usuarios	
Responsabilidades	**Colaboradores**
- StrUsuario() - StrClave() - Usuarios() - ValidarUsuario() - ListarUsuario() - ExisteCuenta()	- Clase: Madre - Clase: ConexionBDD

- Eliminar() - GrabarUsuario() - DesplegarMenu() - IngresaUsuario() - CifrarClave() - GetDatos() - ResetearClave()	

Tabla 4. 6 Tarjeta CRC - Interfaz

Clase: Interfaz	
Responsabilidades	**Colaboradores**
- Interfaz() - GenerarPanel() - Ejecuta_procedure() - Ejecuta_detalla() - ActualizaEquipos()	- Clase: ConexionBBDD - Clase: Madre. - Clase: Equipos.

CAPÍTULO 5
DESARROLLO DEL SISTEMA

5.1. Interfaz del prototipo

Para el desarrollo de la interfaz de la aplicación, se empleara patrones de diseño los cuales nos permitan generar un prototipo de pantalla para cada modulo.

Los patrones a utilizar son Fachada y Factory, porque estos nos permiten una integración de la interfaz rápida fácil y segura, además de la aplicación del Modelo Vista Controlador (MVC), la cual maneja la arquitectura del sistema de manera estructurada.

5.1.1. Patrón Fachada

El patrón Fachada proporciona una interfaz unificada de alto nivel para un subsistema, que oculta las interfaces de bajo nivel de las clases que lo implementan. Con esto se consiguen dos objetivos fundamentales: hacer el subsistema más fácil de usar y desacoplar a los clientes de las clases del subsistema.

Un cliente que trata de utilizar los servicios ofrecidos por un subsistema actuando directamente sobre las interfaces de las clases que implementan dicho subsistema se está exponiendo a los siguientes inconvenientes:

- Debe contener el conocimiento de cómo funcionan esas clases, comprender perfectamente la semántica de sus complejas interfaces de bajo nivel y saber cómo utilizarlas.

- Debe mantenerse informado de los cambios en dichas interfaces o en la distribución de responsabilidades entre las clases de bajo nivel y actualizar su código adecuadamente.

La utilización del patrón fachada nos permite una rápida conexión entre las actividades del usuario dentro del sistema.

5.1.2. Patrón Factory

Su uso es práctico para realizar consultas, basándonos en clases genéricas cuyos atributos cambian, con ello la programación se vuelve más ligera, es decir ahorra varias líneas de código al tener una sola clase “Consulta”, por ejemplo, en el sistema equipos, se requiere buscar los campos de tipos estados, motores, placa, entre otros; el implementar una serie de clases para generar la consulta de cada uno de los ítems, demandaría un sin número de líneas de código ya que todas las clases se implementarían en la misma interfaz y tendrían la misma estructura, es entonces que conviene el uso de una clase genérica que administre todo este tipo de consultas siguiendo el patrón Factory.

5.1.3. Modelo Vista Controlador

Para llevar una organización estandarizada de la codificación del sistema, se emplea el patrón de diseño, “modelo vista controlador” o MVC, con el cual, se separa adecuadamente la interfaz grafica, la lógica del negocio y la conexión con la base de datos.

La lógica de una interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio; por lo que, si se realiza un diseño sin considerar una distribución adecuada, los componentes de la interfaz y del negocio se verían entrelazados; dando como consecuencia que, cuando sea necesario cambiar la interfaz, habría que modificar los componentes de negocio, lo que implica un mayor trabajo y más riesgo de error.

De manera resumida el patrón MVC contempla:

- Modelo: datos y reglas de negocio.
- Vista: muestra la información del modelo al usuario.
- Controlador: gestiona las entradas del usuario.

Enfocando el diseño propiamente de la interfaz gráfica con la que actúa el usuario, se ha manejado prototipos en los objetos visibles en pantalla que permitan estandarizar color, tamaño e imagen.

5.2. Módulo manejo de equipos

En el modulo se ingresaran nuevos equipos como también la actualización de los mismos, además de poder asignar el equipo a nuevos sectores dentro de la casa.

5.2.1. Ingreso de equipos

En primera instancia el sistema carga con todos los datos existentes en la base de datos que han sido solicitados por el respectivo módulo y los visualiza en los distintos elementos de la ventana.

Para el ingreso del nuevo equipo se deberá llenar todos los parámetros que indica el formulario como se puede observar en la siguiente figura 5.1:

Figura 5. 1 Manejo de equipos

La interfaz es muy intuitiva lo que permite que el usuario pueda tener un fácil manejo del sistema a la hora de ingresar, eliminar o modificar un equipo.

Al terminar de ingresar todos los parámetros requeridos por el sistema, se dará clic el botón “Nuevo” este a sus vez envía todos los datos a la clase “Cls_Equipos” la cual es la encarga de conectarse a la base de datos, luego de validar los distintos contenidos se registrara el nuevo equipo si todos los datos caso contrario se indicara el error pertinente.

5.2.2. Ingreso de sectores

Al igual que las demás ventanas esta se cargará con la información solicitada por el sistema, indicando de esta manera los distintos sectores existentes en la base de datos.

En esta ventana se puede realizar las operaciones de:

- Agregar sector.
- Modificar sector.
- Eliminar sector.

Figura 5. 2 Ingreso de sectores.

Al terminar de ingresar todos los parámetros requeridos por el sistema se dara click el botón "Nuevo" este a sus vez envía todos los datos a la clase "Cls_Equipos" la cual es la encarga de conectarse a la base de datos luego de validar los distintos contenidos, se registrara el nuevo sector o caso contrario se indicara el error pertinente

5.2.3. Ingreso de ubicación

En la pantalla de ubicación tenemos las siguientes opciones:

- Actualizar, la cual permite realizar cambios a ubicaciones ya existentes.

- Nueva, la cual permite el ingreso de nuevas ubicaciones las cuales serán usadas en un futuro en los datos del equipo.

La pestaña de nueva ubicación consta de una tabla que indica las ubicaciones existentes en el sistema como lo muestra la figura 5.3:

Figura 5. 3 Manejo de ubicación.

Todas las interacciones con la base de datos se realizan mediante el llamado de la clase "Cls_Equipo"

5.3. Módulo manejo de usuarios

5.3.1. Ingreso de usuarios

En la pantalla de manejo de usuarios se muestran las opciones:

- Nuevo, nos permite el ingreso de nuevos usuarios.
- Actualizar, nos permite modificar datos sobre los usuarios existentes.
- Eliminar, suprime un usuario existente.

En la parte inferior de la pestaña manejo de usuarios, se presenta una tabla con los datos de los usuarios existentes en el sistema.

Figura 5. 4 Manejo de usuarios

Todas las conexiones realizadas a la base de datos se realizan mediante la clase “Cls_User”.

5.3.2. Manejo de perfiles

Para poder tener delimitados los accesos de los diferentes usuarios, se asignan permisos los cuales son asociados los perfiles del usuario.

Según como se determine el perfil tendrá accesos a determinados campos del sistema.

El diseño del manejo del perfil se observa en la figura 5.5:

Figura 5. 5 Manejo de perfiles

Todo acceso a la base de datos se realiza mediante la clase “Cls_User”.

5.4. Módulo panel de control

En el módulo panel de control se llevara el registro de las actividades realizadas por los usuarios en los distintos equipos de la casa.

5.4.1. Manejo por sector

En el panel manejo por sector, se encuentran agrupados los equipos de acuerdo al sector donde se encuentran ubicados, para un fácil manejo de los mismos.

El sistema permite la manipulación de encendido o apagado de los distintos equipos.

El diseño de este frame se puede observar en la figura 5.6

Figura 5. 6 Equipos por sectores.

La conexión con la base de datos se realiza mediante la clase “Cls_Equipo” la cual esta encargada de toda interacción de la ventana indicada con la base.

5.4.2. Manejo por equipo

En este panel se permite la selección del equipo que se necesite indiferente cual sea su ubicación o sector.

En la pestaña del sistema se muestran los distintos equipos ingresados, con sus principales características.

En la figura 5.7 se puede observar los distintos elementos gráficos de la ventana.

Figura 5. 7 Manejo de equipos por grupo.

La conexión con la base de datos se realiza mediante la clase “Cls_Equipo”.

5.5. Módulo reportes

5.5.1. Reporte por equipos

Para realizar un reporte se tiene que seleccionar un equipo del combo box y dar clic en ver reporte.

De acuerdo al parámetro de búsqueda la tabla se llenará con información existente en la base de datos.

Figura 5. 8 Reporte por equipo

5.5.2. Reporte por usuarios

Para ver el reporte se selecciona un usuario del combo box, y se da clic en ver reporte. A continuación en la tabla mostrará las acciones realizadas por el usuario buscado.

En la figura 5.9 se puede ver un reporte de usuario.

Figura 5. 9 Reporte por usuarios.

5.5.3. Reporte por fecha

En el panel se muestra todas las actividades realizadas en determinado rango de fechas, como lo indica la figura 5.10

Figura 5. 10 Reporte por fecha.

5.6. Clases para la conexión

5.6.1. Clase usuario

La clase usuario "Cls_User" permite la intercomunicación de las diferentes interfaces del módulo usuarios con la base de datos, teniendo de esta forma transacciones de manera segura.

Parte del código de la clase es el siguiente:

```
package com.domotica.implementaciones;
import com.mysql.jdbc.Statement;
import com.mysql.jdbc.Connection;
import com.mysql.jdbc.ResultSetMetaData;
import java.awt.List;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;
public class Cls_User {

private Statement sentencia  = null;
private Statement sentencia1 = null;
    private Conexion  cn;
    private String    respuesta;
    private String url;

    public Cls_User() throws Exception {
    cn = new Conexion();
    }
  public void Nuevo_User(String User,String Nombre,String Clave,String
Num_Celular,String TipoUser) throws Exception
{
Conexion conex = new Conexion();

int Id_User = Integer.parseInt(User);
int Celular_User = Integer.parseInt(Num_Celular);
int Tipo_User = Integer.parseInt(TipoUser);

String Sentencia = "INSERT INTO user
(Id_User,Nom_User,Clave_User,Celular_User,Tipo_User) VALUES (" +Id_User +
","'+Nombre +"','"+Clave+"',"+Celular_User+","+Tipo_User+")";
conex.Sentencia(Sentencia);
```

5.6.2. Clase equipo

La clase “Cls_Equipo” es la encargada de intercomunicar las diferentes interfaces del módulo equipo con la base de datos de manera segura.

Parte del código de Cls_Equipo es la siguiente:

```
package com.domotica.implementaciones;

import com.mysql.jdbc.ResultSetMetaData;
import com.mysql.jdbc.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 *  */
public class Cls_Equipo {
private Conexion  cn;
   public Cls_Equipo() throws Exception {
      cn = new Conexion();
   }
 //**********************************Clase para crear nuevo equipo con todos los parametros*********************************************
   public void Nuevo_Equipo(String idEquipo,String NombreEquipo,String MarcaEquipo,String PuertoEquipo,String TipoEquipo,String SectorEquipo,String UbicacionEquipo,String Motor1,String Motor2,int estado) throws Exception
{
```

```
Conexion conex = new Conexion();

int Id_Equipo = Integer.parseInt(idEquipo);
int Puerto_Equipo = Integer.parseInt(PuertoEquipo);
int Sector_Equipo = Integer.parseInt(SectorEquipo);
int Ubica_Equipo = Integer.parseInt(UbicacionEquipo);
int MotorI = Integer.parseInt(Motor1);
int MotorII = Integer.parseInt(Motor2);
// podria colocar un incremental en el id del usuario

String Sentencia = "INSERT INTO equipo (Id_Equipo,Nom_Equipo,Marca_Equipo,Puerto_Equipo,Tipo_Equipo,Sector_Equipo,Ubica_Equipo,Estado_Equipo,Motor1,motor2) VALUES (" +Id_Equipo + ",'"+NombreEquipo+"','"+MarcaEquipo+"',"+Puerto_Equipo+",'"+TipoEquipo+"',"+Sector_Equipo+","+Ubica_Equipo+","+estado+","+MotorI+","+MotorII+")";
conex.Sentencia(Sentencia);

}
```

5.6.3. Clase sms

La clase sms nos permitirá la recepción de los mensajes asi como su lectura y validación a la hora de realizar las acciones solicitadas mediante los mensajes de textos.

Parte del código de la clase es el siguiente:

```
package com.domotica.implementaciones;

//~--- non-JDK imports --------------------------------------------------------
import app.Com;
import app.Parameters;
```

```
import core.SerialPort;

//~--- JDK imports ------------------------------------------------------------

import java.awt.*;
import java.awt.event.*;

import java.io.*;
import java.io.File;    // para manejo de archivos

import java.util.ArrayList;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.swing.*;

public class MainSms {

    static String Res;
    static Com comac;
    private String dato1;
    private boolean estado;
    private SerialPort free;
    private ArrayList<String> portList;
    private Parameters settings;
    private boolean Estadom;

    public MainSms() throws Exception {
```

```
        free = new SerialPort();
        estado = true;

//      settings = new Parameters();
//      settings.setPort("COM6");
//      settings.setBaudRate("9600");
//      settings.setStopBits("1");
//      comac = new Com(settings);    }
```

CAPÍTULO 6

PRUEBAS E IMPLEMENTACIÓN DEL SISTEMA

Teniendo en cuenta que las pruebas en el uso de la metodología XP es el pilar básico, pues a la par que se va desarrollando el sistema se realizan las pruebas del funcionamiento de los códigos que se van implementando,es decir, las pruebas del sistema se realizaron en el desarrollo del mismo y no en la finalización, para garantizar la participación permanente de los usuarios; cabe recalcar que aplicar tests en la fase de desarrollo, es el único modo de garantizar que todo funciona como debe, e inclusive es posible llegar a hacer cambios más o menos importantes sin miedo a problemas inesperados, dado que proporcionan una red de seguridad. La existencia de tests hace el código muy flexible.

6.1. Técnicas de prueba

6.1.1. Prueba de aceptación

Para asegurar el funcionamiento de una historia de usuario particular, se han desarrollado test de aceptación; estos a su vez son creados y utilizados por el cliente, con la finalidad de comprobar que se cumplan a cabalidad las funcionalidades que describe cada historia.

El usuario es el responsable de revisar el funcionamiento óptimo de cada una de las interfaces creadas y las funciones que en ella se generan.

Al tener la aprobación de su funcionalidad, se da por cerrada dicha historia de usuario, con ello se garantiza la participación activa del cliente en el proceso de programación e implantación del sistema. Conforme se realizan las historias de usuario, se concede algunas horas de prueba previa a la implantación de las diferentes interfaces y procesos en el servidor.

Las pruebas fueron generadas en el transcurso del desarrollo del sistema; es decir, conforme se iban mostrando las diferentes interfaces, con ello se realiza un ciclo continuo de trabajo en el que el usuario no requirió esperar la consecución total del sistema para utilizar los módulos e interfaces ya terminadas y aceptadas.

Las pruebas de aceptación realizadas se muestran a continuación:

Tabla 6. 1 Prueba de Aceptación para la historia de usuario; Administrar Usuarios y perfiles

Historia de Usuario de referencia: Administrar Usuarios y perfiles
Precondiciones: Luego de que el administrador asigne al usuario su clave, el sistema encriptará la misma para evitar que esta sea alterada.
Flujo Principal: El administrador habilita un perfil de usuario a cada miembro de la casa que manejará el sistema, le otorga un nombre de usuario y clave personal.
Entradas (Sub-flujos): - El administrador autorizado ingresa a la pantalla de administración de perfiles, escribe la información del nuevo usuario y le asigna un perfil. (E-1). Habilitado el perfil, todos los operadores podrán acceder al sistema y visualizar los módulos que le competen. Para el acceso al sistema, se presenta al operador, la Pantalla Principal, en donde debe ingresar por teclado su nombre y contraseña. Una vez validado el operador (E-1) se continúa con la pantalla de menú principal. **Excepciones** E-1 no hubo validación: El login/password no se validó correctamente. Se solicita al usuario volver a registrarse.
Resultado esperado: 1. El sistema permitirá que solo el administrador del sistema cree nuevo perfiles y los asigne a cada tipo de usuario. 2. Solo administradores de sistema pueden editar perfiles y usuarios. 3. El sistema deberá encriptar la clave inmediatamente después del primer ingreso.
ACEPTACIÓN: El sistema cumple con el resultado esperado satisfactoriamente.

Tabla 6. 2 Prueba de Aceptación para la historia de usuario; Ingresar y/o consultar información del usuario

Historia de Usuario de referencia: Ingresar y/o consultar información de usuarios
Precondiciones: El operador debe haber ingresado al sistema con un perfil de acceso. Para ello, ingresará su usuario y su clave en la pantalla de ingreso de Usuario
Flujo Principal: El operador con permisos consulta los datos de otros usuarios existentes en la base.
Entradas (Sub-flujos): El operador en la pantalla de consulta de usuarios, registrará parámetros de búsqueda del usuario, dependiendo de los parámetros se continuarán con los sub-flujos correspondientes a esta historia de usuario. ***S*-1 Confirmar información en el sistema** El operador en la pantalla de consulta de usuarios obtiene un listado con todos los registros en el que el parámetro de búsqueda indicado coincida en la base de datos de usuarios. Si el operador encuentra el registro, pero parte de la información varía o ha sido actualizada por el usuario, se continúa con el sub-flujo Actualizar Información (S-2). Si el operador no encuentra el registro del usuario, se puede continuar con el sub-flujo Registrar Nuevo cliente (S-3). **S-2 Actualizar información** Permite al operador editar los datos incorrectos que se muestran en el registro, estos datos pueden referirse a campos como nombre, apellido, dirección, celular, entre otros. Editados los campos, el operador puede seleccionar entre las siguientes actividades: "Grabar " y "Menú". Si selecciona “Grabar”, el sistema almacena la información actualizada en la base de datos. Si la actividad seleccionada es "Menú" se regresará al a pantalla del menú principal. (Si aún no se ha presionado "Grabar", la información se perderá). **S-3 Registrar nuevo usuario** Si el operador no encuentra al cliente en la base de datos, toma la opción del menú “Nuevo usuario” (P-2), esta opción direccionará a la pantalla Manejo de usuarios, esta pantalla contiene información de registro que debe ser llenada por el operador, incluye nombre, apellido, teléfonos, etcétera. El operador puede seleccionar entre las siguientes

actividades: "Grabar " y "Regresar". Si el usuario selecciona “Grabar”, el sistema genera un nuevo registro de clientes. Si la actividad seleccionada es "Regresar" se regresará al a pantalla del menú principal. (Si aún no se ha presionado "Grabar", la información se perderá).
Resultado esperado: 1. El sistema verifica la existencia del usuario que solicita un servicio, en caso de no encontrarlo el sistema actualizará la base de datos con la información ingresada del nuevo cliente. **2.** El sistema consulta la información solicitada en la base de datos, y devuelve la información referente a nombre del cliente, celular y la muestra en pantalla.
ACEPTACIÓN: El sistema cumple con el resultado esperado satisfactoriamente.

Tabla 6. 3 Prueba de Aceptación para la historia de usuario; Generar registro de activaciones

Historia de Usuario de referencia: Generar reportes de control
Precondiciones: Se requiere indicar el número de la registro de activaciones. Se necesita su ingreso y poder realizar las activaciones de equipos solicitados.
Flujo Principal: El sistema valida el ingreso de usuario, si este es valido entonces registra el evento lo valida y realiza la accion solicitada por el usuario**.**
Entradas (Sub-flujos): Se presenta al usuario la pantalla de manejo de equipos, el cual tiene el menú d emanejo por sector o por equipo, de los cuales ell usuario elije. **S-1 Selección de equipo:** Independiente del panel seleccionado se podrá activar el equipo por medio del código del equipo, en cada panel se muestra el estado actual del equipo **S-2 Ingreso de evento:** El operador al visualizar el estado del equipo, dependediendo de esto para tomar la decisión de que evento realizar
Resultado esperado: 1. El sistema podrá consultar en la base de datos la información de equipos registrados.

2. El sistema podrá consultar en la base de datos la información del equipo seleccionado. 3. El sistema deberá poder actualizar el estado del equipo y realizar el evento. 4. El sistema almacena la información del evento realizado confirmada en la base de datos.
ACEPTACIÓN: El sistema cumple con el resultado esperado satisfactoriamente.

Tabla 6. 4 Prueba de Aceptación para la historia de usuario; Registrar eventos atreves de SMS

Historia de Usuario de referencia: Registrar eventos provenientes de sms
Precondiciones: Se requiere poder recibir mensajes de texto los cuales son enviados por usuarios con eventos a realizar en la casa
Flujo Principal: El sistema recepta los datos del mensaje de texto, valida y realiza el evento.
Entradas (Sub-flujos): El sistema esta activo receptando mensajes de texto, los cuales valida de acuerdo a la información que este contenga. **S-1 Validacion de usuario** Del mensaje de texto recibido se obtiene los datos, se compara el usuario y clave del mensaje con los existentes en la base de datos, si la validación es correcta entonces pasamos a (S-2), caso contrario (S-5) **S-2 Consulta de equipo** Continuando con la lectura de datos receptamos el código del equipo y el evento a realizar. **S-3 Activacion del evento** Al revisar que el equipo este registrado entonces realizamos el evento solicitado y lo guardamos en la base de datos **S-5 Borrar mensaje** Al terminar la lectura del mensaje y realizar los eventos solicitados realizamos un barrido del buzon de entrada.

Resultado esperado: 1. El sistema consultara los datos de usuario para validación. 2. El sistema permitirá la lectura del mensaje. 3. El sistema permitirá registrar los eventos solicitados via SMS. 4. El sistema permitirá guardar detalles en la base de datos.
ACEPTACIÓN: El sistema cumple con el resultado esperado satisfactoriamente.

Tabla 6. 5 Prueba de Aceptación para la historia de usuario; Registrar nuevo equipo

Historia de Usuario de referencia: Registrar equipos
Precondiciones: Se requiere que el equipo no se encuentre registrado en la base de datos para evitar ambigüedad de información. Se debe contar con toda la información que referencie al equipo, de preferencia mediante un documento escrito como la factura del proveedor por ejemplo.
Flujo Principal: El operador al actualizar el inventario, observa que uno o mas equipos no consta en la base de datos actual, y procede a registrarlo con su información respectiva**.**
Entradas (Sub-flujos): - En la pantalla de “Manejo de Equipos”, el operador se ingresa toda la información relevante del nuevo equipo a registrar. - El sistema asigna un código que será asignado al artículo registrado. - El operador, luego de registrada la información, puede seleccionar entre las actividades; “Guardar” y “Menú”.
Resultado esperado: 1. El sistema permitirá registrar la información referente a un equipo. 2. El sistema automáticamente generará un código que será asignado al nuevo equipo. 3. El sistema permitirá visualizar un listado de los equipos que constan al momento en la base de datos.
ACEPTACIÓN: El sistema cumple con el resultado esperado satisfactoriamente.

Tabla 6. 6 Prueba de Aceptación para la historia de usuario; Generar reportes de control

Historia de Usuario de referencia: Generar reportes de control
Precondiciones: Dependiendo el tipo de reporte solicitado, se requiere el ingreso de fecha inicial y final. Cada reporte es accedido por el perfil para el que fue creado, los demás perfiles no pueden acceder a dicho reporte.
Flujo Principal: El operador solicita diferentes reportes para conocer estadísticas que ayuden a administrar de mejor manera los recursos.
Entradas (Sub-flujos): - Se presenta al operador en el menú principal (P-2) las opciones que identifican las actividades, dentro de cada actividad se encontrará inmersa un submenú que identifica un determinado reporte. - El operador seleccionará el reporte que necesita consultar e ingresará a la pantalla del reporte respectivo. - Todas las pantallas de reporte solicitarán parámetros de búsqueda que agiliten los diferentes procesos de consulta, entre los más importantes están parámetros como fecha inicial, fecha final, código de material, código de técnico, entre otros.
Resultado esperado: 1. El sistema permitirá visualizar en pantalla los reportes solicitados, verificando previamente el ingreso de un determinado parámetro que sirva de filtro en la búsqueda. 2. El sistema permitirá imprimir l los reportes mostrados en pantalla.
ACEPTACIÓN: El sistema cumple con el resultado esperado satisfactoriamente.

6.1.2. Prueba de unidad

El objetivo de la utilización de las pruebas de integración, es verificar la funcionalidad, el rendimiento y la fiabilidad de las necesidades en los principales elementos de diseño. Los escenarios de prueba se construyeron para probar que todos los componentes interactúan correctamente dentro de su respectivo módulo. Por ejemplo, se comprobó que al generar una factura, el material facturado se actualice automáticamente en el stock, y a su vez esta se

encuentre en un estado pendiente de pago, de forma que al momento de realizar su cancelación se registre en el sistema adecuadamente.

Al trabajar en equipo, se optimizó tiempos al dividir el trabajo de codificación, se distribuyeron las funciones a implementar, los métodos y las interfaces que se mostrarán; de cada módulo se fraccionó equitativamente el trabajo permitiendo que cuando se unan en un solo sistema el funcionamiento integral sea notorio y se verifique que el sistema se acopla a los requerimientos solicitados y sus módulos trabajan en conjunto. La integración permitió que los procesos que realizan todos los departamentos por separado, lleguen a concretarse en uno solo, permitiendo que las salidas de uno sean las entradas del otro. Para llevar un control de la integridad de los módulos se desarrollo un listado con todas las actividades de integración que debían ser cumplidas por el sistema:

Tabla 6. 7 Validación de procesos cumplidos

Núm.	Descripción	Cumplido
1	Ingreso de usuarios nuevos con permisos.	Si
2	Toda accion realizada tiene registro.	Si
3	Todo detalle debe tener un número de orden como referencia	Si
4	El número de orden, y detalle debe ser generado automáticamente por el sistema.	Si
5	Todo ingreso es validado.	Si
6	Se lee todos los mensajes de texto pero se valida las acciones.	Si
7	Consultar los usuarios y sus acciones.	Si
8	Actualizar equipos.	Si

CAPÍTULO 7

CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

- Independientemente del tipo de sistema que vaya a ser desarrollado que puede ser de escritorio, web o para dispositivos móviles se debe identificar y validar los requerimientos funcionales que indican el que va a ser desarrollado y los requerimientos no funcionales que indican cómo se los va a desarrollar.

- Es necesario siempre identificar con que metodología va a ser desarrollada la aplicación sea tradicional o ágil orientada a objetos, orientada a la web o para dispositivos móviles para posteriormente seleccionar las herramientas con las cuales se va a desarrollar el aplicativo.

- El aprovechar una arquitectura en tres niveles que organice eficazmente el esquema de la aplicación a generar, permite tener una visión global del sistema logrando integrar patrones de diseño que mejoran la navegabilidad y presentación de la información en cada uno de los módulos desarrollados; en especial, el uso del patrón Factory, nos permitió realizar una programación más ligera.

- Teniendo como vía de comunicación el servicio de SMS, brindado por una operadora de telefonía móvil nacional, el servicio de control remoto del sistema queda ligada a la cobertura de la operadora.

7.2. Recomendaciones

Es recomendable:

- Poder interactuar con el usuario de manera directa permite la mejora del sistema, por lo cual siempre es necesario tener un buen análisis de requerimientos que quede plenamente identificados, recolectados, analizados y lo más importante validados por el mismo.

- Siempre tener un historial de las versiones que se van realizando para que los demás programadores pueden revisar y actualizar sus sistemas a cargo, el historial debe ser claro y compacto.

BIOGRAFÌA

INCA REA PAUL FERNANDO

Dirección	El dorado, Yaguachi e Iquique N 15-26
Teléfono	022-234422/ 095447253
Fecha de nacimiento	Cajabamba, 21 de enero de 1986
Estado civil	Soltero

EDUCACIÒN

Primaria

Unidad educativa “Combatientes de Tapi”

Secundaria

Pedro Vicente Maldonado

Superior

Sangolquí, Escuela Politécnica del Ejercito, Facultad de Ingeniería en Sistemas e Informática.

Proyecto de grado

Diseño y desarrollo de un prototipo de control mediante SMS para casas inteligentes.

Cursos

2009 Quito, ESPE, Facultad de Idiomas- Suficiencia en el idioma Ingles

Sangolquí, ESPE, Primer a cuarto módulo CCNA, CISCO

HOJA DE LEGALIZACIÓN DE FIRMAS

ELABORADO POR

_____

Paul Fernando Inca Rea

DIRECTOR DE LA CARRERA

_____

Ing. Mauricio Campaña

Sangolquí, 10 de mayo del 2012