Tesina de Máster

Time Machine: Desarrollo de una herramienta para la gestión de configuraciones en sistemas de Computación Autónoma

Septiembre 2011, Valencia

Master en Ingeniería del Software, Métodos Formales y Sistemas de la Información

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Autor

Alberto Naranjo Martín

Director

Vicente Pelechano Ferragud

Carlos Cetina Englada

AUTOR

Alberto Naranjo Martín

alnamar2@ei.upv.es

DIRECTOR

Vicente Pelechano Ferragud

pele@dsic.upv.es

Carlos Cetina Englada

ccetina@dsic.upv.es

Tabla de contenido

Listado de Figuras

Listado de Tablas

1.Introducción

1.1. Introducción

En los últimos años la evolución de la tecnología ha ido creciendo exponencialmente. Lo que hace unos años era impensable hoy en día es una realidad que convive con nosotros y comparte nuestro mismo entorno. La evolución es tal que día a día las diferentes tecnologías están más presentes en nuestras vidas, hasta el punto que dependemos de ellas totalmente para nuestra vida diaria y para hacer nuestras tareas de forma más sencilla.

Además los nuevos dispositivos cada día aportan nuevas funcionalidades hasta el punto de interactuar o depender del entorno para su funcionamiento y su variabilidad. Esto hace que los dispositivos tecnológicos acaben funcionando de una forma autónoma sin que nos demos cuenta. Se adaptan a la situación automáticamente para hacernos la vida más fácil. La idea básica es que no necesiten la interacción directa del usuario para poder funcionar o modificar su comportamiento, si no que sea él automáticamente el que se adapte.

Estos avances, como todo, tienen sus ventajas e inconvenientes. Entre sus ventajas la principal es la de facilitarnos la tarea al usuario no teniendo que gestionar los sistemas y permitiéndonos disfrutar de la tecnología sin que realmente seamos conscientes de que está entre nosotros. Si hay un error sabemos que se va a reconfigurar solo, si insertamos un dispositivo nuevo sabemos que lo va a detectar y lo va a introducir en el sistema.

Pero puede llegar el momento en el que queramos tomar nosotros el mando. Bien por necesidad porque se ha producido un error que la maquina no sabe solucionar. O simplemente por deseo nuestro, ya que hay veces que también necesitamos plasmar nuestros deseos y controlar a la maquina.

Además actualmente, hasta los mejores sistemas cuentan con un sistema de copias de seguridad. Nunca se sabe lo que puede pasar y siempre es mejor tener copias de seguridad de

lo que hemos ido haciendo en el tiempo. A veces la mejor forma de arreglar algo es volver a atrás en el tiempo. De momento no podemos retroceder las manecillas del reloj pero si podemos guardar un histórico de los cambios de nuestro sistema para, llegado el momento, volver a un punto determinado en el tiempo y restaurar el sistema.

La presente tesis de máster constituye una primera aproximación a un sistema de gestión para las copias de seguridad de un sistema de Smart-Home. El modelo que se presenta en esta tesis tiene como finalidad la definición de un sistema que, dado los cambios que se pueden producir en una Smart-Home y las diversas reconfiguraciones que sufre el sistema, gestione de una forma eficaz las diversas versiones que se guarden de la configuración del sistema de Smart-Home.

Alguno de los ejemplos en los que será útil nuestra Time Machine será por ejemplo cuando el usuario quiera volver a un estado anterior de la configuración, ya sea por error del sistema o por deseo del usuario. Como cualquier sistema, la Smart-Home interactúa con varios usuarios y es importante también guardar un registro por lo que pueda pasar. Además se ofrecerá al usuario una interfaz para interactuar con la Smart-Home de forma que el usuario pueda modificar en caso de que la Time Machine tome una decisión que no sea la deseada.

Se pretende dar un enfoque práctico presentando una herramienta funcional y sencilla de usar de cara al usuario, que sea capaz de gestionar las versiones del sistema, modificarlas y seleccionarlas, modificando así la configuración del sistema al gusto o las necesidades del usuario.

1.2. Objetivo de la tesina

Los sistemas que funcionan de forma autónoma se caracterizan por ser capaces de reconfigurarse y adaptarse al entorno sin parar la ejecución y sin la intervención del usuario. De esta forma el sistema se irá adaptando al entorno conforme este vaya cambiando sin interrumpir el servicio al usuario.

Debido a la autogestión que hace el sistema de si mismo se producirán muchos cambios para adaptarse a la situación ya sea por cambios en el entorno o por una corrección de fallos. Esto puede provocar que el sistema sufra numerosos cambios en poco tiempo produciendo que se llegue a un estado no deseado, bien por preferencias del usuario o por que se trate de un estado de error en el que el sistema no funciona correctamente.

Hoy en día tan importante es tener un buen sistema que funcione correctamente como mantener a salvo una copia de este sistema para recuperarlo, ya no solo en caso de error como hemos dicho antes, si no por deseo del usuario. Puesto que los cambios en este tipo de sistemas son muy frecuentes muchas veces no sirve con volver a la última copia valida

si no que tenemos que mantener un histórico con las diferentes versiones y los cambios que ha sufrido nuestro sistema.

Además dadas las características de automaticidad y transparencia de estos sistemas, las decisiones que toma el sistema para reconfigurarse pueden no ser siempre las deseadas. Aunque el objetivo máxime de estos sistemas es el de la autonomía y el de la transparencia al usuario, este siempre debe tener, aunque sea, una mínima cooperación con el sistema de forma que pueda expresar sus preferencias que, como ya hemos dicho, pueden no estar siempre en total sincronía con las elecciones hechas por el sistema.

Cuando se plantea este trabajo se marca como objetivo:

1) Proporcionar un mecanismo que permita guardar las distintas configuraciones, consultarlas y recuperarlas en caso de que se desease reconfigurando el sistema al estado en el que estaba en el momento que nosotros le indicamos.
2) Desarrollar una herramienta útil y fácil de usar para el usuario de forma que mediante un panel táctil pueda consultar las configuraciones del sistema en el pasado, clasificadas por fechas, e incluso modificarlas y adaptarlas para reconfigurarlo con sus preferencias.
3) Demostrar mediante el caso de estudio la aplicabilidad de la propuesta en un entorno real y los múltiples usos y ventajas que aporta. En el caso de estudio se plantearán escenarios concretos de uso donde podremos ejemplificar de forma clara las distintas utilidades de back-up y de restauración del sistema.

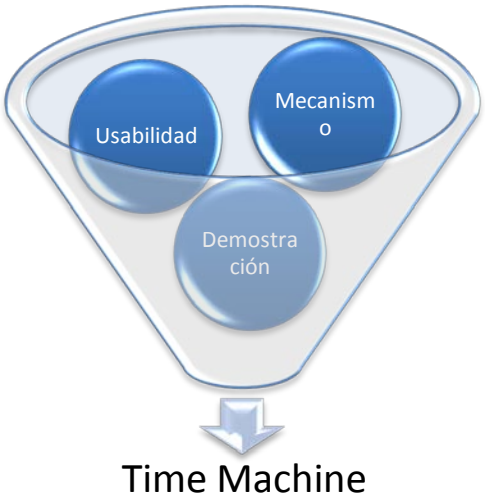

Figura 1 – Objetivos de la Time Machine

Para gestionar los cambios en el entorno se utilizan los modelos de variabilidad, que nos van a permitir definir las modificaciones que se deben aplicar sobre el sistema para adecuar su comportamiento en base a unas condiciones.

Los principios de la metodología de línea de productos y el paradigma del desarrollo dirigido por modelos también nos van a ayudar al desarrollo de esta aplicación. Por una parte la línea de productos nos proporciona la idea de la reutilización de componentes y de unificar las partes comunes para ser reutilizadas. Por otra parte el desarrollo dirigido por modelos automatizará la transformación de una especificación software abstracta en un producto software completamente funcional.

Con todo esto conseguiremos una aplicación para integrar con la Smart-Home y que nos permita que el usuario pueda interactuar con ella utilizándola para restaurar el sistema o modificarlo según su criterio.

1.3. Estructura de la tesina

En este apartado vamos a explicar resumidamente el contenido de los puntos siguientes de esta memoria de los que va a constar la tesina y como se va a organizar el documento.

Capítulo 2: Contexto.

En este apartado se va a presentar el contexto en el que se va a localizar el desarrollo de la tesina. Se va a explicar el problema que se pretende resolver y la forma en la que se va a solucionar dicho problema.

Capítulo 3: Visión general de la propuesta.

Este capítulo presenta las técnicas y las metodologías que se ven involucradas en este trabajo. Se van a presentar explicando sus características más importantes de forma que el desarrollo de la tesis sea más sencillo de comprender. Se presentarán también los conceptos clave para entender el proceso que se va a explicar en posteriores apartados. Además se va a explicar el desarrollo que se ha seguido para la elaboración de la tesina. Por último se presentan las tecnologías que se han utilizado para desarrollar la tesina.

Capitulo 4: Time Machine

En este capítulo se presenta el desarrollo completo y detallado de la propuesta. Centrándonos en las funcionalidades básicas y en cómo se han desarrollado. En este capítulo hablaremos de la funcionalidad que hemos diseñado en la Time Machine centrándonos en la interfaz gráfica y en los gráficos de configuración.

Capitulo 5: Diseño e Implementación de la Time Machine

En este capítulo se presenta el diseño que se ha seguido para el diseño de la Time Machine. Por una parte hablaremos del proceso que se ha seguido para el desarrollo de la tesina y por otra del diseño e implementación de la aplicación. Después pasaremos a hablar de como se integraría con la Smart-Home.

Capítulo 6: Caso de estudio.

Se va a presentar un caso de estudio para ejemplificar el uso de la Time Machine de forma que se muestre en unas situaciones concretas la funcionalidad del trabajo. Se va a presentar los detalles del entorno y del sistema sobre el que se va a probar la Time Machine y

se van a presentar una serie de escenarios para demostrar cómo se comportaría en cada caso y demostrar los distintos usos que se le puede dar al sistema desarrollado.

Capítulo 7: Conclusiones.

En este capítulo se presentan las contribuciones principales del desarrollo que se ha realizado, los resultados y conclusiones obtenidas.

Capítulo 8: Trabajos futuros.

Este capítulo expone las posibles futuras líneas de trabajo y los diferentes aspectos relacionados con la tesina que no se han abordado y que se podrían llevar a cabo en desarrollos futuros.

Anexo A: Detalle de Implementación de la Time Machine

En este capítulo hablaremos en detalle de la implementación realizada para la Time Machine indicando como se han desarrollado algunos métodos y clases de la Time Machine.

2.Contexto

Figura 2 - Alcance del capítulo 2

En este capítulo se sitúa el contexto en el que se plantea la tesina y el problema a resolver. Para ello primero se va a describir el problema a resolver, es decir, el problema de los continuos cambios y de la perdida de las configuraciones antiguas, explicando cuales son las motivaciones principales que han movido el desarrollo de la propuesta.

Después se va a describir la solución que se va a proponer para resolver dicho problema, que consiste en una aplicación que de solución al problema planteado.

2.1. Planteamiento del problema

Cada día los sistemas informáticos están más presentes en nuestras vidas y están más empotrados en nuestro entorno de una forma que muchas veces no nos damos cuenta de lo importante que pueden llegar a ser. Hay veces que hasta que no fallan no nos damos cuenta de lo que realmente tenemos a nuestro alrededor. En el tipo de sistemas que estamos tratando los cambios en el sistema son muy comunes ya que irá cambiando con el entorno. Cuanto más cambiante y mas usuarios interactúen con el sistema más cambios se efectuarán automáticamente en nuestro sistema y las probabilidades de error serán por tanto mayores también.

Dado que se trata de un software cambiante que dependerá de los sistemas externos se pueden producir errores que produzcan que nuestro sistema no funcione correctamente.

Estos errores pueden venir provocados por muchos factores, tanto externos como internos, y dado que el sistema puede cambiar de forma muy habitual, en ocasiones puede ser complicado determinar la causa del error si no contamos con una secuencia de las configuraciones que ha ido teniendo la casa. De esta forma nos será útil guardar las configuraciones que se vayan haciendo tanto automáticas como de usuario de forma que se pueda determinar la procedencia del error. Pudiendo volver a un estado anterior en el que sistema funcionase de una forma estable.

Además el usuario puede querer volver a un estado anterior, bien porque fuese de su agrado o porque la configuración que tenía en ese momento sea la deseada para ahora. Puesto que dentro de la Smart-Home interactúan diversos usuarios es importante tener un registro de las configuraciones que aplica cada uno. Bien para poder ajustarse a ella dependiendo del usuario o para que el usuario pueda modificarlas a su gusto. Hay que tener en cuenta que puede que las decisiones de reconfiguración que la Smart-Home tome no coincidan siempre con lo que desee el usuario en ese preciso instante. Por eso se plantea la necesidad también de que el usuario disponga de una interfaz con la Smart-Home mediante la cual pueda interactuar y cooperar con ella para poder adecuarla a su gusto.

2.2. Solución propuesta

Para solucionar el problema de la perdida de información y de su posible uso en un futuro se ha pensado en desarrollar un sistema que se integre con la Smart-Home y que de esta forma sea capaz tanto de guardar las configuraciones como de mostrarlas y aplicarlas a la casa en caso de que el usuario lo desease.

Como solución al problema planteado se va a utilizar la computación autónoma ya que una parte de la Time Machine deberá de ser totalmente automática y transparente al usuario. Esta parte interactuará con la Smart-Home y realizará un servicio de captura de los cambios guardando un histórico de estos.

Además para representar las configuraciones se van a utilizar conceptos de la teoría de modelos y de línea de productos que nos van a permitir trabajar con la configuración de la casa como de si un modelo se tratase modificando este cuando sea necesario. Utilizar estos modelos nos va a permitir dos cosas principalmente:

- Tener una representación gráfica que mostrar al usuario de forma que pueda ver de un solo vistazo cual es el estado, pasado o presente, de la casa.
- Un modelo que mediante operaciones de reconfiguración pueda pasar de un estado a otro de la casa de una forma rápida.

Esta aplicación deberá tener una interfaz de cara al usuario que cumpla una serie de requisitos. El primero y más importante será que sea atractiva al usuario y fácil de usar. Además deberá permitir al usuario interactuar con la casa permitiéndole modificar la configuración actual o eligiendo una configuración antigua. Por último deberá de ser totalmente funcional. No se debe caer en el error de hacer una herramienta tan sencilla que no tenga una funcionalidad completa.

3. Visión general de la propuesta

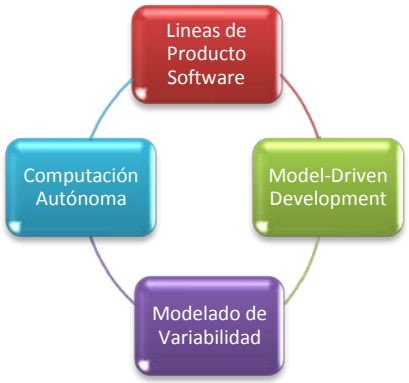

Figura 3 – Alcance del capítulo 3

En este apartado de la tesina se va a mostrar una visión panorámica de la propuesta. Para ello se van a definir los distintos elementos que la componen, presentando los conceptos principales necesarios para entender todo el desarrollo de la tesina que se presentará en el capítulo posterior.

Primero se van a describir las técnicas y ámbitos tecnológicos en los que se va a ubicar la propuesta. Se van a presentar los conceptos básicos de cada uno necesarios para entender el proceso de desarrollo en el que se enmarca la tesis.

Seguidamente se van a presentar los conceptos básicos explicados de una forma resumida, necesarios para entender el desarrollo explicado en los siguientes capítulos.

3.1. Ámbito tecnológico

En este apartado vamos a presentar las técnicas más importantes en las que se va a basar nuestra propuesta. Sobre estas metodologías y disciplinas es sobre las que se ha construido este trabajo.

Se denominan **sistemas autónomos** a aquellos sistemas de computación que pueden gestionar determinados objetivos de alto nivel de los administradores. La esencia de los

sistemas de computación autonómica es la autogestión, la intención es que el sistema esté libre de administradores para los detalles de las operaciones y mantenimiento del sistema y proporcionar a los usuarios un sistema que funciona a máximo rendimiento 24 / 7.

Modelo Dirigido por Modelos viene definido por el objetivo principal de que el desarrollo de software viene definido principalmente por modelos en lugar de por programas de ordenador. La principal ventaja es que expresamos los modelos usando conceptos que están menos ligados a la implantación tecnológica. Esto hace que los modelos sean mucha más fáciles de especificar comprender y mantener.

Las **líneas de producto software** es una estrategia para la producción de productos intensivos en software. La estrategia abarca la gestión organizacional, gestión técnica, y aspectos de ingeniería de software de la producción del producto.

Modelado de la variabilidad es una técnica de modelado que tiene como objetivo capturar los puntos de variación de un sistema evolutivo. Mediante esta técnica se pretende que el sistema por sí mismo pueda adaptar su comportamiento para dar respuesta a los cambios que se vayan sucediendo. Estos modelos son ampliamente utilizados en el desarrollo de software siguiendo la metodología de líneas de producto.

3.1.1. Computación autónoma

La computación autónoma recibe este nombre por una metáfora basada en la biología. El sistema nervioso autónomo del cuerpo humano es fundamental para un gran número de actividades inconscientes que nos permiten proceder con un mayor nivel de actividad en nuestra vida cotidiana. La tasa de latidos del corazón, la frecuencia respiratoria o el reflejo ante un objeto cortante o caliente son algunos de los ejemplos típicos. Con el uso de esta metáfora se pretende expresar la visión de lo que se quiere conseguir a nivel de computación, una autogestión de una gran parte de las funciones informáticas para aliviar a los usuarios de las actividades de gestión. Lo que le permitirá prestar más interés a otras preocupaciones de más alto nivel.

La necesidad y la justificación de los sistemas autónomos se basa en la complejidad cada vez mayor en los sistemas de hoy en día. Se ha expresado que la industria de tecnologías de la información (TI) ha enfocado sus esfuerzos en mejorar el rendimiento del hardware, con el software creciendo con características adicionales para maximizar esta capacidad adicional, en contra de otros criterios vitales.

Auto-configuración

La configuración, instalación e integración de grandes sistemas es un trabajo complejo y que consume mucho tiempo, además de ser muy susceptible de errores.

Los sistemas autónomos se configuran automáticamente en base a las políticas de alto nivel que especifican que es lo que se desea, no como se va a lograr. Cuando un componente se introduce este se incorpora a la perfección y el resto del sistema se adapta a su presencia.

Auto-optimización

Sistemas complejos como Websphere u Oracle pueden tener cientos de parámetros ajustables que se deben establecer correctamente para el óptimo funcionamiento del sistema. Sin embargo no todo el mundo sabe como configurarlos. Estos sistemas que normalmente se integran con otros, igual de complejos, puede hacer que el rendimiento de funcionamiento del sistema final no sea del todo óptimo.

Figura 4 – Características de la Computación Autónoma

Los sistemas autónomos por la contra, buscan constantemente formas de mejorar su funcionamiento, identificando y aprovechando las oportunidades para ser más eficientes. Los sistemas autónomos supervisarán, experimentarán y ajustaran sus propios parámetros para obtener un rendimiento óptimo.

Auto-reparación

Grandes empresas de TI cuentan con grandes departamentos dedicados a la identificación, seguimiento y determinación de la causa raíz de los fallos en los sistemas de información complejos. Grandes problemas pueden provocar que los equipos de programadores tengan que invertir largas jornadas de búsqueda, diagnóstico y solución del problema.

Los sistemas de computación autónoma en cambio son capaces de detectar, diagnosticar y reparar sus errores de software o hardware

Auto-protección

A pesar de los firewalls existentes y las herramientas de detección las personas aun deben de decidir la forma de proteger sus equipos ante ataques maliciosos. Los sistemas autónomos se auto protegen en dos sentidos. Por una parte defienden el conjunto del sistema contra los ataques y los problemas derivados de los fallos en cascada y además se anticipan a

los problemas de los sensores basándose en informes tempranos de estos y tomando medidos para evitar el fallo.

3.1.2. Desarrollo Dirigido por Modelos

Desde que los seres humanos comenzaron a usar los ordenadores, las investigaciones han estado trabajando para elevar el nivel de abstracción en la que los ingenieros de software escriben los programas. Desarrollo Dirigido por Modelos es la continuación natural de este trabajo. En lugar de exigir a los desarrolladores que expliquen todos los detalles de un sistema utilizando un lenguaje de programación, MDD les permite modelar lo que se necesita en cuando a la funcionalidad y lo que en general la arquitectura del sistema debe tener. Esto hace que los modelos sean más fáciles de especificar, comprender y mantener. Además hace que los modelos sean menos sensibles a la tecnología y a la evolución de esta. Por supuesto si los modelos se quedan meramente en documentación, su valor se ve muy limitado, ya que la documentación en muchos casos se aparta de la realidad. Por tanto una premisa importante detrás de MDD es que los programas se generen automáticamente a partir de dichos modelos.

Para que sea útil y eficaz, un modelo de ingeniería debe poseer, en grado suficiente, las siguientes cinco características clave:

- **Abstracción**: Un modelo es siempre una representación reducida del sistema que representa. Al remover u ocultar los detalles que no es pertinente para un punto de vista determinado, nos permite comprender la esencia más fácilmente. Teniendo en cuenta la constante demanda de funcionalidad cada vez más sofisticadas de nuestros sistemas de software, la abstracción es casi el único medio de hacer frente a la complejidad resultante.
- **Comprensibilidad**: es una función directa de la expresividad de la forma de modelización (expresividad es la capacidad de transmitir una idea compleja con poca información directa). Un buen modelo proporciona un acceso directo al reducir la cantidad de esfuerzo intelectual necesario para la comprensión. Una de las razones por qué los programas no son muy expresivos, incluso cuando se basa en lenguajes que admiten abstracciones sofisticadas, es que requieren un análisis demasiado detallado de texto para ser debidamente comprendido.
- **Exactitud**: Un modelo debe proporcionar una representación real de las características del sistema modelado.

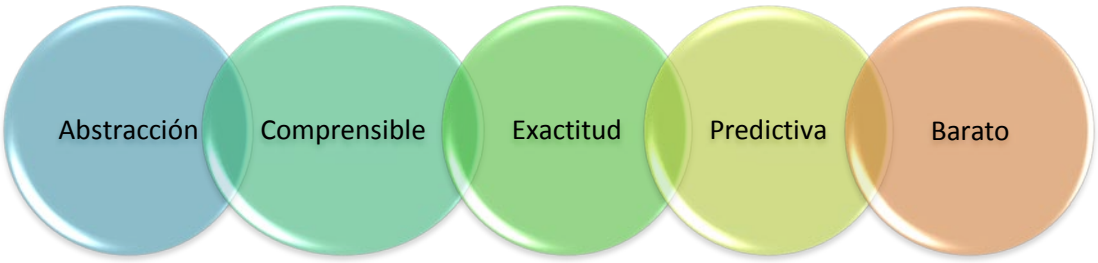

Figura 5 – Características del Desarrollo Dirigido por Modelos

- **Predictiva**: Se debe de ser capaz de utilizar un modelo para predecir correctamente las propiedades interesantes, pero no evidentes del sistema modelado, ya sea a través de la experimentación (por ejemplo, mediante la ejecución de un modelo en el ordenador) o a través de algún tipo de análisis formal.
- **Barato**: debe ser significativamente más barato de construir y analizar que el sistema modelado.

3.1.3. Líneas de producto Software

La estrategia de productos en línea se ha utilizado comúnmente en la fabricación de productos no perecederos, pero ha sido recientemente cuando ha empezado su influencia en los procesos de desarrollo de software. Un enfoque de línea de productos tiene por objetivo conseguir mejoras en la productividad y el tiempo de comercialización mediante el diseño de un conjunto de productos que tienen muchas partes en común. Así que esta es, en cierto sentido, un nuevo esquema de reutilización de software, pero es uno que ha demostrado ser eficaz en la experiencia industrial real.

El enfoque de línea de producto, también trata de identificar y gestionar las variaciones entre los productos. El éxito de la estrategia de producción de software en línea se debe, al menos parcialmente, a su carácter global.

Una línea de producto de software es un conjunto de sistemas intensivos en software que comparten un común, administrado conjunto de características que satisfacen las necesidades específicas de un determinado mercado o misión, y que se desarrolló a partir de un conjunto común de activos básicos de una manera prescrita, de acuerdo con la definición utilizada por el Software Engineering Institute (SEI) [Clements01].

La línea de producto software trata de producir familias de sistemas parecidas en lugar de producir sistemas individuales. La ingeniería de línea de productos consiste en tres procesos principales: ingeniería del dominio, ingeniería de la aplicación y gestión. Estos procesos son complementarios entre ellos.

Figura 6 – Elementos Líneas de Producto Software

Ingeniería del dominio

Se define como la actividad de coleccionar organizar y almacenar la experiencia pasada en el desarrollo de sistemas o partes del sistema en un particular dominio en forma de activos (arquitectura, modelos, código etc.), así como proveer una forma adecuada de reusar esos activos cuando se construye un nuevo sistema.

Ingeniería de la aplicación

Se denomina Ingeniería de la aplicación dentro de la Línea de Productos Software al proceso de desarrollar un sistema particular en un dominio. La ingeniera de la aplicación es responsable de derivar un producto concreto de la línea de producto software usando una aproximación de "diseño de reúso". Para conseguir esto, se reúsa los activos reusables que se han desarrollado antes.

Gestión

Se trata de un proceso separado donde activos organizados se manejan específicamente. Este proceso es el responsable de proveer los recursos, coordinación y supervisión a las actividades de ingeniería del dominio y de ingeniería de la aplicación.

3.1.4. Modelado de Variabilidad

El Modelado de la variabilidad se considera como la tecnología que permite ofrecer una amplia variedad de sistemas software de una manera coherente e integral. La clave está en construir una base con los elementos comunes y expresar y manejar la variabilidad de los sistemas. Una comunidad es un supuesto de manera uniforme a través de un conjunto determinado de sistemas. Con frecuencia, estos supuestos son los componentes con las

mismas especificaciones para todos los sistemas. Por el contrario, una variabilidad es una suposición verdadera sólo en algunos sistemas, como un componente con diferente especificación para, al menos, dos sistemas

Los Modelos de variabilidad nos permiten especificar no sólo las características actuales de un sistema, si no también tener en cuenta su potencial, ya que pueden ser activadas en el futuro. En respuesta a cambios en el contexto, el propio sistema puede consultar en estos modelos la variabilidad con el fin de determinar las modificaciones necesarias para su arquitectura.

3.1.4.1 MODELADO DE LA VARIABILIDAD Y LÍNEAS DE PRODUCTO SOFTWARE

El modelado de variabilidad está muy relacionado con las líneas de producto. Las líneas de producto software hacen referencia a métodos, herramientas y técnicas para crear y mantener una colección similar de sistemas software a partir de un conjunto de activos.

La línea de producto software puede ser descrito en términos de cuatro conceptos como se muestra en la Figura 82.

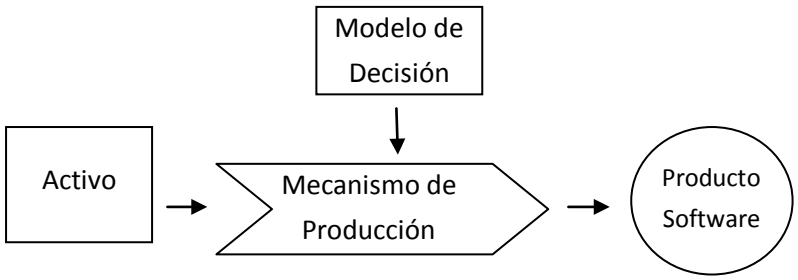

Figura 7 – Conceptos de la Líneas de Producto Software

- **Activo software de entrada:** una colección de activos software (requerimientos, código, arquitectura etc.) que pueden ser configurados y compuestos de diferentes formas para crear todos los productos de una línea de productos. Cada uno de los activos tiene un rol bien definido dentro de una arquitectura común para la línea de productos. Para dar cabida a la variación de los productos, algunos de estos activos pueden ser opciones y otros pueden tener puntos de variación interna que permitan configurarlos de distinta forma.
- **Modelo de Decisión:** Las decisiones describen las características opciones y variables para los productos en una línea de productos. Cada producto de la línea está únicamente definido por sus propias decisiones de producto.
- **Mecanismo de Producción y proceso:** Los mecanismos para componer y configurar productos desde los activos software de entrada. Las decisiones de producto se usan durante la producción para determinar que activo software de entrada se usa y como configurar los puntos de variación entre estos activos.

- **Producto software como salida:** La colección de todos los productos que pueden ser producidos en la línea de producción. El objetivo de las líneas de producción se determina por el conjunto de productos software de salida que pueden ser producidos por el software de entrada y por el modelo de decisión.

Los principales objetivos de las líneas de producto software son:

- **Capitalizar en común** mediante la consolidación y compartición dentro de los activos de un sistema, evitando así la duplicación y divergencia.
- **Gestionar la variación para reducir el tiempo**, esfuerzo, coste y complejidad de crear y mantener una línea de producción de sistemas software similares. Esto se logra definiendo claramente los puntos de variación y el modelo de decisión, con la ubicación y dependencias para una variación determinada.

En definitiva, la ingeniería de la línea de producción software (SPLE) optimiza el desarrollo de sistemas individuales dentro de una dominio de aplicación mediante la promoción de características comunes y la gestión de las diferencias de una forma sistemática. En la SPLE, sistemas individuales pueden construirse rápidamente a partir de activos reusables, como un conjunto de componentes y/o una plataforma común.

3.1.4.2 DESARROLLO DIRIGIDO POR MODELOS Y LÍNEAS DE PRODUCTO SOFTWARE

El desarrollo de software generativo y otras aproximaciones relacionadas como las de Factoría de Software, han ido propagando la integración de las líneas de producto software y el Desarrollo Dirigido por Modelos.

La ingeniería SPL y MDD no solo son complementarias, sino que además su integración su integración tiene un gran potencial. Mientras que MDD puede ayudarnos a representar diferentes aspectos de una línea de productos de forma más abstracta, SPL provee un objetivo de aplicación bien definido, que pone el desarrollo y la selección de lenguajes de modelado apropiados en una buena base.

Además, el análisis automático y la generación de código que ofrece mediante modelos precisos pueden ayudar a automatizar la creación de configuraciones de sistema. MDD provee técnicas efectivas para transportar los resultados de especificaciones de variabilidad de la siguiente forma:

- **Meta modelado**, que define tipos de sistema que expresan de forma precisa las principales características de sintaxis abstractas y las restricciones semánticas estáticas asociadas con la línea de producto para un dominio particular de la aplicación.
- **Domain-specific languages (DSL)**, provee notaciones que son guiadas por un meta modelo extendido para formalizar el proceso de especificación de la estructura de línea de producto, y los requerimientos en un dominio.
- **Transformaciones del modelo y generación de código** que garantizan la coherencia de las implementaciones de la línea de productos con información de análisis

relacionados con los requisitos funcionales y de calidad del servicio capturado por los modelos estructurales y de comportamiento.

3.2. Conceptos básicos

En este apartado se van a describir de forma detallada los conceptos básicos y la terminología que se va a utilizar para explicar el desarrollo del resto de la tesina. Nos vamos a centrar en los conceptos de Modelado de características y en la Arquitectura Dinámica de Línea de Productos.

3.2.1. Modelado de características

El modelado de características se utiliza para describir el conjunto de productos en una línea de productos de software en términos de características. En estos modelos, las características se vinculan jerárquicamente en una estructura en árbol a través de las relaciones de la variabilidad (opcional, obligatoria, opción única y de opción múltiple) y opcionalmente se conecta por las limitaciones como exige o excluye.

Las relaciones de variabilidad más comunes que se van a encontrar en nuestros modelos son los siguientes:

Relación opcional

La relación opcional, representada por la figura 8, indica que la activación de la característica hija es opcional aunque se active la característica padre.

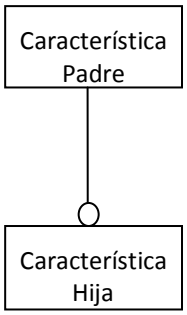

Figura 8 – Relación opcional

Relación obligatoria

La relación obligatoria, representada en la Figura 9, indica que al activarse la característica padre obliga a que se active la característica hija.

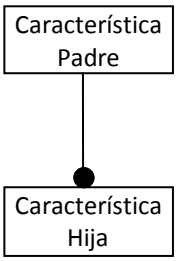

Figura 9 – Relación obligatoria

Relación de opción única

La relación de opción única, representada en la Figura 10, indica que la activación de la característica padre implica la activación de una y solamente una de las opciones disponibles entre las características hijas.

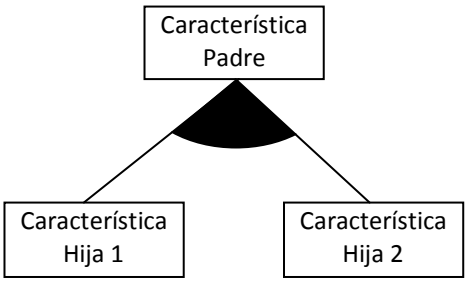

Figura 10 – Relación de opción única

Relación de opción múltiple

La relación de opción múltiple, representada por la figura 11, indica que la activación de la característica padre implica la activación de una o más de las opciones disponibles entre las características hijas.

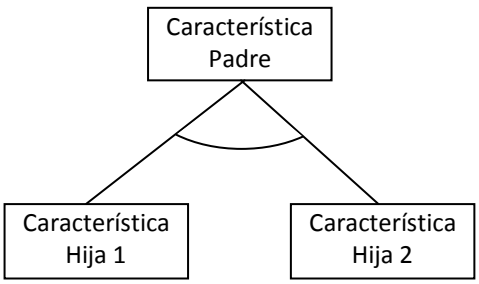

Figura 11 – Relación de opción multiple.

Las relaciones que hemos visto son relaciones verticales o jerárquicas, es decir son relaciones que van del padre al hijo. Además de estas relaciones en los diagramas de características podemos encontrar dos restricciones más. Estas restricciones, excluyentes y requerida, pueden afectar a cualquier nivel de la jerarquía.

Restricción excluyente

Con esta restricción, representada en la figura 12, indicamos que la activación de una de las características implica que no se puede activar las demás características de las que se excluye.

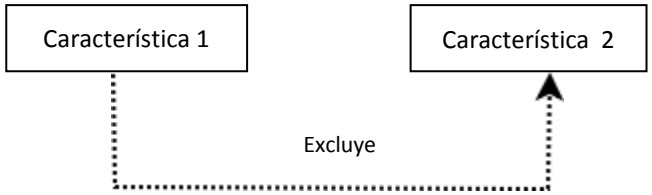

Figura 12 – Restricción excluyente

Restricción requerida

Con esta restricción, representada en la figura 13, se indica que la activación de la característica requiere la activación de las características con las que está conectada.

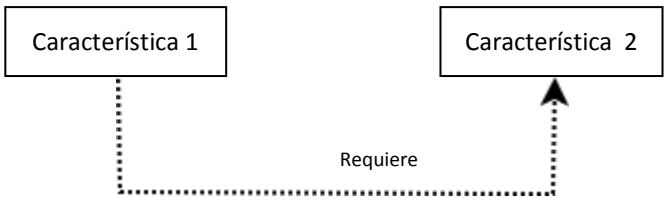

Figura 13 – Restricción requerida

En la siguiente imagen se muestra el modelado de características de una casa mostrando los conceptos que hemos explicado anteriormente tanto de relaciones como de restricciones. Además, como podemos ver en la leyenda del gráfico, en gris se muestran las características activas en ese momento mientras que en blanco se muestran las posibles variaciones que puede tener la configuración de la casa. Esta Smart Home tendría activo las luces automáticas y el sistema de seguridad.

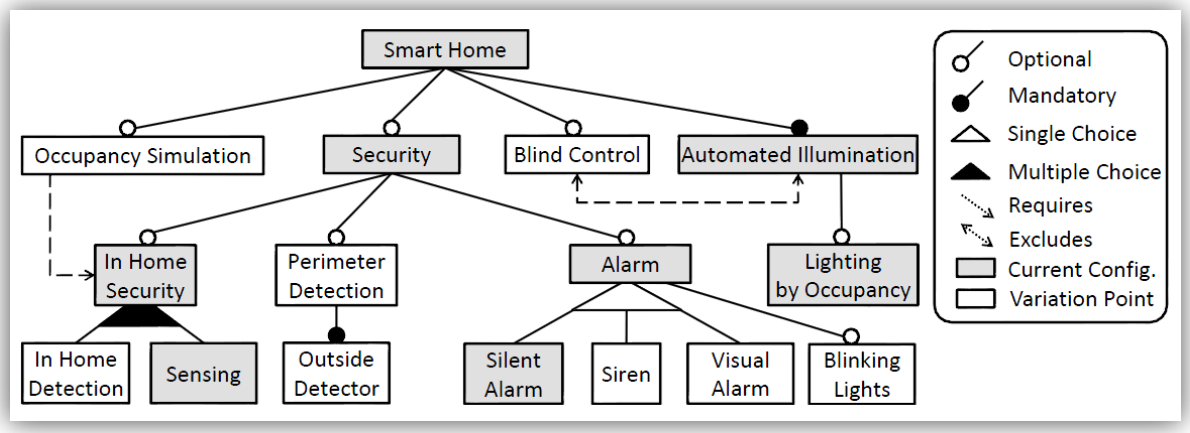

Figura 14 – Ejemplo Modelo de Características

Denotaremos [[FM]] al conjunto de todas las características (activas o inactivas) del modelo de características. Definiremos la Configuración Actual (Current Configuration, CC) de un sistema como todas sus características (Features, F) activas en el modelo de características.

$$CC \stackrel{\text{def}}{=} \{F\} \mid F \in [[FM]] \wedge F.state = Active \wedge CC \subseteq FM$$

De este modo para el ejemplo anterior definiríamos la CC del Modelo de Características con la siguiente expresión:

$$CC_{Figura\ 14} = \{SmartHome; Security; InHomeSecurity\ Sensing; Alarm; AutomatedIllumination; SilentAlarm; lightingByOccupancy\}$$

3.2.2. Dynamc Product Line Architecture

Para proveer una configuración flexible de reconfiguración, se considera una arquitectura que está basada en diferentes componentes con canales de comunicación entre ellos. Estos componentes se clasifican en dos categorías: Servicios y Dispositivos. Esta arquitectura permite realizar configuraciones de forma sencilla donde los canales de comunicación se pueden estableces dinámicamente entre componentes y estos componentes pueden desaparecer o aparecer de las configuraciones.

La figura 15 muestra esta arquitectura de reconfiguración de acuerdo a la sintaxis de PervML. Los servicios se representan por círculos y los dispositivos mediante cuadrados. Los canales son líneas que unen ambos componentes.

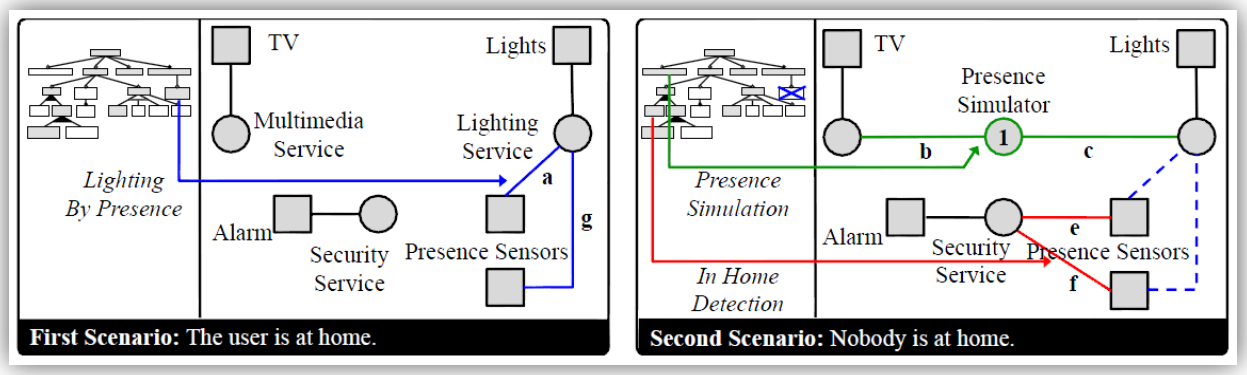

Figura 15 – Arquitectura de reconfiguración

Para definir que componentes y canales de la Smart Home se necesitan cuando se activa cierta característica se define el operador Superimposición (⊙) . Este operador coge una característica y devuelve los canales y componentes necesarios.

$$\odot (LightingByOccupancy) = \{a; g\}$$

$$\odot (OccupancySimulation) = \{1; b; c; d\}$$

$$\odot (InHomeDetection) = \{e; f\}$$

En la formula anterior, por ejemplo, para activar lightingByOccupancy se necesitan los canales a y g, para OccupancySimulation se necesitan los canales b, c y d y la componente 1 y finalmente para InHomeDetection se necesitan los canales e y f, como se puede observar en la Figura 12.

3.2.3. Resolución

Una resolución (R) se define como el conjunto de cambios necesarios para una reconfiguración. Una resolución está compuesta por una lista de pares formados por una característica (Feature, F) y el estado de esa característica (S). Cada reconfiguración está asociada a un contexto y representa los cambios producidos en el sistema cuando la condición se cumple.

$$Rx = \{(F, S)\} \epsilon\, F \in [FM] \wedge S \in \{Activo, Inactivo\}$$

Para las condiciones EmptyHome y Confort están asociados las siguientes resoluciones:

$$R_{EmptyHome} = \{(PresenceSimulation, Active), (InHomeDetection, Active), (LightingByPresence, Inactive)\}$$

$$R_{Comfort} = \{(PipedMusic, Active), (AutomatedIllumination, Active)\}$$

La resolución EmptyHome quiere decir que cuando la Smart-Home detecta que está vacia, debe reconfigurarse desactivando “Lighting by presence” y activando “Presence Simulation” y “In Home Detection”.

3.2.4. Incremento y Decremento de la arquitectura

El incremente y decrementeo de la arquitectura se calcula para determinar las modificaciones que hay que hacer para modificar la arquitectura. Concretamente, se necesitan dos operaciones: ArchitectureIncrement ($A\,\Delta$) y ArchitectureDecremente($A\,\nabla$). Estas operaciones tienen una resolución como entrada y calculan las modificaciones de la arquitectura en términos de componentes y canales. Estas operaciones las definimos de la siguiente forma:

$$A\,\Delta \stackrel{def}{=} \left((F, S) \in R \,\middle|\, S = \text{Active}\right) \setminus \odot\,(CC)$$

$$A\nabla \stackrel{def}{=} \left((F, S) \in R \,\middle|\, S = \text{Inactive}\right) \setminus \odot\,(F, S) \in R \mid S = \text{Active}$$

Estas operaciones indican como los componentes del sistema deben reorganizarse para la reconfiguración para moverse desde una configuración del sistema a otra configuración.

3.3. Desarrollo de la propuesta

Dada la complejidad y la variabilidad de los modelos de una Smart-Home debido a los numerosos cambios y usuarios que pueden interactuar con ella se hace necesaria una herramienta de almacenamiento de esas configuraciones que permita almacenar y mantener un histórico de esas configuraciones. Las posibilidades que se nos plantean ante esta herramienta son, como ya hemos comentado, de distinta índole. Ya sean la de la necesidad de restaurar la configuración debido a un fallo en el sistema o por deseo del usuario. Así como el mantener un histórico para poder consultar los cambios y las configuraciones pasadas.

El objetivo que se nos plantea por tanto es el de hacer una herramienta sencilla de cara al usuario, puesto que está dirigido a un público amplio y no solo con un perfil técnico alto, pero a la vez tiene que ser totalmente funcional, capaz de restaurar la configuración de la Smart-Home y volver a un estado anterior. Puesto que es una herramienta que va a estar de cara al usuario los gráficos que se muestren para representar la funcionalidad deben de ser sencillos y de fácil comprensión para el usuario. Preferiblemente que de un solo vistazo puedan interpretar cual es el estado de la casa o cuáles son los cambios que van a hacer. Para ello utilizaremos la arquitectura de línea de producto dinámica vista antes, en la que representábamos la configuración de la casa en servicios, dispositivos y canales de comunicación.

Internamente nuestra aplicación deberá ser capaz de realizar las operaciones que hemos visto anteriormente para poder reconfigurar la casa y para poder modificar las configuraciones que le mostremos al usuario. Para ello también utilizaremos los conceptos que hemos visto antes en cuanto a las reconfiguraciones, incrementos y decrementos.

Por último para demostrar la utilidad de la herramienta se va a diseñar un caso de estudio completo que permita reproducir los distintos usos de la Time Machine y ver su funcionamiento en un caso hipotético de funcionamiento. El objetivo de esto último es el de ver la utilidad en un caso real de utilización de la Time-Machine.

3.4. Tecnologías Utilizadas

En el siguiente apartado se va a hablar de las distintas tecnologías que se han utilizado para el desarrollo de la tesina. La tesina se ha desarrollado bajo tecnologías de código libre como Java y MySQL. Concretamente para algunos apartados del diseño de la interfaz se ha

utilizado el toolkit Prefuse, que está basado en Java y que permite generar gráficos para mostrar información de forma dinámica. Para el desarrollo se ha utilizado Eclipse.

3.4.1. Prefuse

Para desarrollar algunos componentes dinámicos de la aplicación, tales como los gráficos de configuración o la barra de histórico, se ha utilizado el toolkit Prefuse.

Prefuse es un toolkit basado en Java para la creación de aplicaciones interactivas de visualización de información. Soporta un amplio conjunto de características para modelado de datos, visualización e interacción. Proporciona estructuras de datos optimizadas para tablas, grafos y árboles, un host de diseño y técnicas de codificación visual y soporte de animación, consultas dinámicas y búsqueda integrada.

Prefuse utiliza la biblioteca de gráficos Java 2D y se integra fácilmente en aplicaciones Swing o applets. Prefuse está sujeto a los términos de la licencia BSD y puede utilizarse libremente para fines comerciales y no comerciales.

Prefuse pretende simplificar los procesos de visualización, control y asignación de datos, así como la interacción del usuario. Ha sido utilizado en proyectos de curso de desarrollo de software escolar, y de investigación académica e industrial y comercial.

Algunas de las características de Prefuse incluyen:

- Tablas, grafos y árboles de estructuras compatibles con atributos de datos arbitrarios, indexación de datos y consultas de selección, todo con una huella de memoria eficiente.
- Componentes de diseño, color, tamaño y codificaciones de forma, técnicas de distorsión.
- Una biblioteca de controles para operaciones comunes de interactivos, manipulación directa.
- Apoyo a la animación a través de un mecanismo de programación de actividades generales.
- Visión de transformaciones panorámica y zoom, incluyendo zoom geométrico y semántico.
- Búsqueda de texto integrado usando varios buscadores.
- Un motor de fuerza física de simulación para diseño dinámico y animación.
- Flexibilidad para múltiples vistas, incluyendo la "visión general + detalle" y mostrar "múltiples pequeños".
- Construido en SQL lenguaje de expresión para escribir consultas a Prefuse estructuras de datos y crear derivados de campos de datos.
- Soporte para emitir consultas a la base de datos SQL y asignación de resultados de la consulta en estructuras de datos Prefuse.

- API Simple y fácil de usar para crear componentes de procesamiento, interacción y procesamiento personalizado.

El diseño de las herramientas de Prefuse se basa en el modelo de referencia de visualización de información, una arquitectura de software de patrones que divide el proceso de visualización en una serie de pasos específicos, de adquisición de datos y modelado para la codificación visual de datos para la presentación de pantallas interactivas.

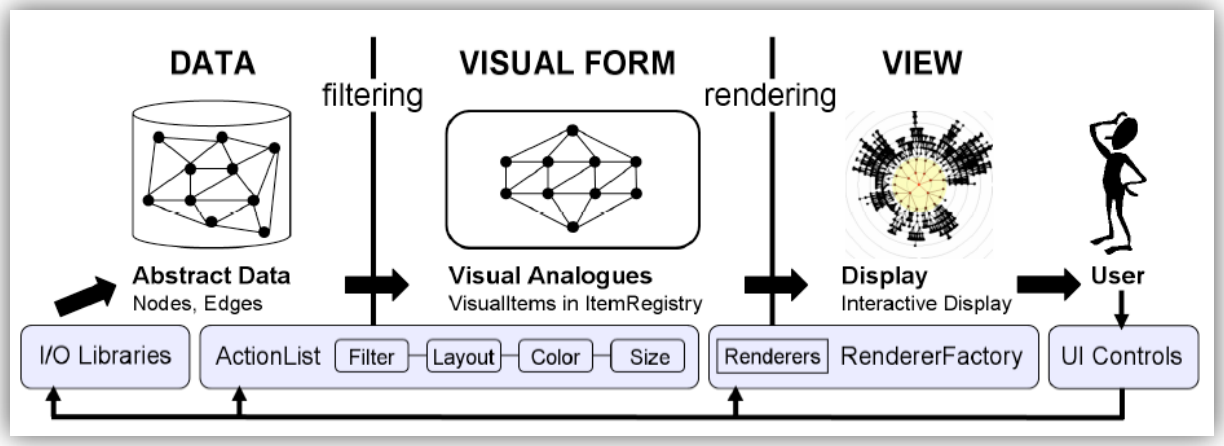

Figura 16 – Estructura de Prefuse

3.4.2. Java

Para el desarrollo de la aplicación se ha utilizado Java. Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. Tiene un gran recorrido en el mercado del desarrollo de software y cuenta con una gran aceptación.

Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre.

El lenguaje Java se creó con cinco objetivos principales:

- Debería usar la metodología de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Una de las características más importantes que nos ha hecho decantarnos por este lenguaje de programación es el de la independencia de la plataforma. Significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, “write once, run anywhere”.

El diseño de Java, su robustez, el respaldo de la industria y su fácil portabilidad han hecho de Java uno de los lenguajes con un mayor crecimiento y amplitud de uso en distintos ámbitos de la industria de la informática. Es posible desarrollar aplicaciones Java para dispositivos móviles, sistemas empotrados, navegador web, sistemas de servidor o aplicaciones de escritorio.

Recientemente se ha sacado la versión Java SE 7. Para el desarrollo de la tesina se ha utilizado la versión Java SE 6, que era la que estaba disponible en el momento del comienzo de la tesina. Entre otras en esta versión se mejoró la interfaz gráfica y el rendimiento.

3.4.3. Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código.

3.4.4. MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. 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. Fue comercializado por primera vez en 1981 por IBM, el cual fue presentado a ANSI y desde entonces ha sido considerado como un estándar para las bases de datos relacionales. MySQL funciona sobre múltiples plataformas, incluyendo GNU/Linux, Mac OS X, Solaris, SunOS, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista, Windows 7 y Windows Server (2000, 2003 y 2008), entre otros.

Para el desarrollo de la tesina se han utilizado diferentes programas como MySQL Administrator o MySQL Query Browser, que permiten de manera sencilla y mediante herramientas de modelado crear y administrar una base de datos relacional.

4. Time Machine

Figura 17 – Logo de la Time Machine

En este apartado se va a explicar detalladamente el proceso que se ha seguido para desarrollar la propuesta, así como las funcionalidades concretas y detalladas de la Time Machine. Se van a explicar el funcionamiento concreto de cada una de las pantallas y como se integran entre ellas. Primero vamos a comentar de forma detallada la funcionalidad básica general de la Time Machine basándonos en la perspectiva del usuario y explicando el diseño de la interfaz basándonos en las distintas funciones de la Time Machine.

Después pasaremos a explicar el proceso elegido para representar los esquemas de configuración de la Time Machine. Y Por último se va a detallar las características concretas de cada una de las pantallas de la Time Machine centrándonos en su funcionamiento y en las opciones de funcionamiento que tiene el usuario disponible para interactuar con ella.

4.1. Funcionalidad Básica

La interfaz se ha desarrollado siguiendo un patrón que sea visual y sencillo para el usuario. Intentando que la navegación sea intuitiva y el flujo de información sea coherente entre las distintas pantallas. Pese a que en cada pantalla hay una gran cantidad de información para el usuario, al hacerlo de una forma visual se ha conseguido que sea más atractivo y fácil de usar. El objetivo es que de un vistazo el usuario visualice la información y pueda manipularla según sus preferencias.

Para desarrollar la interfaz se ha utilizado Java ya que es una tecnología multiplataforma y ampliamente utilizada en la actualidad. Para algunos aspectos concretos de la interfaz, como los gráficos, se ha utilizado Prefuse, que permite crear visualizaciones dinámicas de información.

Prefuse es un toolkit basado en Java para la creación interactiva de aplicaciones de visualización de la información. Soporta un amplio conjunto de características para modelado de datos, visualización e interacción. Proporciona estructuras de datos optimizadas para tablas, grafos y árboles, un host de diseño y técnicas de codificación visual y soporte para animación, consultas dinámicas, búsqueda integrada y conectividad de bases de datos. Prefuse utiliza la biblioteca de gráficos Java 2D y se integra fácilmente en Aplicaciones Swing o applets como la que hemos desarrollado.

Con estas dos tecnologías unidas conseguimos, por una parte, una herramienta accesible para el amplio público no dependiente de la plataforma en la que se utilice y, por otra parte, una herramienta atractiva y de fácil uso para el usuario.

Figura 18 – Vista general de la Time Machine

En la figura 18 podemos observar una vista general de la Time Machine en su pantalla principal. En total la Time Machine va a contar de tres zonas bien diferenciadas que después pasaremos a explicar más detenidamente. Pero en líneas generales vamos a tener una Zona Principal donde el usuario podrá ver la configuración actual, modificarla y aplicarla a la Smart-Home. Una zona detallada, que se accederá mediante el botón lupa, en la cual podremos ver en mayor tamaño una habitación o zona de la casa y modificarla. Y una Zona Diferencias, a la

cual se accede desde el menú lateral de fechas, donde el usuario podrá ver las diferencias entre la configuración actual y la configuración de la fecha seleccionada.

En cada pantalla de la Time Machine tenemos elementos comunes que se repiten durante todas las pantallas. Estas funciones comunes ayudan al usuario a no desorientarse y hacerle más fácil la navegación entre las distintas pantallas. Dentro de cada pantalla estos elementos comunes pueden tener funciones ligeramente diferentes pero lo normal es que sean muy similares o iguales. Estas funciones comunes a todas las pantallas son:

- **Histórico:** En la parte lateral derecha de la Time Machine tendremos siempre visible un histórico con las fechas y horas en las que hay guardadas una configuración de la Smart-Home. En la parte superior se muestra la fecha y hora actual y en cada una de las fechas al lado aparecerá un icono indicando si se trata de una configuración guardada automáticamente por el sistema o una guardada manualmente por el usuario. La navegación sobre ellas se realiza de manera sencilla aumentando el tamaño cuando te posicionas encima de ellas como se puede ver en la figura 19. Pulsando sobre una de las fechas, mostrada en color rojo cuando te posicionas sobre ella, podrás acceder a la configuración de esa fecha.

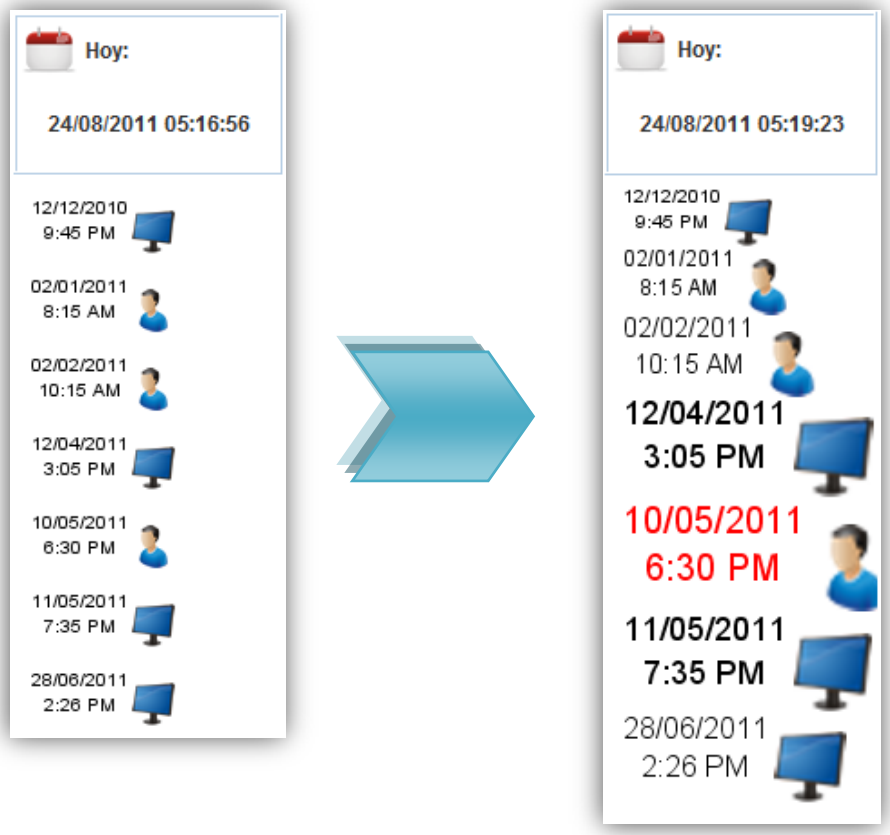

Figura 19 – Histórico

• **Opciones de funcionalidad:** En el menú lateral derecho encontraremos las distintas opciones de configuración que se le pueden aplicar a nuestra casa. Pulsando sobre una de ellas se le aplicará esa configuración a toda la casa o a la habitación en concreto dependiendo de la parte de la aplicación en la que nos encontremos. Estas opciones no se encuentran disponibles dentro de la zona comparativa ya que se trata de una zona de consulta únicamente y no para modificación de las configuraciones antiguas. Estas opciones de configuración no son otra cosa si no que las distintas características o features del modelo de características de nuestro sistema. De esta forma cuando pulsemos sobre uno de ellos lo que haremos será activar esa característica.

Figura 20 – Opciones de Configuración

• **Opciones de Edición:** En la parte inferior podremos encontrar distintos botones de edición para volver a la pantalla principal (botón Home), o navegar entre las distintas configuraciones así como para aceptar o rechazar los cambios. Estás opciones pueden variar en cuanto a su funcionalidad dependiendo de la zona en la que nos encontremos, aunque lo normal es que la funcionalidad sea muy similar. En la figura 20 podemos ver cómo sería la barra de opciones de Edición.

Figura 21 – Opciones de Edición

Dentro de la funcionalidad y la navegación de la Time Machine encontramos 3 zonas o pantallas bien diferenciadas donde se desarrollan los distintos procesos de funcionamiento de la misma. Estas zonas son las siguientes:

- **Zona Principal:** Es la pantalla principal de la Time Machine que se presenta al usuario nada más acceder a la Time Machine. En ella encontramos la configuración actual de cada habitación representada en un gráfico. En ella podemos consultar mediante el menú de fechas las distintas configuraciones almacenadas. Mediante la barra lateral de configuraciones, además, se le podrán aplicar configuraciones comunes para todas las habitaciones.

- **Zona Detallada:** Pulsando sobre el botón "Detalle", representado mediante una lupa en la Zona Principal de la configuración de una habitación, accederemos a la pantalla en la que se nos muestra la configuración de la habitación de forma principal ampliada. En esta pantalla al igual que en la anterior podemos navegar por configuraciones antiguas (mediante la barra menú de fechas), aplicar nuevas configuraciones (únicamente para esta habitación) o navegar entre las distintas habitaciones, mediante las flechas colocadas a cada lado del gráfico de configuración.

- **Zona Comparativa**: Pulsando sobre una fecha del menú de fechas lateral accederemos a la Zona Comparativa. En ella podremos ver la configuración antigua viendo cómo estaba configurada en ese instante en el tiempo mostrando las diferencias respecto a la actual (marcadas en fondo rojo o verde) para que de una forma intuitiva sepamos lo que ha cambiado. Podremos aceptar los cambios y aplicar la configuración a nuestra Smart-Home. En está pantalla no mostraremos las opciones de configuración.

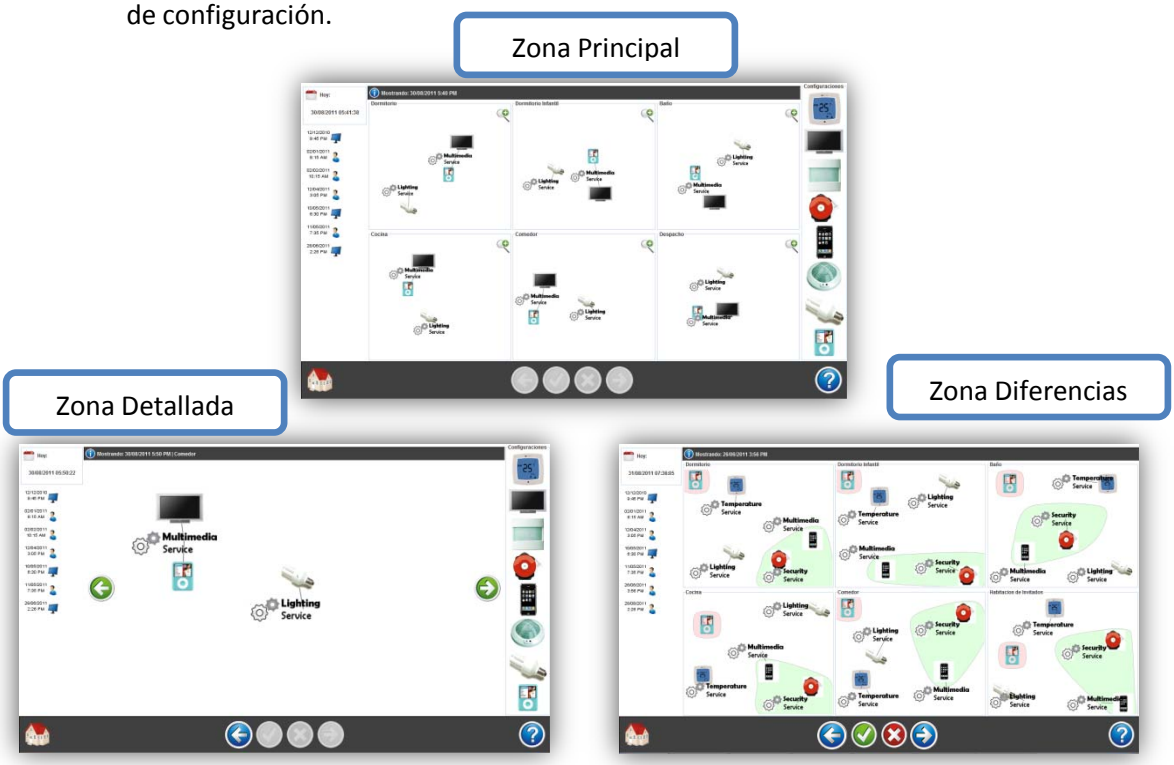

Figura 22 – Pantallas de la Time Machine

Como podemos ver en la figura 22 el aspecto visual entre las distintas pantallas es bastante similar entre ellas.

En el gráfico de la figura 23 podemos observar la navegabilidad de la aplicación por sus diferentes zonas. Como podemos ver en cualquier momento podemos acceder a la Zona Principal y a la Zona Comparativa. En cambio, solo podemos acceder a la Zona Detallada desde la Zona Principal ya que en la Zona Detallada solo veremos el detalle de la configuración actual, mostrada en la Zona Principal.

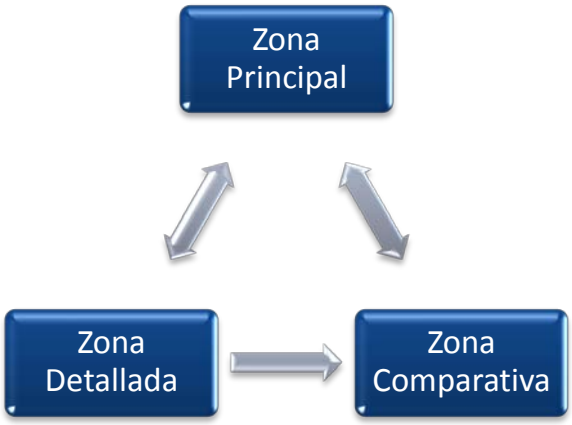

Figura 23 – Navegabilidad entre las pantallas

A continuación se va a describir de forma más detallada la funcionalidad haciendo hincapié primero en los gráficos de configuración para después pasar a describir detalladamente la funcionalidad de cada una de las pantallas de la aplicación. De esta forma podremos hacernos una idea global del funcionamiento de la aplicación.

4.1.1. Gráficos de configuración

Figura 24 – Relación entre configuración y otros componentes

Para representar la configuración de la casa, tanto la actual como las pasadas, se ha utilizado, como ya hemos comentado antes, el toolkit Prefuse que está basado en el lenguaje de programación Java. Este toolkit nos ha permitido crear unos gráficos sencillos y atractivos para el usuario y a la vez funcionales para representar la configuración de la casa, que es nuestro objetivo. Los gráficos tienen un aspecto dinámico permitiendo que el usuario pueda moverlos para visualizarlos de la forma deseada. Esto nos permite una interacción mayor con el

usuario y una mayor configurabilidad de la visualización a las preferencias del usuario.

Estos gráficos tienen una base en la arquitectura DSPL. Como podemos ver en la figura 26 los gráficos de nuestra Time Machine están compuestos por nodos y por líneas que los unen, al igual que los gráficos de la arquitectura DSPL que están formados por componentes y por canales de comunicación. Al igual que en la arquitectura DSPL en nuestros gráficos los componentes están divididos en Servicios y Dispositivos. Nosotros representamos los dispositivos con una imagen representativa de este y los servicios con su nombre.

El grafico de configuración de una habitación nos indicaría por tanto los dispositivos y los servicios que están activos en ese instante de tiempo, en esa habitación y los canales de comunicación que existen entre ellos. Por ejemplo en el primer gráfico de la Figura 26 podemos ver cómo están activos los servicios Multimedia y de Iluminación. Asociado al servicio multimedia tenemos el dispositivo de video y asociado al servicio de Iluminación tenemos las luces automáticas y la alarma por luz.

Nuestra Time Machine muestra la información de configuración de la casa en base a los datos, por una parte la fecha en la que se creó esa configuración y por otra parte la habitación a la que pertenece esa configuración. Otro de los datos a los que está asociada una configuración es al usuario, pero este dato va a ser menos visual para el usuario ya que en la aplicación al ver el histórico solo va a poder ver si la configuración fue creada por un usuario de forma manual o, por el contrario, fue creada de forma automática por la máquina. Internamente cada configuración sí que va a estar relacionada con un usuario y para la simulación se tendrá en cuenta. Pero la gestión completa de los usuarios queda fuera del ámbito de esta tesina.

Para representar los graficos se ha utilizado el componente llamado AggregateDemo del toolkit Prefuse implementando la interfaz RadialGraph. Los métodos de esta interfaz nos permite introducir nuevos nodos, quitarlos, añadir nuevos canales o quitarlos con los métodos siguientes:

```
3 public interface RadialGraph {
4       public void addNode(String nodeName, String nodeImage);
5       public void removeNode(String nodeName);
6       public void addEdge(String sourceNode, String targetNode);
7       public void removeEdge(String sourceNode, String targetNode);
8 }
```

Figura 25 – Interfaz RadialGraph

Además, para la zona comparativa los gráficos tienen una componente grafica adicional. Para hacer más sencilla la comparación con la configuración actual y para que no haya necesidad de tener los gráficos actuales y los que se está consultando al mismo tiempo en pantalla, los gráficos de la zona comparativa se han diseñado de forma que se presenten en fondo verde los dispositivos y servicios que se han añadido y en rojo los que se han eliminado.

De esta forma, el gráfico que vemos en la zona comparativa no es exactamente un gráfico de la configuración de la casa en ese momento de tiempo, si no que se trata de el gráfico de configuración de la casa en ese momento añadiendoles los dispositivos y servicios que están actualmente y no estaban en ese momento. Estos dispositivo y servicios estarán marcados con un fondo rojo, ya que representa los dispositivos que si aplicamos la reconfiguración se van a eliminar. Además en fondo verde como ya hemos dicho tendremos los dispositivos que estaban en la configuración en ese momento de tiempo y no están en la configuración actual. Están representados en verde ya que simbolizan los dispositivos y servicios que añadiremos si aplicamos esa configuración.

Por ejemplo en la figura 26 podemos ver como el grafico actual tiene activados el servicio multimedia y el servicio de iluminación. Cuando consultamos el gráfico de esa habitación en otro instante de tiempo vemos que si aplicasemos esa configuración añadiriamos un dispositivo al servicio multimedia, marcado en fondo verde y quitariamos uno de los dispositivos del servicio de iluminación , remarcado con un fondo en color rojo.

Esto viene a coincidir con lo que hemos explicado antes sobre los incrementos y decrementos. De esta forma se ofrece al usuario una visualización intuitiva de los incrementos en este caso representados con fondo verde y de los decrementos representados con fondo rojo.

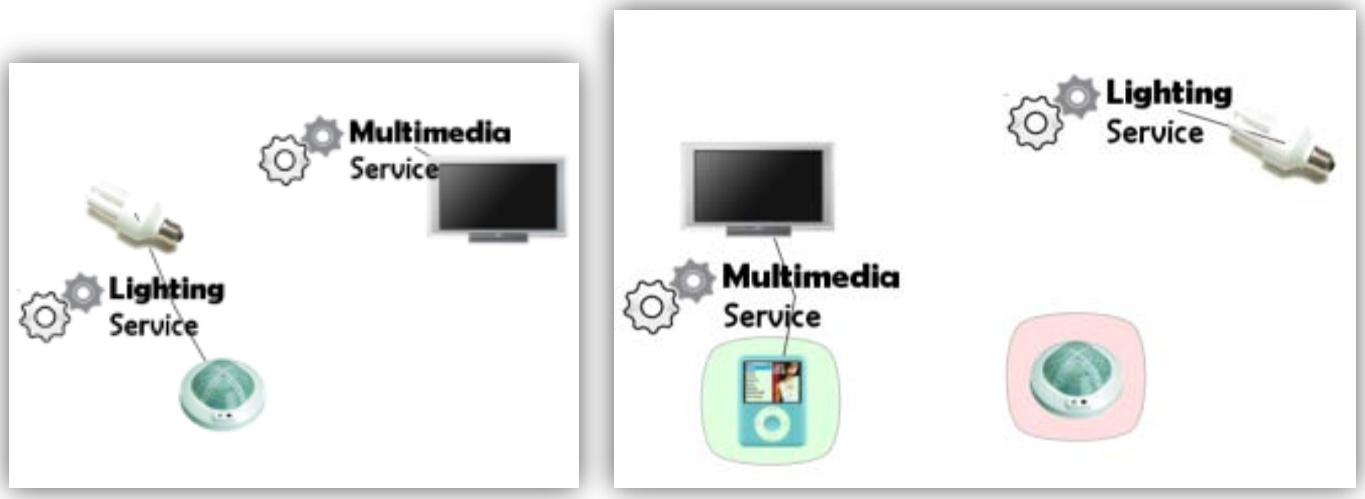

Figura 26 – Gráficos de configuración

4.1.2. Zona principal

Nada más entrar en la Time Machine visualizaremos la “Zona Principal”. Su objetivo principal es la de presentarnos la configuración actual de la casa de un solo vistazo. Para ello, mediante una división en rejilla, encontraremos en cada una de las celdas la configuración de las distintas habitaciones presentándonos el nombre de la habitación en la parte superior del cubículo.

De esta forma en cada celda de la cuadrícula tendremos la configuración actual de una habitación o zona de la casa representada en un sencillo gráfico donde se pueden observar los dispositivos y servicios activos para esa zona. Dentro de cada celda tendremos un botón lupa donde podremos acceder a la Zona Detallada para esa habitación y modificar las propiedades de la configuración de esa habitación. Encima de la cuadricula podemos ver una barra de información donde nos indica la fecha de la configuración que está mostrando.

A la derecha podremos observar los distintos botones con las distintas opciones de configuración que le podemos aplicar a la Smart-Home. Si pulsamos sobre cualquiera de ellos aplicaremos esa configuración a todas las habitaciones de la casa. Si alguna de ellas ya tuviera esa opción no se aplicaría al igual que si para una determinada habitación no estuviera disponible ese dispositivo tampoco se aplicaría en la configuración.

Figura 27 – Vista general de la zona principal

Al añadir un dispositivo nuevo a la configuración actual, veremos que los botones de navegación, en un principio deshabilitados, se activan. Tendremos entonces disponibles las opciones de aceptar los cambios y reconfigurar la Smart-Home con el botón verde de la barra de navegación o por el contrario podremos deshacer los cambios con el botón rojo con una cruz blanca. Mediante los botones azules con flechas blancas podremos hacer y deshacer los cambios que hayamos efectuado en los gráficos.

Hay que decir que todas estas operaciones que se hacen sobre la configuración son una simulación hasta que no se pulsa el botón de aceptar cambios. Es decir, todos los cambios que hagamos no se van a aplicar a la Smart-Home hasta que no los aceptemos. La aplicación lo único que hace es mostrarnos como quedaría la configuración de nuestra casa si efectuáramos los cambios.

En resumen el usuario podrá hacer uso desde esta pantalla de las siguientes funciones y características:

- Ver una vista general de la configuración actual de la casa.

- Modificar la configuración actual.
- Una vez modificada, ejecutar las distintas opciones de edición (aplicar la configuración, rehacer cambios, deshacer cambios o desechar cambios).
- Navegar a la Zona de Diferencias para ver una configuración del histórico mediante la barra de fechas.
- Navegar a la Zona Detallada mediante el botón lupa.

A continuación vamos a ver una secuencia de pasos para ver cuáles serían los efectos y el funcionamiento de los distintos botones. Tras la explicación de cada paso veremos una imagen mostrando el estado de la Time Machine.

Paso 1- Nada más entrar los botones de edición están deshabilitados puesto que aun no se ha realizado ninguna modificación en la configuración y por tanto no hay cambios que aplicar o deshacer.

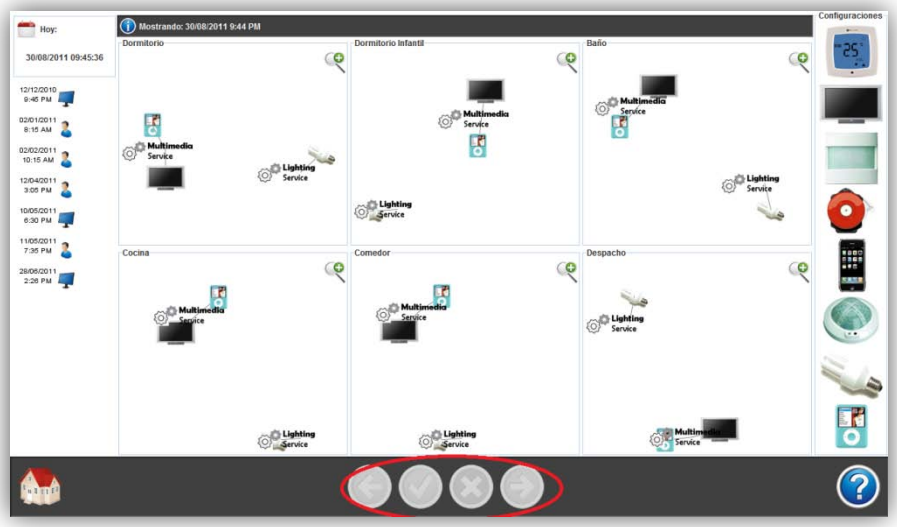

Figura 28 – Vista de entrada Zona Principal

Paso 2- Aplicamos dos configuraciones a la Configuración actual de la casa. Mediante los botones de configuración el usuario elige los dispositivos que desea aplicar. Los botones Deshacer, Aceptar Cambios y Descartar Cambios se activan y los dispositivos se añaden a las configuraciones como podemos ver en la secuencia de la figura 29.

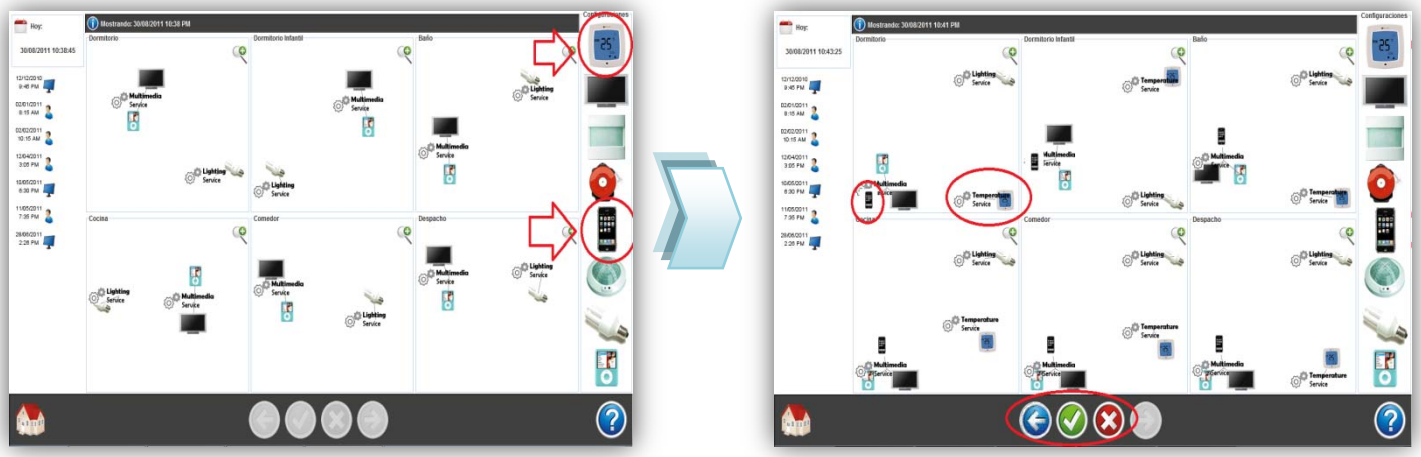

Figura 29- Añadir dispositivos

Paso 3 – Deshacemos una configuración. En caso de que el usuario se haya equivocado en el último paso se le permite deshacer los cambios. El botón Rehacer se activa y si lo pulsamos volvería a aparecer la configuración. En la figura 30 podemos ver que si pulsamos sobre el botón deshacer se deshace la configuración (desaparece el dispositivo) y se activa el botón rehacer.

Figura 30 –Deshacer configuración.

Paso 4.1 – Descartamos cambios. Finalmente el usuario puede no querer aplicar los cambios y por tanto desea eliminarlos. Mediante el botón rojo “Descartar Cambios” las configuraciones se borran. Los botones de edición además se desactivan.

Figura 31 – Descartar cambios

Paso 4.2- Aceptamos cambios. Si el usuario finalmente desea aplicar los cambios pulsando el botón se reconfigurará la casa. Se aplican los cambios reconfigurando la casa y se crea una nueva entrada en el histórico con la configuración antigua. Los botones de edición se desactivan.

Figura 32 – Aceptar cambios

4.1.3. Zona Detallada

A esta zona accederemos mediante el botón  presente en cada uno de los cubículos correspondientes a cada habitación. El objetivo es mostrar en mayor tamaño y por tanto con una mejor visualización una habitación en concreto, además de permitir acciones sobre una única habitación. Una vez se ha accedido a la Zona Detallada encontraremos ocupando toda la zona central el esquema de configuración de la habitación correspondiente.

Al igual que en la zona principal a la derecha tendremos los botones de configuración. Pulsando sobre ellos aplicaremos esa configuración únicamente sobre la habitación que estamos visualizando actualmente.

A izquierda y derecha del esquema encontraremos unas flechas de dirección. Con ellas podremos navegar entre las distintas habitaciones de la casa para poder visualizar sus configuraciones. Durante los movimientos entre las distintas habitaciones no perderemos los cambios que hayamos realizado sobre las distintas configuraciones.

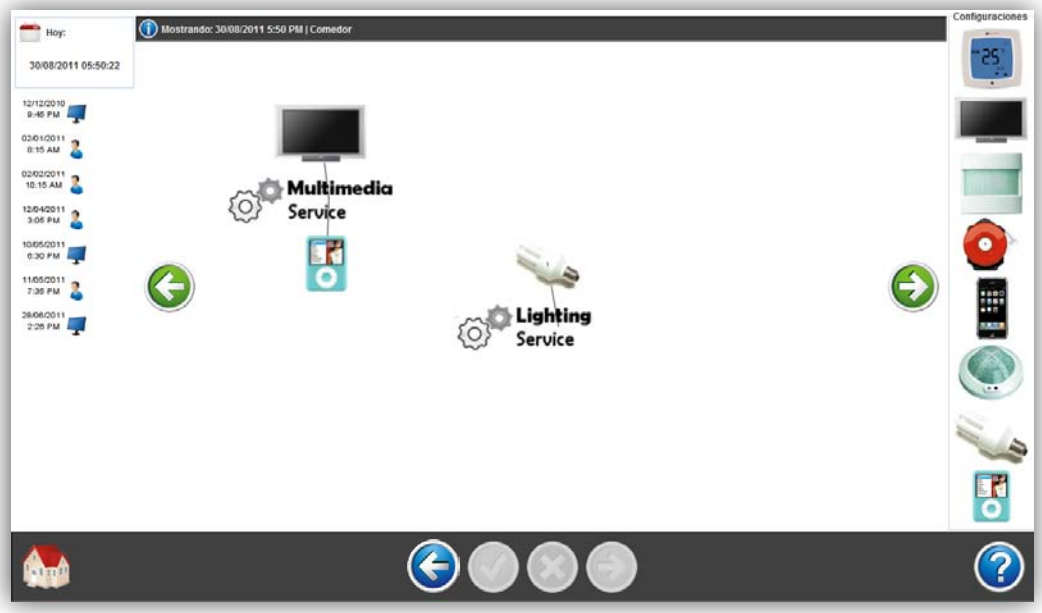

Figura 33 – Vista general de la Zona Detallada

Al igual que en la zona principal en la parte inferior tendremos los botones de edición mediante los cuales podremos hacer o deshacer los cambios, aceptar o rechazarlos.

En resumen las características y opciones que tendrá disponible el usuario en esta pantalla serán:

- Visualización de la configuración de una habitación a tamaño completo.
- Modificación de la configuración de una habitación mediante los botones de configuración.
- Una vez modificada, ejecutar las distintas opciones de edición (aplicar la configuración, rehacer cambios, deshacer cambios o desechar cambios).
- Navegabilidad entre las distintas habitaciones mediante las flechas laterales al gráfico de configuración.
- Navegar al histórico mediante la barra de fechas.
- Navegar a la ventana principal mediante el botón Home.

A continuación se va a ver una secuencia del funcionamiento de los distintos botones para mostrar cuál sería el funcionamiento de esta pantalla.

Paso 1 – Desde la ventana principal pulsamos sobre el botón lupa, bien para visualizar de forma más detallada una habitación o bien porque se quiera modificar únicamente una habitación. Accederemos a la ventana detalle de la habitación correspondiente.

Figura 34 – Acceso Zona Detalle

Paso 2- Buscamos la habitación deseada con las flechas de dirección, en caso de que queramos acceder a una habitación distinta de la que habíamos seleccionado en un principio. El gráfico cambia y arriba nos indica en que habitación estamos.

Figura 35 – Cambio de Habitación

Paso 3- Aplicamos una configuración a esa habitación mediante los botones laterales de configuración. El usuario puede querer modificar solo una habitación. Situándose en la habitación indicada los cambios que se apliquen solo afectarán a dicha habitación. Vemos que solo se ha aplicado en esa habita**ci**ón y no en las demás. Al mismo tiempo los botones de edición se activan de la forma correspondiente.

Figura 36 – Aplicar configuración a una habitación

Paso 4- Los botones de edición funcionarían igual que en la pantalla principal. Aceptamos cambios y vemos que se crea una nueva configuración en el histórico con la configuración antigua. Los botones de edición se desactivarán.

Figura 37 – Aceptar cambios

Paso 5- Pulsamos sobre el botón Home y volvemos a la pantalla principal. Vemos que los cambios se han aplicado correctamente únicamente a la habitación que queríamos como podemos ver en la figura 38.

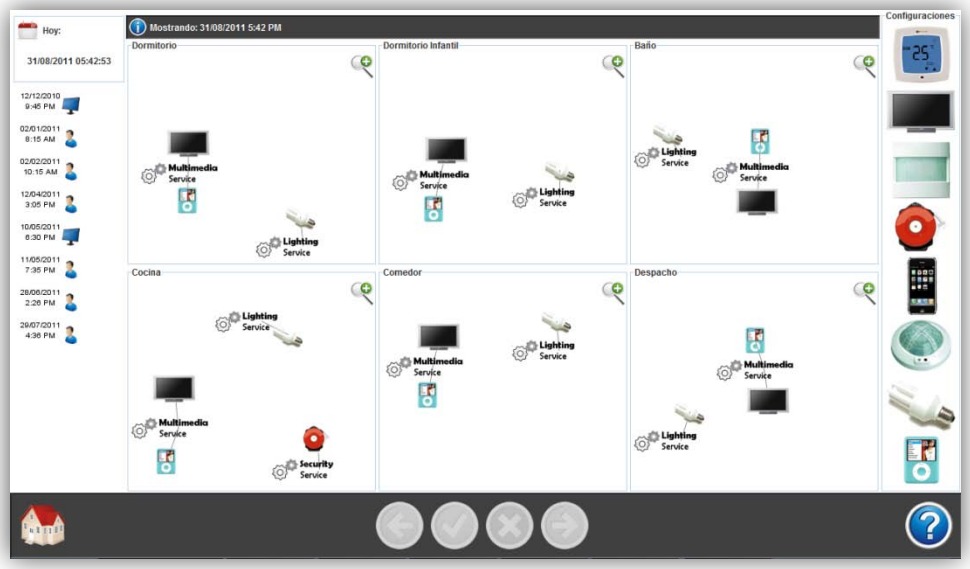

Figura 38 – Ventana principal con cambios en una habitación

4.1.4. Zona Comparativa

Pulsando sobre una fecha del histórico accederemos a la zona comparativa. Esta zona será muy parecida a la zona principal salvo algunas diferencias. Básicamente no tendremos disponibles las opciones de funcionalidad y el botón para acceder a la zona detallada. Además en los gráficos de las configuraciones tendremos señalado en color verde y rojo las diferencias añadidas y eliminadas respectivamente, como ya hemos explicado anteriormente cuando hemos visto el funcionamiento de los gráficos.

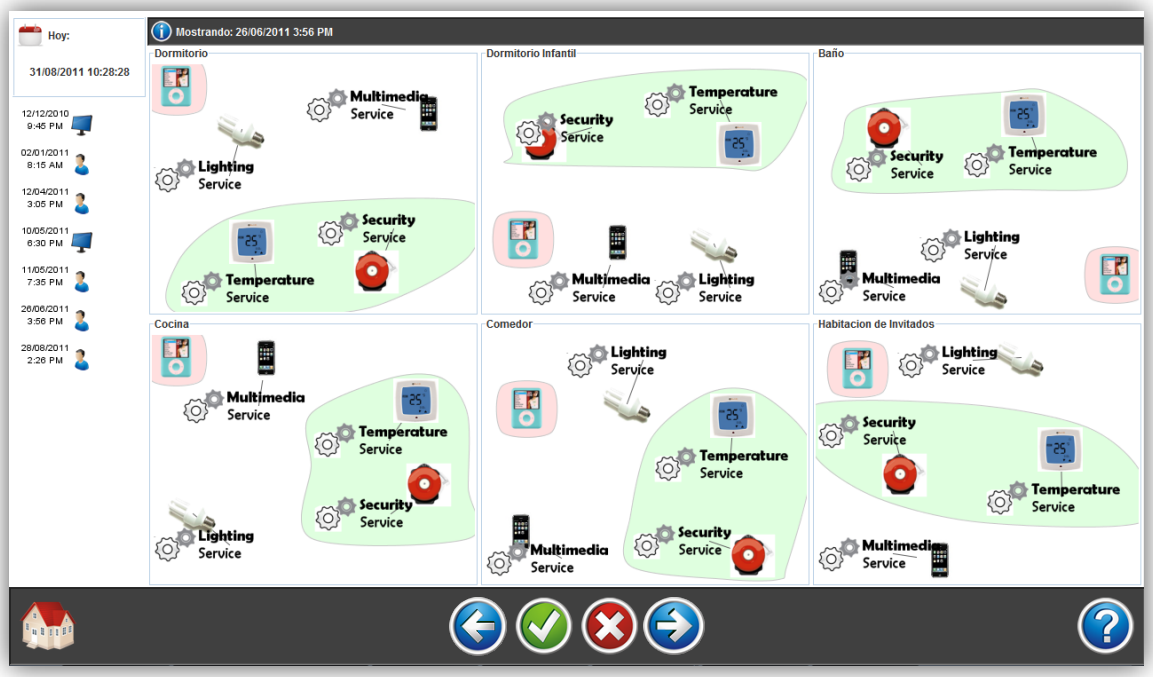

Figura 39 – Vista general zona Comparativa

En esta pantalla no tendremos disponibles las opciones de configuración, así como la opción de ver en detalle las distintas configuraciones, ya que no se nos permitirá modificar la una configuración pasada. Solo podremos aplicarla y reconfigurar la casa. Una vez aplicada la configuración a nuestra casa esta pasará a ser nuestra configuración actual y entonces podremos modificarla.

Por otra parte las opciones de edición son ligeramente diferentes. La opción de aceptar estará siempre activa y servirá para aceptar la configuración y aplicarla a la casa. Por su parte el botón de borrar no borrará los cambios realizados sobre la configuración como en las pantallas anteriores si no que servirá para eliminar del sistema de histórico esa configuración. Por último los anteriores botones de deshacer y rehacer ahora se convierten en botones de navegabilidad para volver atrás y adelante en las configuraciones del histórico.

Por tanto las opciones que el usuario tendrá disponibles serán las siguientes:

- Visualización de la configuración de la casa en la fecha seleccionada, mostrando las diferencias mediante un fondo de distinto color.
- Navegación entre las distintas fechas mediante las flechas de navegación de la barra de edición.
- Aplicar los cambios y reconfigurar la casa.
- Borrar una configuración del histórico.
- Acceder a otra configuración mediante la barra de fechas.
- Volver a la pantalla principal de la Time Machine mediante el botón Home.

A continuación se va a mostrar una serie de pasos para mostrar cual sería el funcionamiento y el efecto de los distintos botones:

Paso 1 – Pulsando sobre la entrada del histórico accedemos a la configuración de esa fecha. El usuario en este caso desearía consultar las configuraciones antiguas guardadas en la Time Machine

Figura 40 – Entrada en Zona Comparativa

Paso 2- Mediante los botones de navegación nos movemos entre las distintas fechas. El usuario de esta forma puede ir viendo la evolución en el tiempo de la configuración de su casa. En la barra superior nos indica que configuración estamos mostrando. Como podemos ver en la figura 41, donde la fecha en la barra superior de información va cambiando conforme van cambiando las distintas configuraciones del histórico.

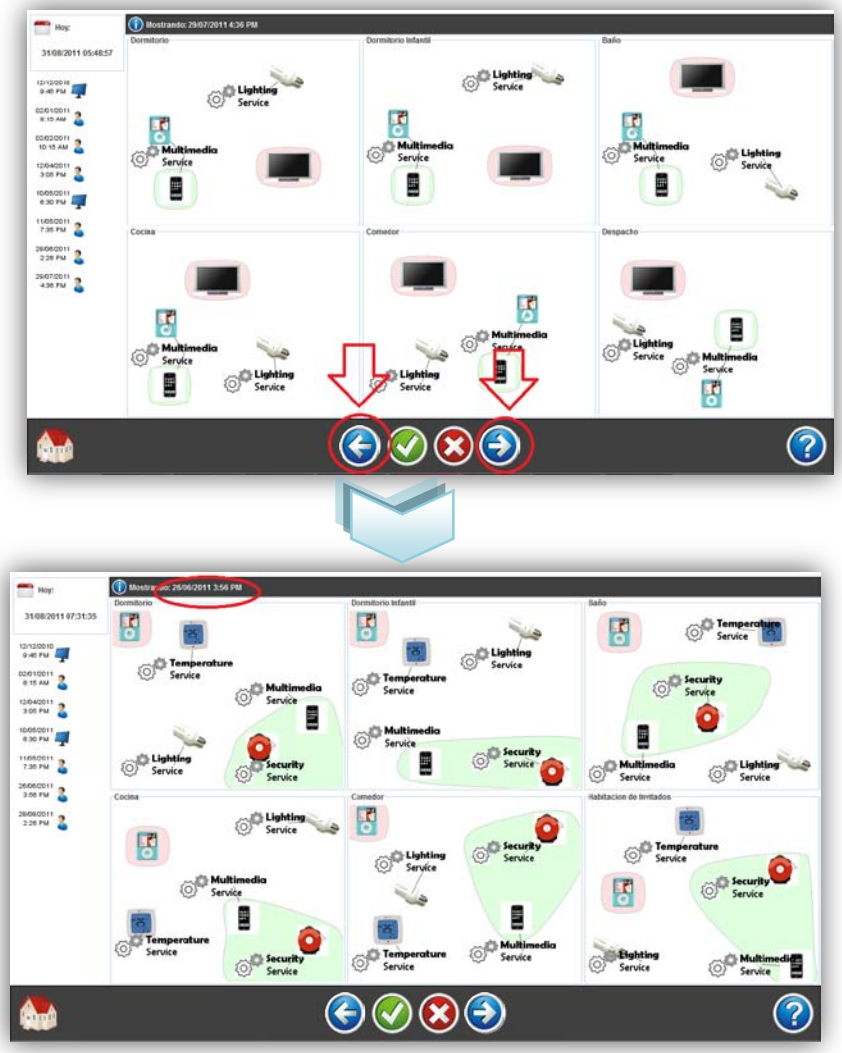

Figura 41 – Navegación en Ventana comparativa

Paso 3 – Pulsando el botón borrar, borramos esa configuración. El usuario puede querer quitar una configuración que no sea relevante o que no desea guardar. Vemos como desaparece del histórico y nos mostrará la configuración siguiente. En la segunda parte de la figura 42 podemos observar como en el histórico hay una fecha menos correspondiente a la configuración que hemos borrado. Además la configuración que vemos ha cambiado y se trata de la siguiente en la lista.

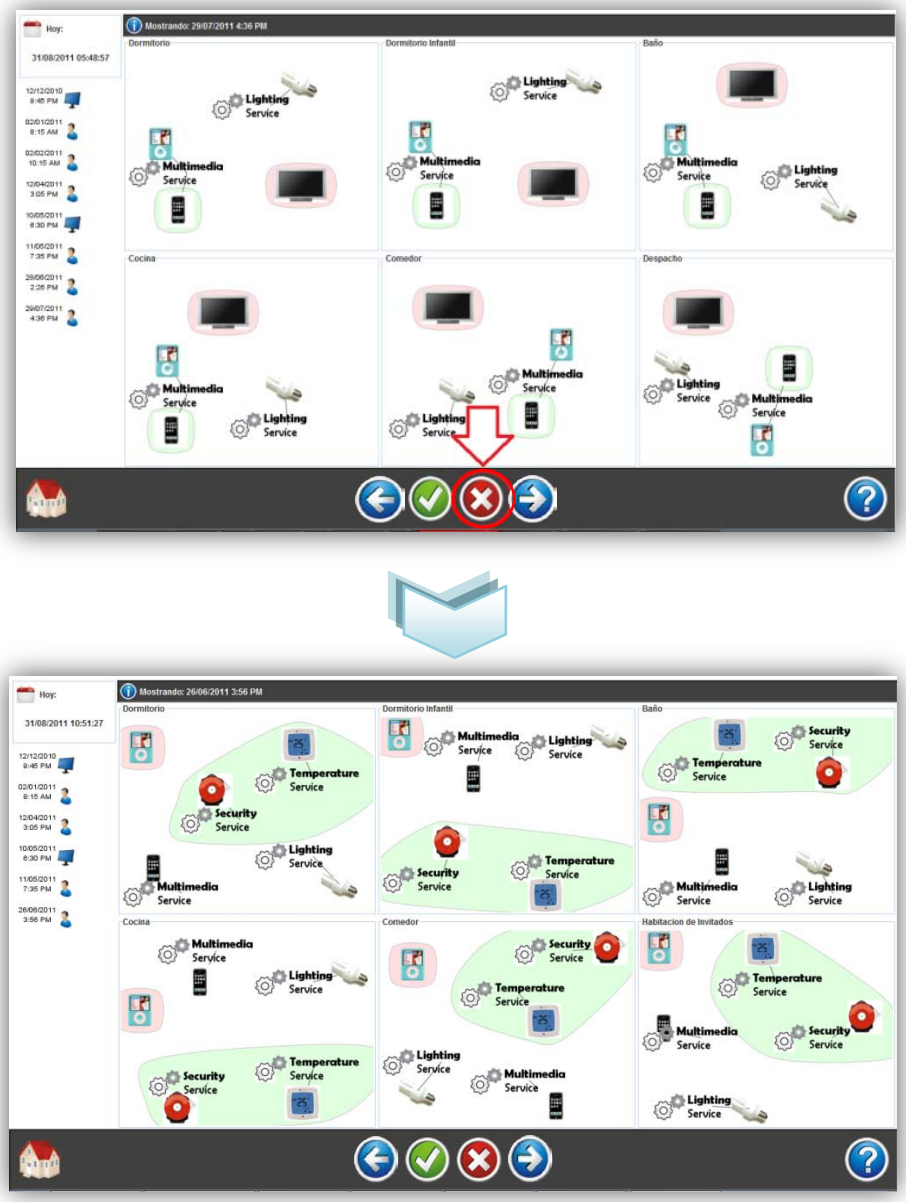

Figura 42 – Borrar configuración

Paso 4 – Pulsando sobre el botón aceptar la casa se reconfigura. Finalmente el usuario aplica la configuración antigua a la casa reconfigurándola. Veremos, al igual que en las pantallas anteriores, como se crea una nueva entrada en el histórico con la configuración que hasta ahora era la actual y se aplica la configuración elegida. La aplicación nos mostrará directamente la pantalla principal con la nueva configuración que acabamos de aplicar a la Smart-Home.

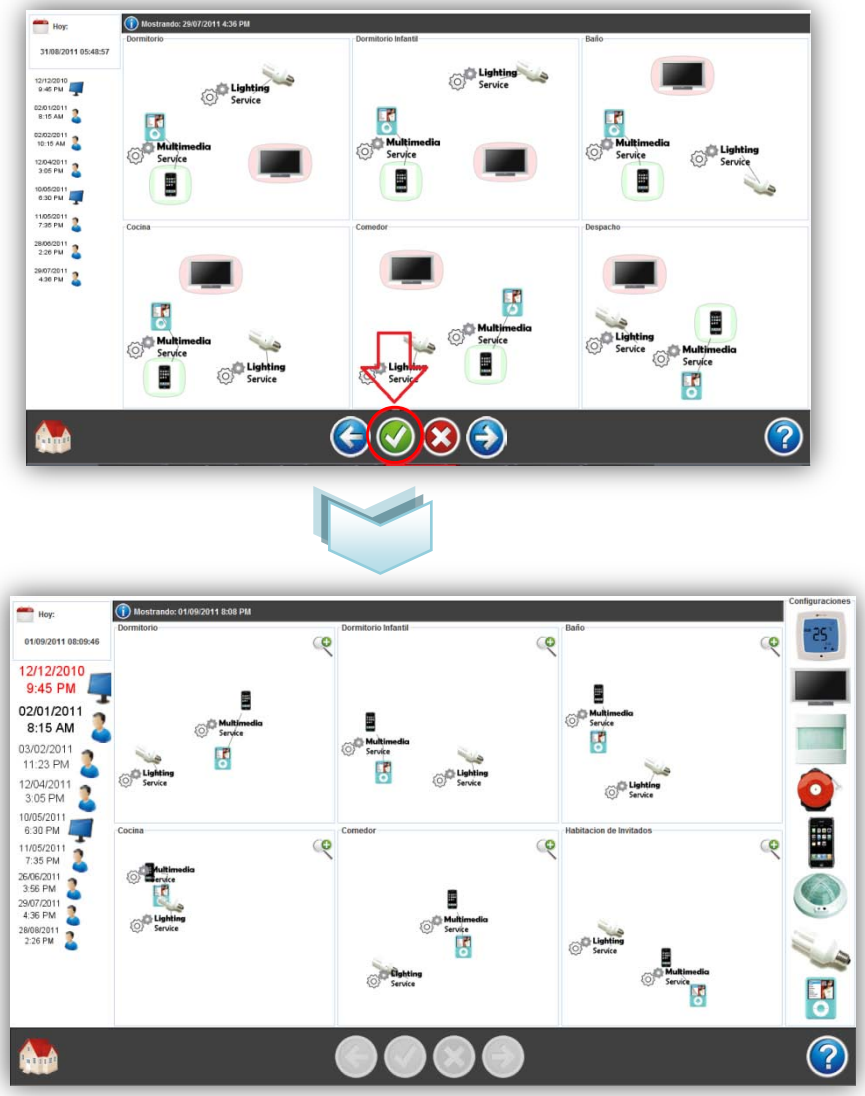

Figura 43- Reconfiguración con configuración antigua

5. Diseño e Implementación de la Time Machine

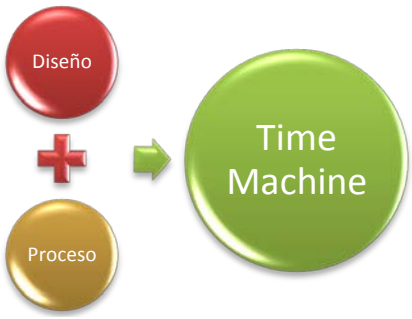

Figura 44 – Alcance del capítulo 5

En este apartado se va a describir los aspectos relacionados con el diseño de la aplicación. Primeramente se va a proceder a explicar el proceso que se ha seguido para desarrollar la aplicación contando las fases que se han seguido para su desarrollo. Después se pasará a explicar las claves del diseño interno que se ha desarrollado para la Time Machine contando cómo se adaptaría la Time Machine para otros casos.

5.1. Proceso

El proceso de desarrollo de la Time Machine ha sido puramente incremental. En un principio se marcaron unos objetivos básicos que era los de mostrar la información de las configuraciones del pasado y dar la opción al usuario de aplicarlas a la Smart-Home. En base a eso se creó un diseño simple en el que se mostrase está información.

Primeramente se empezó creando, mediante una demo, el diseño de la interfaz para ir especificando poco a poco las distintas características y funcionalidades de la Time Machine.

Dentro del diseño de la interfaz, uno de los primeros pasos fue elegir Prefuse para diseñar las componentes dinámicas como la barra de fechas o los gráficos de configuración,

que son los dos elementos más importantes de la aplicación a nivel de funcionalidad y de objetivos básicos. Por ello se eligieron dos componentes de Prefuse, el fisheye para el menú de fechas y el AggregateDemo para los gráficos de configuración, y se adaptaron a nuestras necesidades. Se decidió además que la pantalla principal debería mostrar un gráfico por cada habitación y que la mejor opción era utilizar una rejilla para, en cada celda, colocar cada uno de los gráficos. Además se decidieron colocar los botones de edición en la parte inferior de forma que el usuario pudiera aplicar los cambios.

A continuación se pensó que sería una buena idea que el usuario pudiera modificar la configuración actual. Por ello se añadió una barra lateral mediante la cual el usuario pudiera aplicar distintas configuraciones a toda la casa. A partir de ahí se planteó que, tal vez fuese necesario ver los gráficos con mayor detalle. Por ello se introdujeron los botones lupa para poder ampliar los gráficos de configuración. Para ver los gráficos en detalle se decidió crear otra ventana donde los gráficos se mostrasen a tamaño completo. Una de las necesidades que se planteó en este punto en el que navegábamos entre distintas pantallas es que el menú de fechas siempre estuviera visible. Además los botones de edición se mantuvieron, al igual que la barra de configuración. Con la diferencia en que esta última solo modifica la habitación actual. Con esto conseguíamos otra mejora y objetivo para la Time Machine que es la de poder modificar una única habitación. Para no tener que volver a la zona principal en caso de querer ver otra configuración ampliada se añadieron unos botones de navegación a cada lado del gráfico para que se pudiera navegar entre las distintas habitaciones.

En cuanto a la visualización de las configuraciones antiguas se planteó que sería mejor mostrarlas en una nueva pantalla para mejorar la navegabilidad y hacer un sistema más claro y manejable. Por esta misma razón tampoco estarían las opciones de ampliación ni de modificación. Además se pensó en mostrar con un fondo distinto las diferencias para que el usuario tuviera más claro los cambios que iba realizar.

Por último en cuanto interfaz, se añadieron distintas mejoras como añadir un reloj para ver la fecha actual, un dato importante de información cuando estamos hablando de un histórico, o una barra de información para que el usuario en todo momento pudiera saber la información (fecha y habitación) de los datos que esté visualizando.

El siguiente paso en el proceso de la Time Machine fue la de dotar de funcionalidad a los distintos apartados. En una primera aproximación los datos provenían de una clase estática que se dedicaba a dotar de datos a los distintos componentes como los gráficos o las fechas. Con estos datos de prueba se fueron diseñando las distintas funcionalidades de edición, navegabilidad o reconfiguración de la Time Machine.

Finalmente para dotar a la aplicación de más realismo y poder hacer un mejor uso de ella para el caso de estudio se decidió crear un modelo de persistencia en el que se almacenasen los datos de las distintas configuraciones, fechas, habitaciones etc. Este diseño de persistencia se explicará en detalle en el Anexo A juntamente con las funcionalidades internas de la Time Machine.

Uno de los últimos pasos fue el de crear una capa de comunicación entre la aplicación y el modelo de persistencia. Esta capa se ha hecho de forma que se pueda conectar a cualquier otro modelo de persistencia sin necesidad de realizar grandes cambios en el resto de la aplicación.

Por último se diseño un caso de estudio que demostrase la utilidad de la Time Machine y pusiera de manifiesto los beneficios que conllevaría su utilización. Se ha intentado plantear un caso de estudio lo más real posible con distintos usuarios y situaciones de la vida cotidiana en una Smart-Home en las que se pudiera requerir el uso de la Time Machine que se ha desarrollado para esta Tesina. Para ello se han elegido una serie de escenarios típicos donde se ven claramente las necesidades y bondades de la Time Machine. Se explica su uso en cada escenario y los resultados que ofrecería.

5.2. Diseño

El diseño que se ha creado para la aplicación se puede decir que es un modelo en tres capas (interfaz, lógica y persistencia) aunque, en algunas ocasiones, la lógica se mezcla con la capa de interfaz. Se ha realizado una aplicación configurable y adaptable a distintos casos de uso, de forma que aparte del caso de estudio se pueda utilizar para probar otros casos de estudio.

Además se ha intentado realizar un diseño interno claro basado en los distintos componentes gráficos que forman la Time Machine. Para ello se ha reutilizado el máximo código posible creando componentes de distinto tipo que se utilizan varias ocasiones dentro de la Time Machine.

Por último se han creado interfaces de conexión a los datos para que puedan modificarse fácilmente en caso de que fuese necesario, por ejemplo para adaptarlas a la hora de conectarlas con la Smart-Home o en el caso de querer utilizar otro tipo de sistema de base de datos.

Se ha intentado hacer configurable de forma que se pueden modificar los nombres o el número de las habitaciones (siempre que sea menor a 6) o las distintas opciones de configuración de la casa. De esta forma se pueden modificar las opciones de la Time Machine de forma que se pueda utilizar para probar los distintos casos de estudio.

5.2.1. Interfaz

En cuanto a la capa de interfaz se ha intentado reutilizar el máximo código posible creando componentes reutilizables para las distintas pantallas. Por ejemplo, la barra de fechas

es un componente que se reutiliza en las tres pantallas. A su vez la barra de fechas se compone de otros dos objetos: el panel indicativo de la fecha actual y el menú de fechas creado mediante la componente de Prefuse FishEye. Se han creado también componentes de menor importancia pero que se utilizan varias veces como los botones de ampliación o los botones de edición.

En cuanto al resto del diseño de la interfaz se han creado tres ventanas (diferencias, detalle y principal) que se añaden a la aplicación principal y que se van mostrando dependiendo de las acciones del usuario. Estas ventanas a su vez se han dividido en componentes gráficas más pequeñas como el panel central donde se colocan los gráficos de configuración o el panel de configuraciones.

Otros componentes de menor importancia son el panel de información (también presente en todas las pantallas) o el panel de reloj incluido en la componente que incluye el menú de fechas.

Se ha intentado reutilizar el máximo código posible para realizar un desarrollo lo más limpio y óptimo posible.

5.2.2. Lógica

La capa de lógica se ha intentado independizar de la capa de interfaz pero en muchos puntos está mezclado con esta. Se ha creado una clase de utilidad donde se han añadidos algunas de las funcionalidades referentes a la edición así como algunos de los esquemas y objetos lógicos que se utilizan durante la aplicación. Mediante una serie de métodos estáticos se realizan distintas funcionalidades útiles para el funcionamiento de la Time Machine.

Entre estos métodos de utilidad se encuentran métodos que se encargan de crear los objetos AggregateDemo que representan las configuraciones y almacenarlos en un vector o distintos métodos para el control de cambios y modificaciones (botones deshacer y rehacer). Además se incluyen otros métodos que nos permite obtener la fecha actual o el número de modificaciones que hemos realizado, entre otros. La funcionalidad concreta de cada método así como los distintos parámetros de entrada y salida de cada método se explican con detalle en el Anexo A.

El resto de funcionalidad está imbuida en los distintos componentes gráficos que conforman la aplicación como los distintos paneles, ventanas o componentes gráficos.

5.2.3. Persistencia

A rasgos generales lo que se ha pretendido es dotar a la aplicación de una capa donde sea posible guardar los datos. Al fin y al cabo se trata, entre otras cosas, de una aplicación para guardar un histórico de configuraciones y por tanto es lógico que se almacenen en algún sitio.

Además se ha aprovechado para dotar de mayor configurabilidad a la aplicación de forma que modificando los datos de la Base de Datos, como las habitaciones o los dispositivos que se quieren mostrar como opciones de configuración, se pueda adaptar a las opciones de cada sistema. Por supuesto también se podrán modificar los distintos dispositivos, servicios y canales de comunicación entre ellos para que se pueda adaptar a las necesidades específicas del sistema donde se ejecute.

La estructura que se ha creado para la base de datos gira en torno a la tabla de configuración. Está configuración estará relacionada con un usuario y una habitación y contendrá una fecha. A su vez se almacenarán los distintos dispositivos, servicios y canales y cada configuración se relacionará con los dispositivos, servicios y canales que tengan activos. La estructura de las tablas y de sus datos se comentará con más detalle en el Anexo A.

En cuanto a programación, se ha diseñado mediante una interfaz de comunicación una serie de métodos que comunican la aplicación con los datos de la BD. De esta forma si cambia el modelo de datos solo habrá que modificar esta clase y no habría que tocar el resto de la aplicación. Así los datos se podrían obtener de cualquier repositorio de almacenamiento de datos sin tener que tocar el resto de la aplicación. Mediante esta interfaz que hemos implementado podremos obtener los datos para la Time Machine, tanto los estáticos (nombre de las habitaciones, opciones) como los dinámicos (fechas, configuraciones).

Mediante esta interfaz tendremos las siguientes funcionalidades:

- Obtener los dispositivos activos en una habitación para la fecha indicada.
- Obtener todas las fechas de las que tenemos configuraciones guardadas que no pertenezcan a la configuración actual.
- Obtener el nombre de las habitaciones de la casa.
- Obtener todas las opciones de la time machine.
- Obtener los servicios activos en una habitación para la fecha indicada.
- Obtener los canales activos en una habitación para la fecha indicada.
- Obtener los servicios que están conectados a un determinado dispositivo
- Confirmar si una configuración de determinada fecha ha sido creada por el sistema o no.
- Borrar una configuración.
- Reconfigurar una sola habitación con las opciones elegidas por el usuario.
- Introducir una nueva configuración en el sistema de almacenamiento de la Time machine.
- Obtener el usuario correspondiente a una configuración.

Con estos métodos cuando se cree la componente AgreggateDemo (representación del gráfico de configuración), sabiendo la fecha y la habitación, podemos obtener los dispositivos y servicios para representar los gráficos de la componente y obtener los canales para representar los ejes entre dichos nodos.

Además a la hora de reconfigurar una habitación, o la casa entera, mediante estos métodos nos permitirán aplicar los cambios correspondientes. En nuestro caso los guardaremos en BD como veremos más adelante pero en el caso de estar conectada a la Smart-Home estos métodos, además, serán los encargos de hacer de conexión con el sistema de la Smart-Home y de aplicar las reconfiguraciones a la vivienda.

Para el desarrollo de la tesis y del caso de estudio se ha desarrollado una implementación de la interfaz comentada anteriormente basado en la conexión a una Base de Datos MySQL. Concretamente se ha elegido MySQL para el desarrollo de la Tesina porque es un sistema de Base de Datos sencillo y de código abierto. De esta forma almacenamos las distintas configuraciones en un modelo de persistencia para poder consultarlo en cualquier momento.

Dada la estructura que se ha creado para la Time Machine, esta parte está independizada del resto de la aplicación (interfaz y lógica) permitiendo, si desea, crear otra estructura de persistencia acorde a las necesidades del usuario. Por supuesto, en la Time Machine tal y como está montada se pueden cargar otras configuraciones y realizar otras pruebas y casos de estudios diferentes al planteado en la tesina.

Durante la ejecución de la Time Machine se irán actualizando los datos siempre que el usuario acepte los cambios. Hasta entonces únicamente los cambios que se muestran en la Time Machine únicamente son una representación pero no se guardan en el modelo de persistencia hasta que lo confirme el usuario.

5.2.4. Adaptación para otros casos

A la hora de adaptar la Time Machine para conectarla con la Smart-Home es muy posible que el entorno para obtener los datos no sea el mismo. Precisamente por ello la Time Machine se ha desarrollado para que sea sencilla de adaptar y con pocas modificaciones se pueda adaptar y obtener los datos desde otras fuentes.

Las opciones son múltiples. Se puede utilizar otro cliente de BD o bien guardar los datos en XML, para lo cual habría que adaptar algunos métodos. Además, como hemos comentado en apartados anteriores, la Time Machine es solo una demo que realmente no está conectada a la Smart-Home. Por ello aunque para el sistema de persistencia se quisiera mantener intacto habría que adaptar por lo menos, el método de reconfigurarHabitacion.

Los principales métodos de acceso a BD y por tanto a la información de la Time Machine están en la clase DatosTimeBDMySQL que implementa la interfaz DatosTime y por

tanto serán estos los que haya que modificar. Los datos de conexión habría que modificarlos en el constructor de dicha clase.

Es posible que en vez de modificar estos métodos, se prefiera mantenerlos y crear otra clase que extienda de la interfaz DatosTime. Si se sigue esta vía habría que modificar en las clases de la tabla 1 la línea mostrada en la figura 78 modificándola por la clase que creemos nosotros.

CLASES A MODIFICAR		
AggregateDemo	DatosTimeUtil	FisheyeMenu
MenuFechas	MenuOpciones	PanelCentral
TimeMachine	VentanaDiferencias	ZonaIndividual

Tabla 1 – Clases a Modificar

Por ejemplo, si la clase que implementamos se llama DatosTimeBDOracle tendremos que sustituir DatosTimeBDMySQL por el nombre de nuestra clase. El único requisito será el de que implemente la interfaz DatosTime. De este modo no hará falta modificar más código. Únicamente implementar los métodos que se declaran en la interfaz.

```
77      private DatosTime datos = DatosTimeBDMySQL.getInstance();
```

Figura 45 – Declaración clase acceso a datos

Concretamente para conectar la Smart-Home habrá que seguir algunos pasos básicos para conectar su funcionamiento con el de la Time Machine.

El modelo de características se tendrá que convertir al modelo DSPL para saber los dispositivos, servicios y canales de los que dispondrá el sistema. Para cada dispositivo, servicio y canal del modelo se deberá insertar un registro en las tablas dispositivo, servicio y canal correspondientes. Los atributos de la tabla dispositivo y servicio a parte del nombre llevarán asociados la ruta de la imagen representada. En el caso del canal a parte del nombre hará referencia al par servicio-dispositivo o servicio-servicio que conecte.

Por su parte cada característica del modelo de características que queramos que el usuario tenga disponible para aplicar como una configuración nueva debemos indicarlo en el campo esConfiguracion del dispositivo al cual esté asociado.

Cada característica llevará asociada una localización y por tanto se tendrá que asociar con la configuración de la habitación correspondiente guardado en la tabla configuración.

Por tanto cuando nos llegue una configuración deberemos insertar una entrada con la fecha y habitación correspondiente en la tabla configuración. Y un registro para cada dispositivo, servicio o canal que este activo en las tablas Configuracions_has_Dispositivo, configuracion_has_servicio y configuracion_has_canal correspondiente.

Posteriormente para conectar la Time Machine con la Smart-Home se deberá implementar el método reconfiguraHabitación para que envíe a la Smart-Home los datos de los servicios, dispositivos y canales que se activan o desactivan. En definitiva para que envíe la resolución correspondiente a las operaciones efectuadas en la Time Machine.

6. Caso de estudio

En este apartado se va a explicar un caso de estudio que ejemplifique el funcionamiento y las distintas funciones de la time-machine. El objetivo es comprobar las distintas funcionalidades de la herramienta desarrollada así como explicar los distintos posibles usos y demostrar su utilidad en un caso real.

Para desarrollar el caso de estudio primero se va a explicar en rasgos generales la situación en la que se va a desarrollar este. Después se explicarán las funciones que tendrán nuestra Smart-Home y que servirán para ejemplificar mejor las distintas situaciones del caso de estudio. Además se van a usar una serie de usuarios que nos ayudaran a comprender los usos de la Time Machine en una ambiente donde van a interactuar distintos usuarios con distintos permisos. Por último desarrollaremos los ejemplos y situaciones para mostrar la utilidad de la Time Machine.

6.1. Ubicación del caso de estudio

El caso de estudio se va a ubicar en una vivienda familiar donde se dispondrán de diversos servicios para facilitar la vida a una familia. La casa dispone de un sistema de Time Machine donde se guarda un histórico de las configuraciones de una casa. Además permite crear nuevas configuraciones manualmente y guardarlas para que la casa se reajuste a los cambios indicados. Las estancias de las que va a disponer la vivienda para el caso de estudio serán:

- Dormitorio.
- Salón/Comedor.

- Cocina.
- Cuarto de Baño.
- Dormitorio Infantil.
- Habitación de Invitados.

6.2. Usuarios de la Smart-Home

Se han identificado una serie de usuarios básicos pero suficientes para identificar los casos más representativos de uso, así como las distintas restricciones dependiendo del usuario. Nuestra Time Machine va a tener tres tipos de usuarios representados por el diagrama de usuarios que vemos en la figura 45. Como vemos en el diagrama estos usuarios siguen una jerarquía en la que cada uno tiene nuevos permisos sobre el anterior. Vamos a explicar los usuarios empezando por el usuario con menos privilegios, el usuario con permisos de habitación, y acabando por el usuario con más privilegios, el usuario administrador.

- **Usuario con Permisos de Habitación:** Este usuario únicamente puede ver modificar y reconfigurar las configuraciones correspondientes a una habitación. Está pensado para invitados ocasionales para que puedan estar a gusto en su habitación y manejar su estado.

- **Usuario con Permisos de Usuario:** Este usuario puede ver las configuraciones de todas las habitaciones pero únicamente las que haya creado él o el sistema automáticamente. Está pensado para familiares o amigos de mayor confianza que puedan venir de visita y quieras que se sientan cómodos pero no deseas que vean las configuraciones creadas por otros usuarios, permitiendo así una mayor privacidad.

- **Usuario con Permisos Totales:** Este usuario puede ver y modificar todas las configuraciones de todos los usuarios. La única opción que no tiene disponible es la de borrar las configuraciones. Está pensado para técnicos del sistema que necesiten hacer operaciones sobre la Time Machine.

- **Usuario Administrador:** Este usuario tiene permisos sobre todo los demás. Por ello puede ver y modificar todas las configuraciones que haya creado cualquier usuario. Además podrá

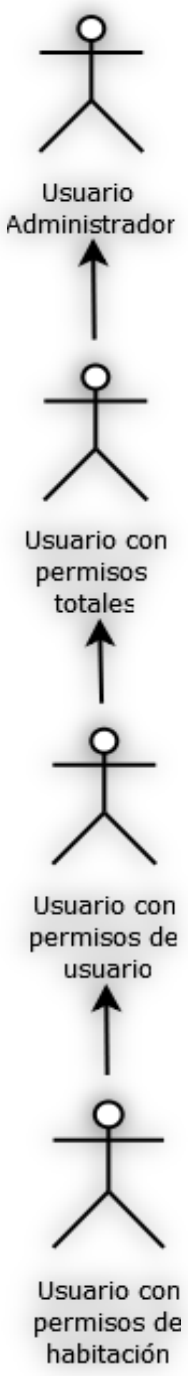

Figura 46 – Diagrama de Usuarios

borrarlas si lo desea.

Los usuarios que van a utilizar la Smart-Home en nuestro caso de estudio y por tanto la Time Machine que hemos desarrollado son los siguientes. En ellos se ven representados los roles comentados anteriormente.

- **Propietario**: Propietario de la vivienda. Control total sobre la Time Machine. Tiene el usuario de Administrador.

- **Servicios externos**: Servicio técnico de los servicios externos proporcionados a la casa. Este usuario podrá mover la configuración de la casa a una anterior para hacer mediciones. Tiene el perfil de usuario con permisos totales

- **Invitados:** Invitados de la casa. Podrá modificar la configuración de la habitación de invitados viendo únicamente las configuraciones creadas por él mismo. Tiene el rol de usuario con permisos de habitación.

- **Familiares:** Invitados familiares de la casa. Podrá modificar la configuración de toda la casa viendo las configuraciones creadas por él mismo. Tiene el perfil de usuario con permisos de usuario.

- **Servicio Domestico:** Servicio domestico como asistenta o niñera. Tiene el perfil de usuario con permisos de usuario.

6.3. Funciones de la Smart-Home

La Smart-Home que hemos definido para nuestro caso de estudio consta de los siguientes servicios básicos sobre los que se fundamentan las funcionalidades de la Time Machine:

- **SEGURIDAD:** sistemas de seguridad y/o alarma antirrobos mediante sistemas acústicos o luminosos. Dentro de estos sistemas de seguridad también se podrían incluir sistemas de seguridad antiincendios. Los dispositivos disponibles para este servicio serán:
  - Visual Alarm
  - Siren
  - Blinking Lights
- **ILUMINACION:** sistemas de iluminación automática dependiendo del momento del día o de la presencia de gente en la casa o estancia. Los dispositivos asociados para este servició son:
  - Gradual Lights

- **CLIMATIZACION:** sistemas de climatización ajustables a las necesidades de la familia en las diversas situaciones y las distintas estancias de la casa. Los dispositivos asociados a este servicio son:
    - Sensor de temperatura.
- **GESTIÓN MULTIMEDIA:** Sistemas para el uso de los distintos dispositivos multimedia desde el teléfono móvil, reproductor de música o cadena musical. Los dispositivos asociados a este servicio son:
    - Ipod
    - Dispositivo Móvil
    - VLC
- **SISTEMA DE PRESENCIA:** sistema para el control infantil especialmente diseñado para la vigilancia de los niños cuando estos se encuentren en alguna habitación de forma que los padres sean avisados si ocurriera algo. Los dispositivos asociados a este servicio son:
    - Sensor de presencia.

6.4. Modelo de características

En la siguiente figura podemos observar el modelo de características, siguiendo la semántica que hemos explicado en anteriores apartados de la tesina, que representa la configuración de nuestra casa y sus posibles variaciones. En él podemos observar las distintas características de las que dispone la Smart-Home que vamos a utilizar en el caso de estudio y las diferentes posibilidades de activación y desactivación de características de las que disponemos. Como hemos visto antes, las características activas las representaremos con un rectángulo con fondo gris mientras que las inactivas las representaremos con un rectángulo de fondo blanco. En este caso hemos representado como que todas las características de la casa están activas.

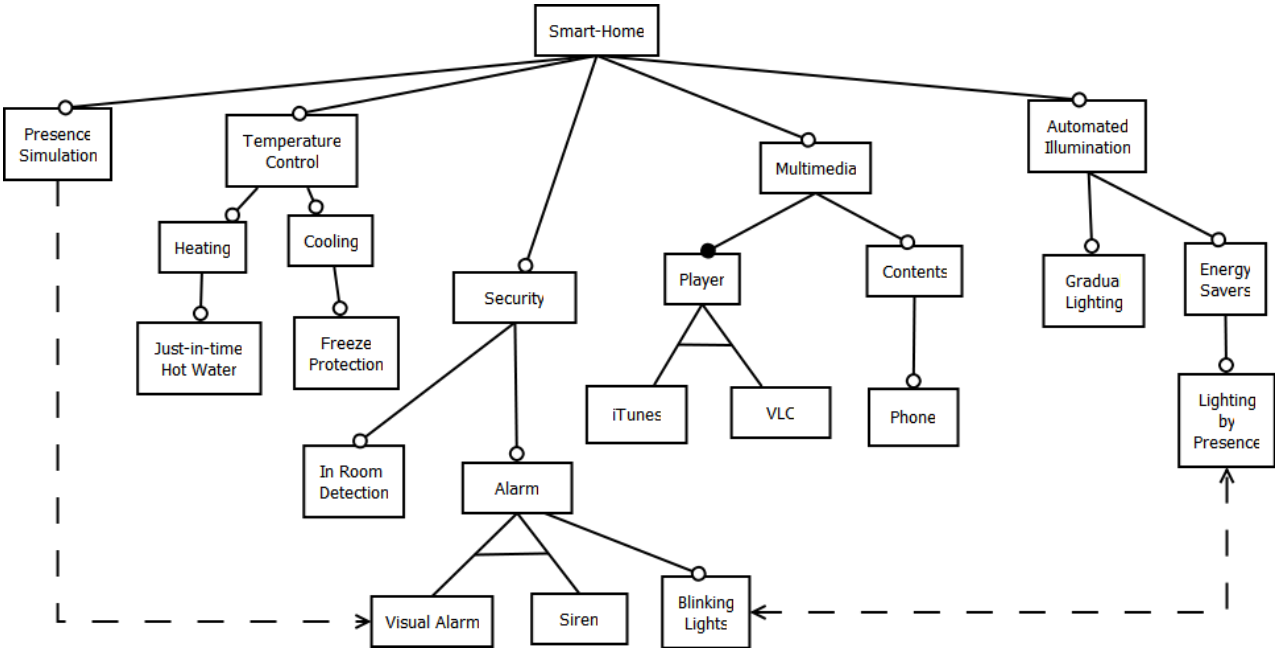

Figura 47 – Modelo de características.

6.5. Esquema PervML

Para poder visualizar mejor las reconfiguraciones, tenemos el esquema PervML siguiendo la simbología que también hemos explicado antes. En ella podemos encontrar los servicios representados por círculos, los dispositivos representados por cuadrados y los canales de comunicación representados mediante líneas entre ellos. Estos dispositivos, servicios y canales aparecerán y desaparecerán dependiendo de la configuración. En la figura 47 podemos observar el esquema en la sintaxis tradicional de PervML, en las figuras que mostraremos más adelante podremos ver partes de este esquema con la representación gráfica que se le va a mostrar al usuario, mucho más visual e intuitiva de ver a primera vista para alguien que no tenga conocimientos de la sintaxis especifica.

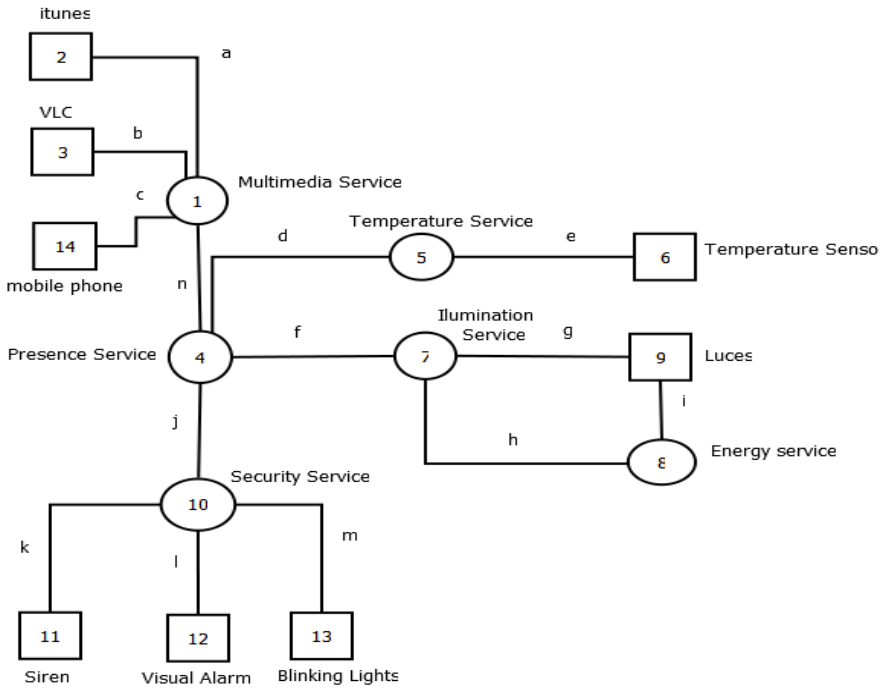

Figura 48 – Esquema PervML

En la siguiente figura podemos ver una correspondencia entre el modelo de características, el esquema PervML y el esquema de la TimeMachine. En ella podemos ver como se relacionan las características referentes a la seguridad del modelo de características con los servicios y dispositivos del esquema PervML.

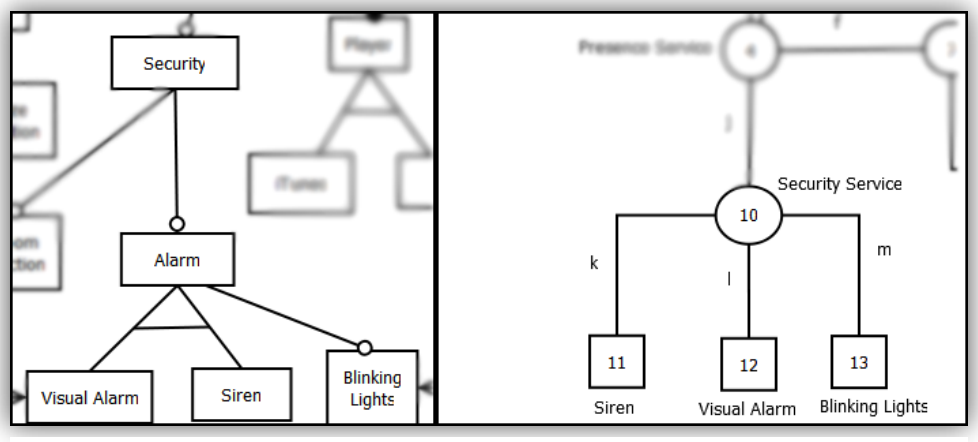

Figura 49 – Comparativa entre Model Feature y esquema PervML

6.6. Descripción global

A continuación se va a dar una descripción global del caso de estudio que posteriormente dará lugar a los distintos escenarios y las distintas demostraciones sobre la Time Machine.

Javier, un amigo de la familia, viene de visita a la casa de nuestros usuarios y se va a quedar un par de días en la habitación de invitados. Javier viene con su Smartphone y la casa detecta que hay nuevos dispositivos. Se guarda una configuración automáticamente en la Time Machine incluyendo este dispositivo. Cuando Javier llega a su habitación busca en la Time Machine las configuraciones guardadas por él (solo puede ver las suyas) y elige la que prefiere para esa habitación. Cuando Javier se va de la casa se reconfigura de nuevo con las preferencias anteriores quitando de la configuración los dispositivos que se lleva con él.

Durante el uso normal de la vivienda se estropea el dispositivo sonoro de seguridad. La casa se reconfigura para utilizar las luces como alarma de seguridad. Se guarda una nueva configuración en la Time Machine con la nueva configuración.

Llega la asistenta a la casa y esta se adapta a ella, pero desea modificarla. La modifica y la guarda para que a partir de ahora se use esa configuración.

Por un fallo en el sistema, este falla y va el servicio técnico a revisarlo. Pone la última configuración en la que el aparato funcionaba y detecta el fallo. Lo soluciona y devuelve la casa al estado en el que estaba.

Los padres de los propietarios vienen a pasar unos días. Buscan en la Time Machine la configuración creada por ellos para la casa y la reconfiguran. Los propietarios llegan a la casa pero la configuración que han puesto sus padres no es de su gusto así que deciden cambiarla y ponen la que había anteriormente. Además activan los sistemas multimedia en el comedor para relajarse un poco. Cuando se acaba la visita de los padres los propietarios aprovechan para borrar las configuraciones que habían guardado. Ya de paso aprovechan para borrar las configuraciones antiguas que ya no desean.

6.7. Descripción detallada de los escenarios

Se va a proceder a realizar una descripción detallada de los escenarios. Para ello se ha dividió el caso de estudio anteriormente descrito en seis escenarios distintos de aplicación donde se van a poner en práctica todas las funcionalidades de la Time Machine. Con ello demostraremos su utilidad en los distintos casos y explicaremos de forma práctica su funcionamiento. Para detallar los distintos escenarios nos vamos a basar en capturas de

pantalla mostrando lo que vería el usuario en el caso concreto de actuación. De esta forma podremos ejemplificar el caso de estudio y ver cuál sería el funcionamiento de la Time Machine.

Al principio de cada escenario se va a indicar en cursiva la parte del caso de estudio al que hace referencia para que podamos ejemplificarlo mejor y explicaremos cual es la motivación de cada escenario.

6.7.1. La casa se reconfigura automáticamente

Javier, un amigo de la familia, viene de visita a la casa de nuestros usuarios y se va a quedar un par de días en la habitación de invitados. Javier viene con su Smartphone y la casa detecta que hay nuevos dispositivos. Se guarda una configuración automáticamente en la Time Machine incluyendo este dispositivo.

La motivación principal de este escenario es la de ver como la casa al reconfigurarse es capaz de guardar una configuración en el histórico de configuraciones de forma automática.

Descripción

Javier viene a pasar unos días a la ciudad y le hemos invitado para que pase unos días con nosotros. Durante el tiempo que esté en nuestra casa dormirá en la habitación de invitados. Después de recibirlo le acompañamos a su habitación para que se ponga cómodo. Cuando Javier entra en la habitación esta detecta que Javier lleva consigo un dispositivo móvil. La habitación incluye este dispositivo en su configuración y la Time-Machine, que ha detectado un cambio guarda una nueva configuración en su Base de datos.

La casa se reconfigura automáticamente. La Time Machine muestra la nueva configuración y pasa la configuración que hasta ahora era la actual al histórico. En las figuras podemos ver cómo era la configuración antes y después de reconfigurarse automáticamente. En la figura se muestra la visión que tendría el usuario administrador de la casa.

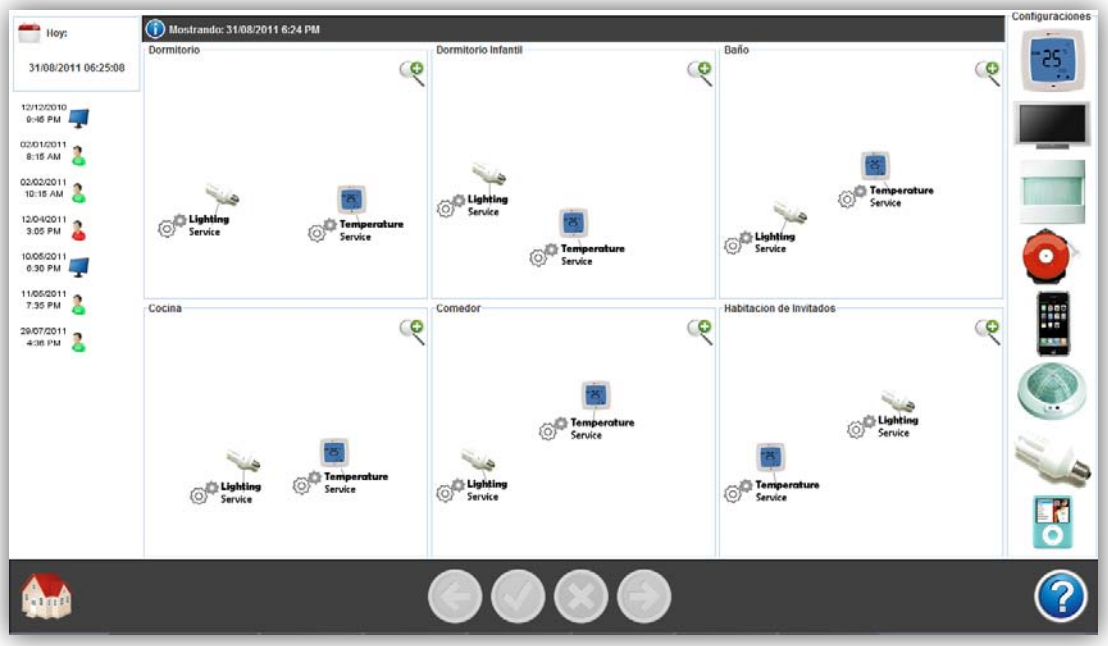

Figura 50 – Time Machine en estado habitual

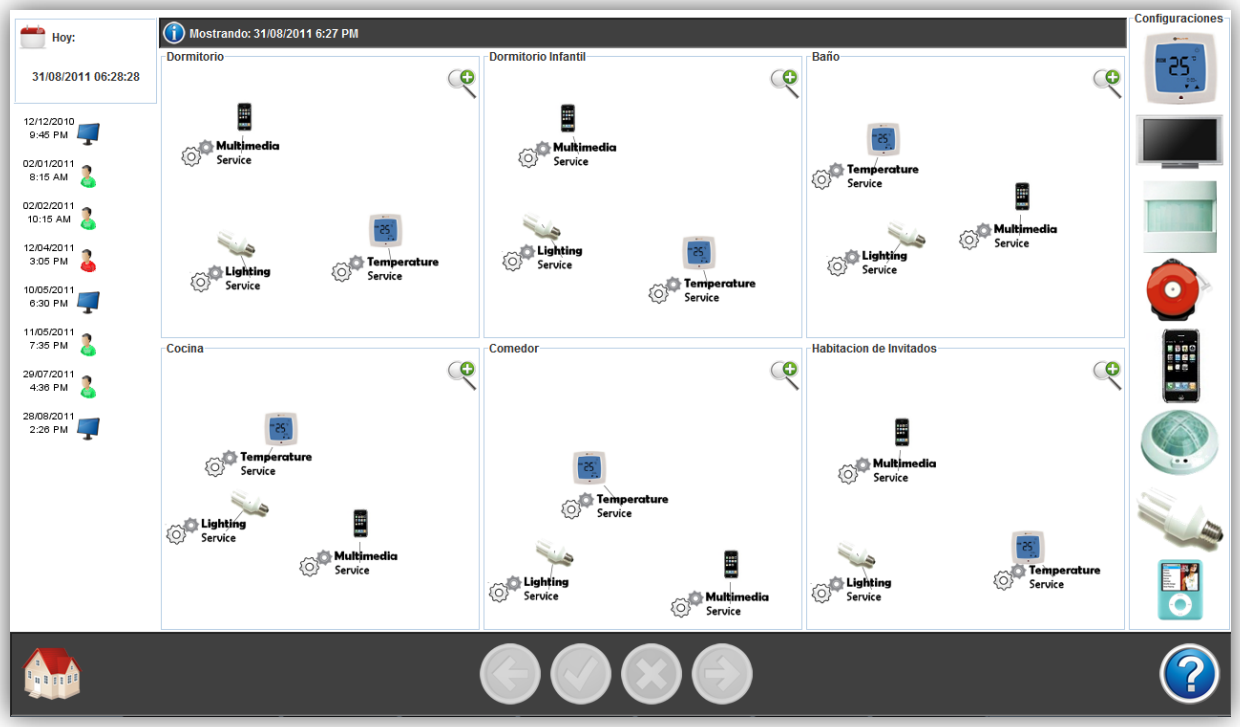

Figura 51 – Time Machine tras la reconfiguración automática

6.7.2. Usuario con restricciones de habitación reconfigura una habitación

Cuando Javier llega a su habitación busca en la Time Machine las configuraciones guardadas por él (solo puede ver las suyas) y elige la que prefiere para esa habitación. Cuando Javier se va de la casa se reconfigura de nuevo con las preferencias anteriores quitando de la configuración los dispositivos que se lleva con él.

La motivación de este escenario es la de ejemplificar el uso de la Time Machine en un usuario que solo tiene permisos sobre una única habitación y ver como se reconfigura únicamente una habitación.

Descripción

Después de acomodarse en la habitación de invitados, Javier se acerca a la Time Machine para comprobar la configuración de su habitación. Tiene la sensación de que sus amigos no tienen activado el sistema de temperatura. Cuando abre la Time Machine, esta detecta su nivel de usuario y solo le muestra las configuraciones guardadas por el sistema automáticamente y sus configuraciones, siempre para esa habitación únicamente. Cuando estuvo de visita hace tiempo ya tubo el mismo problema y guardo una configuración personalizada. La busca y reconfigura la casa. Ahora su habitación está perfecta para pasar los próximos días.

Cuando el usuario abre la Time Machine la pantalla principal únicamente le ofrece la configuración de la habitación de invitados como podemos ver en la siguiente figura.

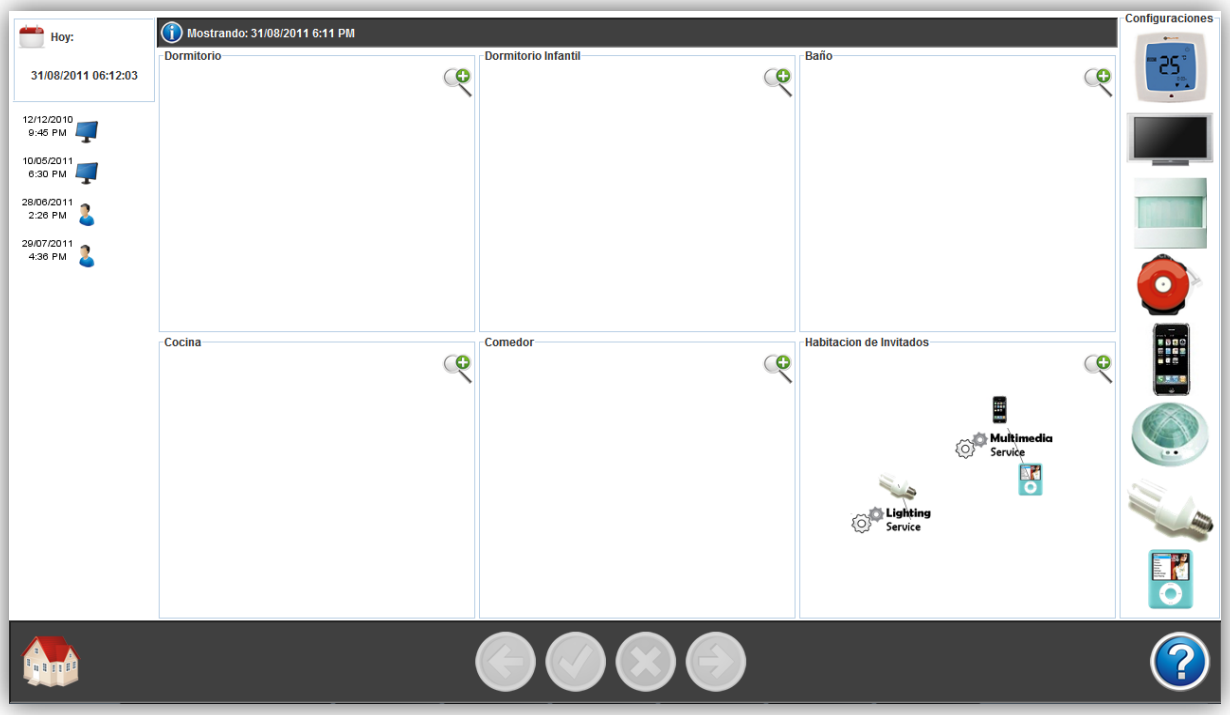

Figura 52 – Vista general usuario con permiso de habitación

Además en el histórico solo puede ver las configuraciones creadas por el o por el sistema. Cuando entra en una configuración pasada también se le muestra solo la configuración de su habitación. Como podemos ver en la figura busca una habitación donde esté activo el sensor de temperatura, la alarma y desactive el hilo musical.

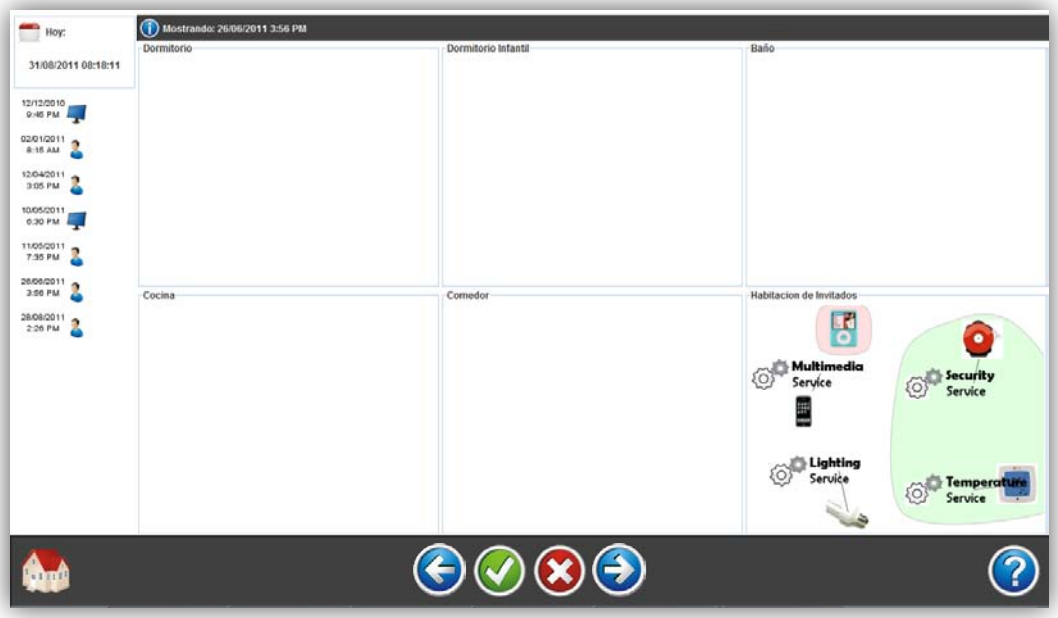

Figura 53 – Vista comparativa usuario con permiso de habitación

Tras darle a aceptar la casa se reconfigura y la configuración elegida pasa a ser la configuración actual. Como podemos ver en la figura 53, ahora en la pantalla principal que representa la configuración actual tenemos la configuración elegida por el usuario.

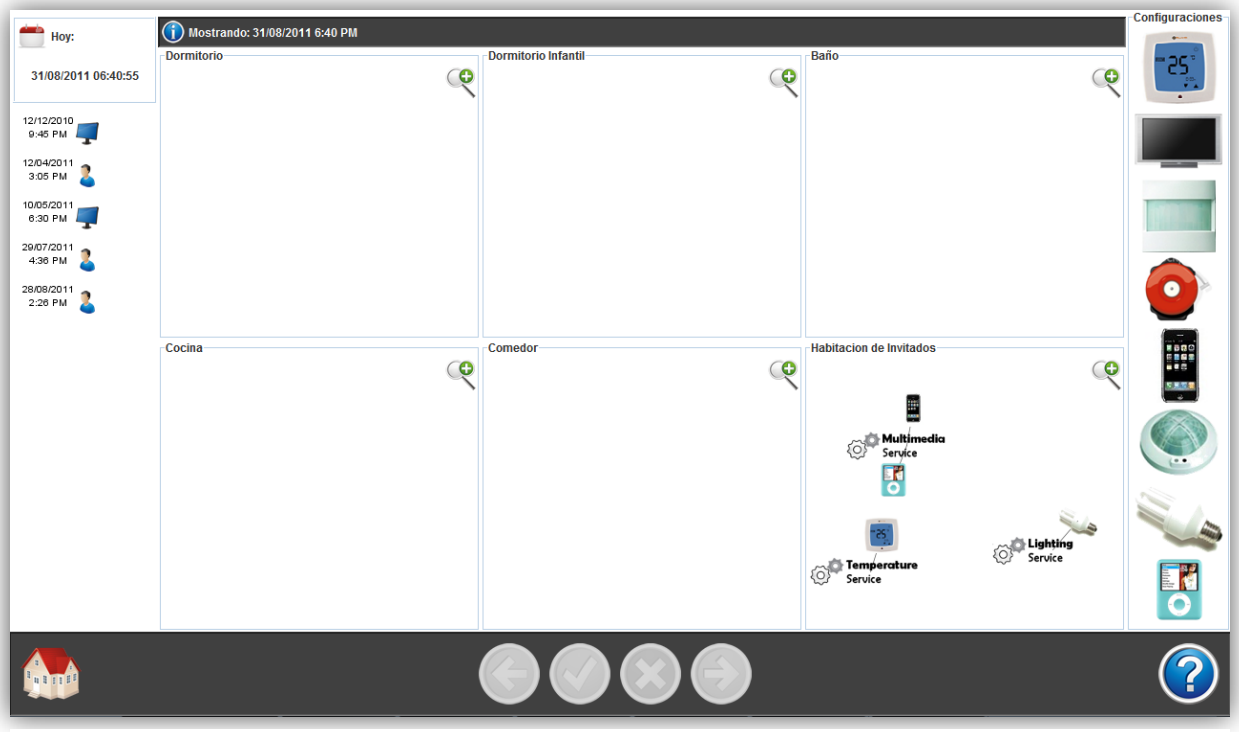

Figura 54 – Reconfiguración de usuario con permiso de habitación

6.7.3. Back-up de la configuración de la casa

Por un fallo en el sistema, este falla y va el servicio técnico a revisarlo. Pone la última configuración en la que el aparato funcionaba y detecta el fallo. Lo soluciona y devuelve la casa al estado en el que estaba.

La motivación de este escenario es la de ver como se realizaría un back-up de la casa, en este caso se debe a un fallo en el sistema pero podría deberse a una decisión personal del usuario.

Descripción

Cuando Francisco, el dueño de la vivienda llega a la casa se da cuenta de que el sistema ha fallado y la Smart-Home no está funcionando como debería. Decide llamar al servicio técnico y tras detectar que no puede hacer nada de forma telemática envía a un técnico para que revise la instalación. El técnico, visualizando las últimas configuraciones guardadas en la Time Machine y reconfigurando la casa a los estados anteriores en los que sí que funcionaba, detecta donde se ha producido el error y porque no se ha reconfigurado bien la Smart-Home cuando este se ha producido. Lo repara y retorna la Time-Machine a su configuración actual.

El usuario busca entre las configuraciones antiguas. Mediante los botones adelante y atrás se va moviendo por las configuraciones de la casa hasta encontrar una que funcione. Como podemos ver en la figura inferior no tiene disponible la opción de borrar. Pero puede ver las configuraciones de todos los usuarios representados por varios iconos en el histórico.

Figura 55 – Búsqueda en configuraciones antiguas

Cuando encuentra la configuración deseada mediante el botón de aceptar se reconfigura la casa con esa opción y en la pantalla principal nos muestra la configuración ya aplicada a la Smart-Home como se muestra en la figura 55.

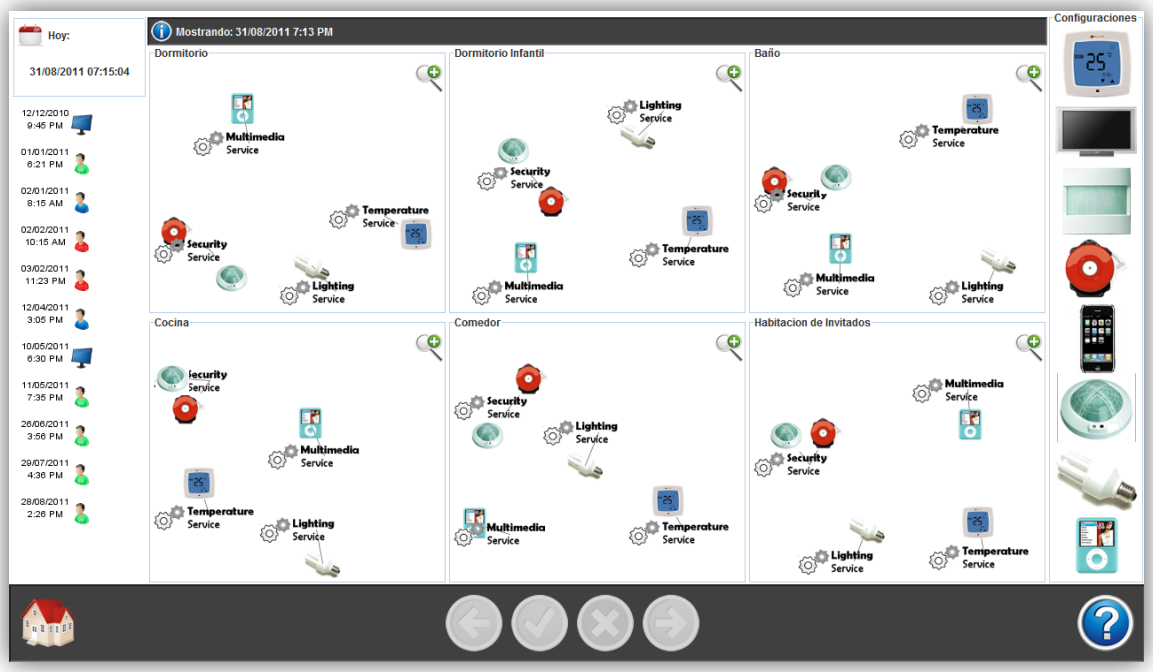

Figura 56 – Reconfiguración de la casa

Para saber que dispositivo es el que ha fallado va probando uno a uno los dispositivos añadiéndolo mediante las opciones de configuración laterales y quitándolos mediante las opciones de hacer y deshacer de la barra de edición anterior. En las figuras 56 y 57 vemos el proceso de aplicar configuración y de deshacer una configuración. En la primera podemos ver como se ha añadido el dispositivo de video y en la segunda se ha deshecho esa configuración volviendo al estado anterior. Los botones de edición se activan y desactivan como ya hemos explicado en anteriores apartados.

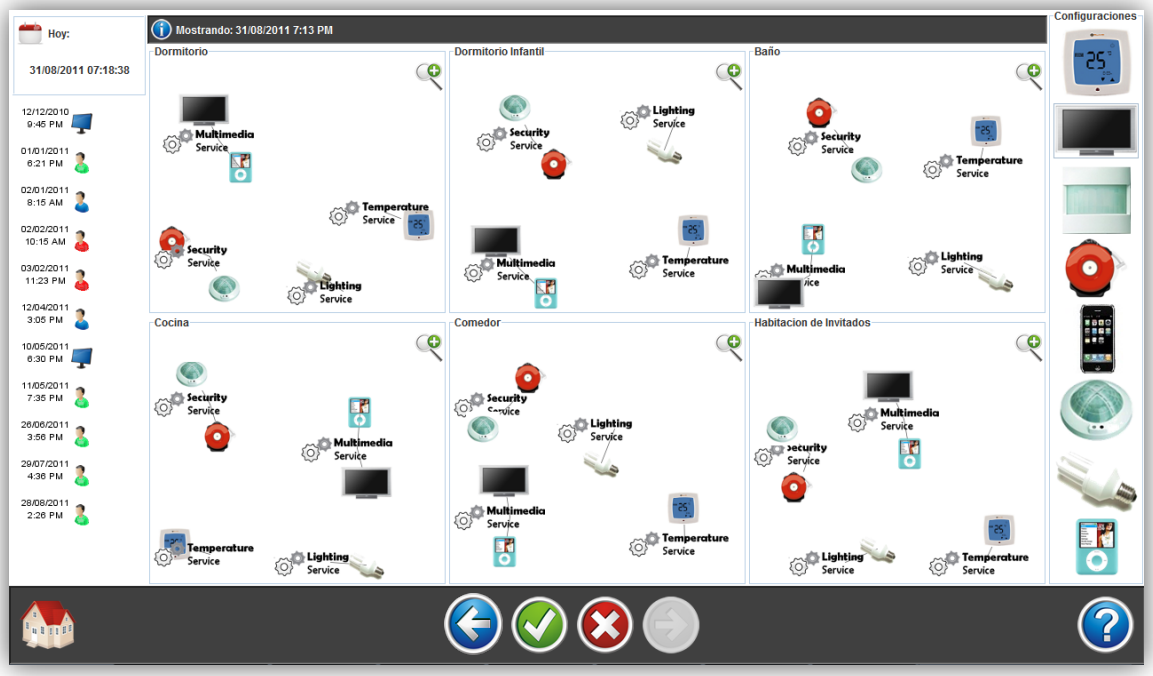

Figura 57 – Aplicar configuración

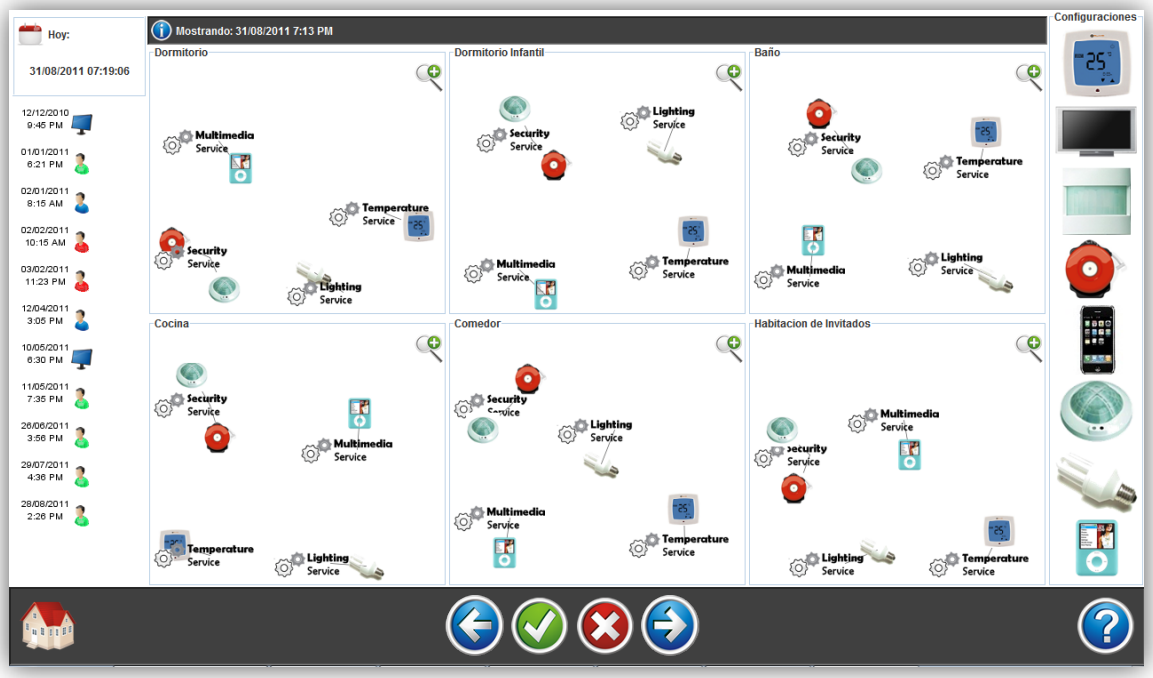

Figura 58 – Deshacer configuración

Cuando detecta cual es el dispositivo que falla lo repara y reconfigura la casa con la configuración que tenía antes de que se produjera el error. En esta figura podemos ver cómo queda la casa tras la reconfiguración.

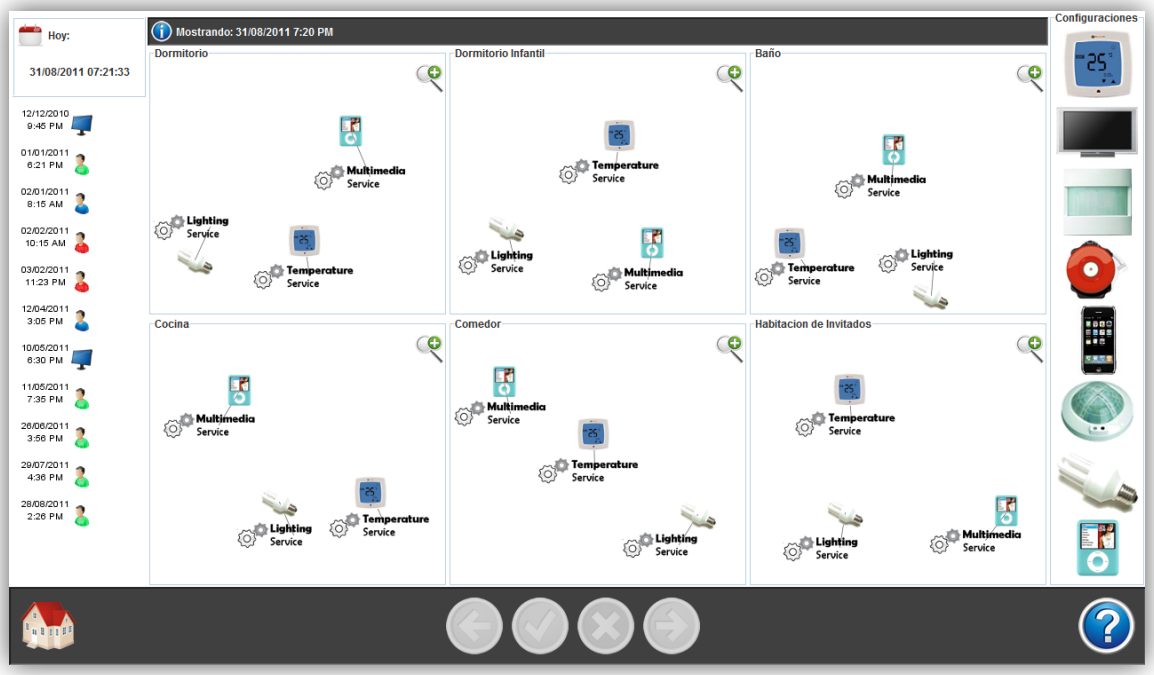

Figura 59 – Reconfiguración de la casa

6.7.4. Usuario con restricciones reconfigura la casa

> *Los padres de los propietarios vienen a pasar unos días. Buscan en la Time Machine la configuración creada por ellos para la casa y la ponen.*

El propósito de este escenario es el de mostrar el uso de un usuario con restricciones de usuario buscando una configuración antigua y reconfigurando la casa con ella.

Descripción

Los padres de Francisco, Miguel y Victoria, vienen a pasar unos días para cuidar de sus nietos, Iván y Quico, mientras los padres de estos están de viaje. Cuando llegan a la casa y después de acomodarse en la habitación de invitados se acercan a la Time Machine. Como son previsores, en su día, la primera vez que se quedaron a pasar unos días guardaron una configuración para poder cargarla cada vez que vinieran de visita. Buscan en el listado donde

pueden ver sus configuraciones y las guardadas por el sistema para toda la casa y eligen la deseada y reconfiguran la casa. Ahora ya pueden disfrutar de su estancia y de sus nietos.

Como podemos ver este usuario puede ver toda la casa pero únicamente sus configuraciones o las del sistema.

Figura 60 – Vista de usuario con permiso de usuario

Cuando elige una de sus configuraciones la casa se reconfigura al igual que hemos visto en el caso anterior.

6.7.5. Modificación de la configuración actual.

Los propietarios llegan a la casa pero la configuración que han puesto sus padres no es de su gusto así que deciden cambiarla y ponen la que había anteriormente. Además activan los sistemas multimedia en el comedor para relajarse un poco.

La motivación de este escenario es la de modificar la configuración actual añadiendo nuevas características y reconfigurar la casa con ella. En este caso aunque se haya copiado todo el párrafo para darle sentido, realmente es solo la última frase lo que vamos a ejemplificar ya que el resto ya se ha demostrado en los demás escenarios.

Descripción

Francisco y su pareja vuelven cansados de viaje y cuando llegan encuentran la configuración de su casa patas arriba. Tras saludar a sus hijos se dirigen a la Time Machine y restauran la casa al estado en el que estaba cuando se fueron. Además tienen la impresión de que sus padres han sido un poco blandos con sus hijos y aun no tienen los deberes hechos. Manda a sus hijos a su habitación para que estudien y activan los sistemas multimedia en el comedor para relajarse mientras.

Cuando abre la Time Machine este usuario se le muestran todas las configuraciones en el histórico, tanto las suyas como las de los demás usuarios.

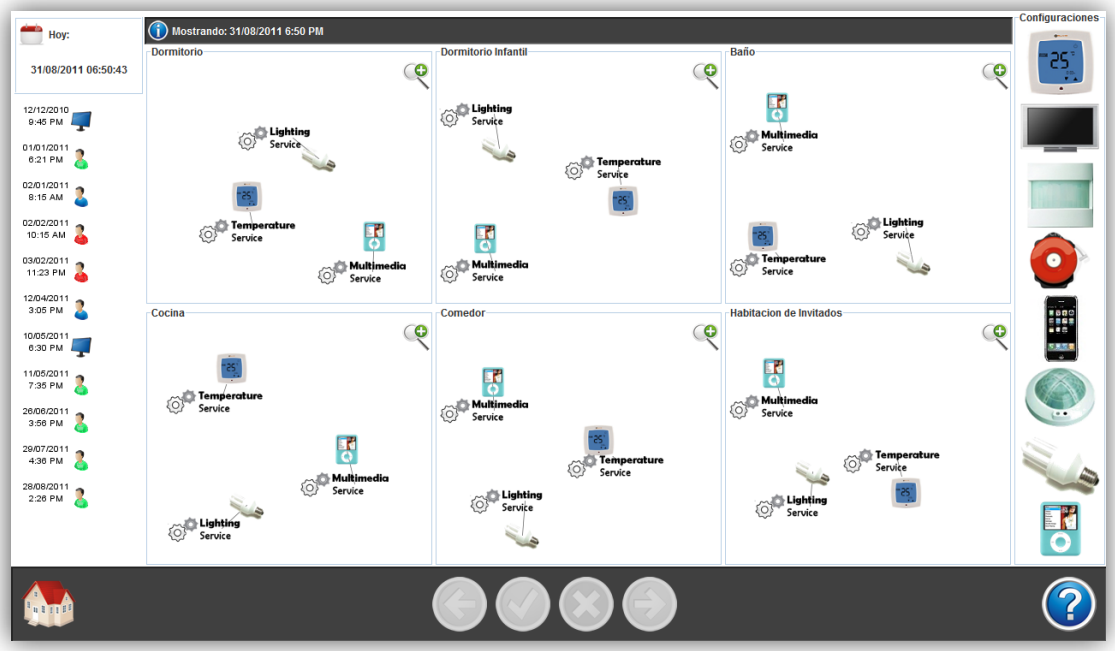

Figura 61 – Vista de usuario con permisos totales

Además se mete en el detalle de la configuración del comedor y se le muestra su configuración de forma más ampliada.

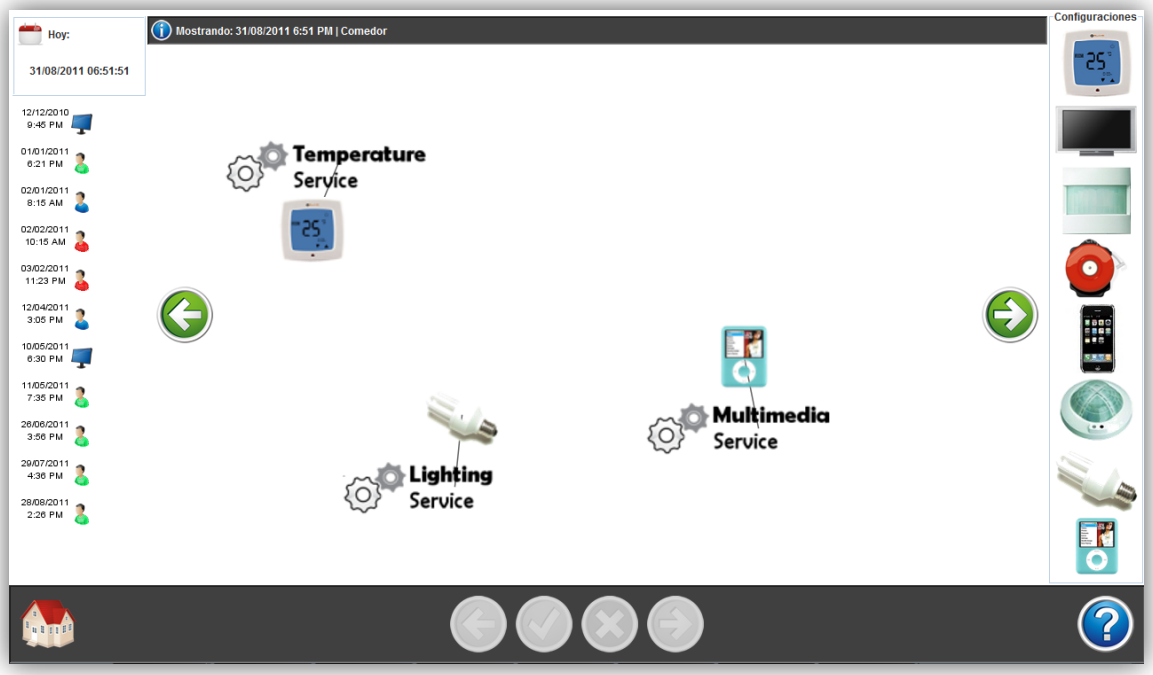

Figura 62 – Vista detallada de usuario con permisos totales

Añade la configuración deseada y acepta los cambios. Mostrándose la habitación reconfigurada en la figura 62.

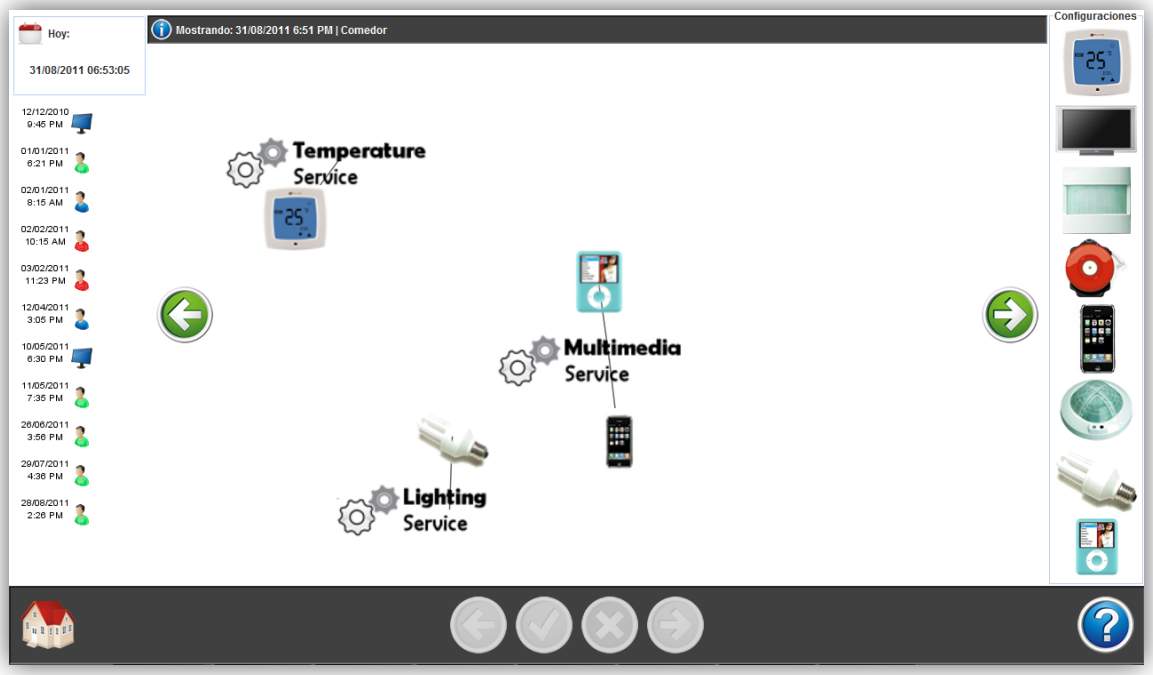

Figura 63 – Aplicar configuración en vista detallada y acepta cambios

Por último vuelve a la ventana principal para ver cómo ha quedado la configuración global de la casa.

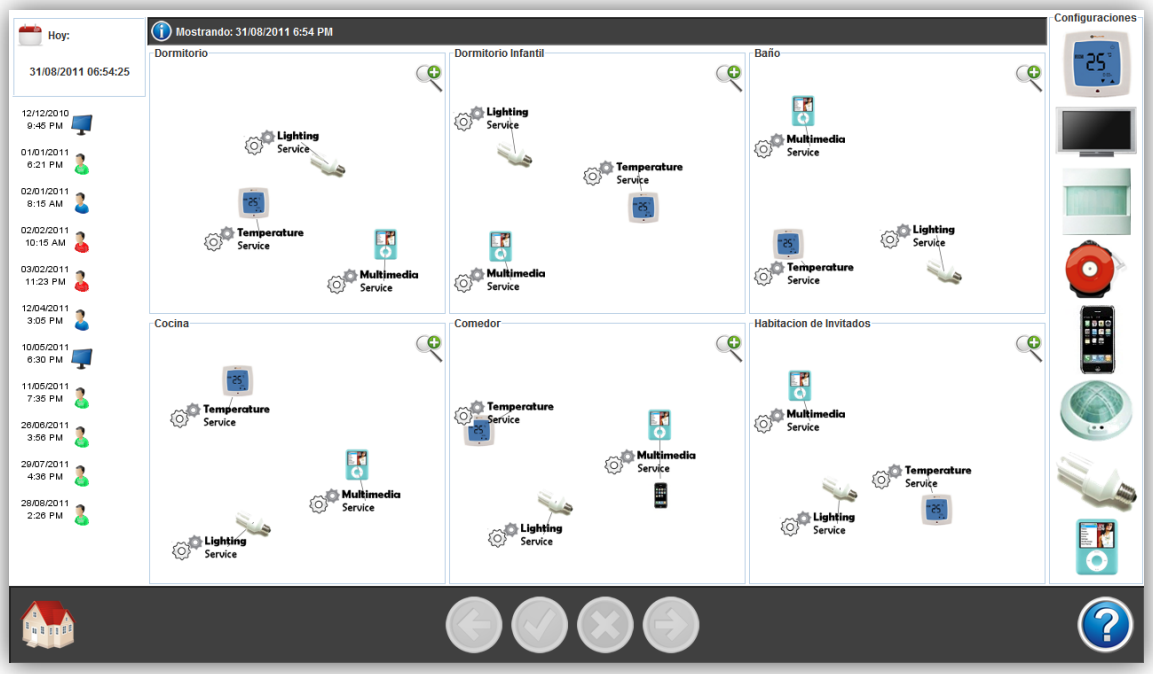

Figura 64 – Vista global tras reconfiguración de una habitación

6.7.6. Borrado de una configuración

Cuando se acaba la visita de los padres los propietarios aprovechan para borrar las configuraciones que habían guardado. Ya de paso aprovechan para borrar las configuraciones antiguas que ya no desean.

La motivación en este escenario es la de ejemplificar el borrado por completo de una configuración por decisión del usuario.

Descripción

Cuando se van sus padres Francisco decide poner orden en la Time Machine, hay demasiadas configuraciones guardadas y hay muchas que ya no necesita. Decide hacer una limpieza e ir borrando algunas configuraciones tanto suyas, como del sistema, como de otros

usuarios que han estado en la casa y han guardado configuraciones tanto para toda la casa como para alguna habitación en concreto.

El usuario va moviéndose por las distintas configuraciones borrando mediante el botón borrar las configuraciones que no desea. Como podemos ver este usuario sí que tiene activado el botón de borrar.

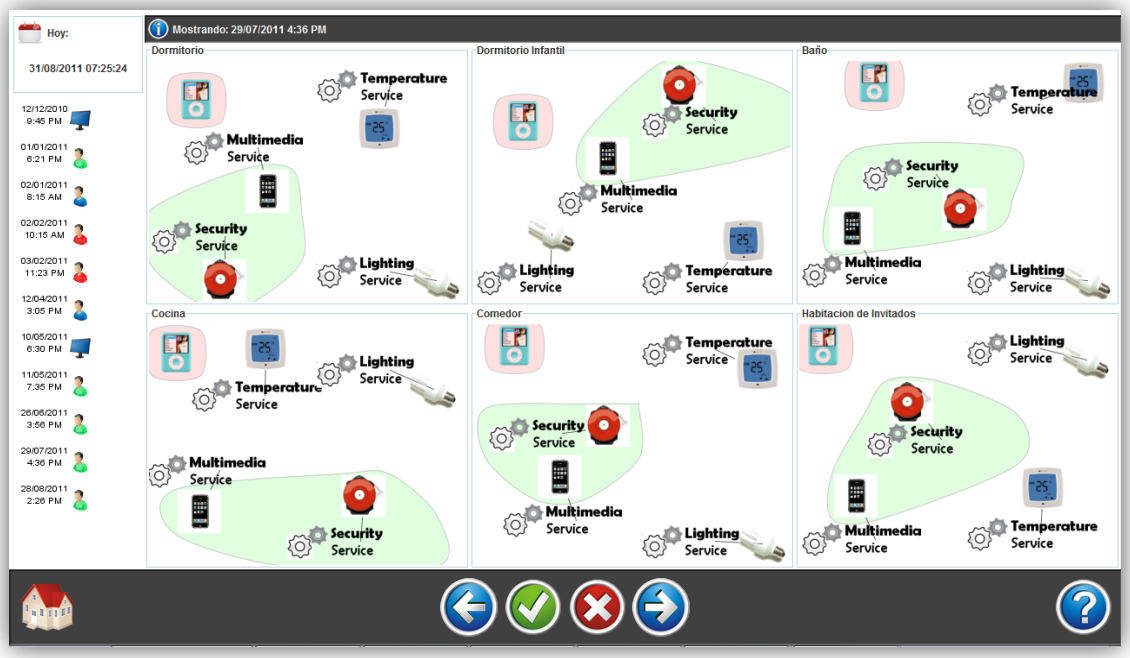

Figura 65 –Movimiento por las configuraciones antiguas

7. Conclusiones

Figura 66 – Alcance del capítulo 7

A lo largo del documento se han explicado las metodologías y las técnicas para la gestión y desarrollo de software cambiante.

Esta tesis se ha desarrollado en dicho contexto poniendo especial hincapié en los modelos de características y los modelados de variabilidad y en la representación de la información, que es al fin y al cabo lo que estamos realizando. Dentro de este contexto la variabilidad del sistema y las múltiples configuraciones que puede tener la casa en el tiempo nos hizo pensar en una solución que nos permitiera llevar un control de estos cambios.

El objetivo de esta tesis era el de crear una aplicación para representar dicha información y almacenarla en el tiempo. Permitiendo al usuario consultarla y poder volver a un punto en el pasado. De este modo aparte de un sistema de seguridad ofreceríamos al usuario una visión gráfica de la configuración de la Smart-Home y una zona donde interactuar con ella.

Puesto que iba a estar de cara al usuario debía de ser sencilla y atractiva ya que no se trata de una herramienta de trabajo si no de una herramienta de uso diario para un usuario que no tiene porque conocer la metodología en la que está basada estas representaciones y sobre la que se ha desarrollado la Time Machine, de hecho lo más probable es que no la conozca.

Para demostrar los beneficios del desarrollo y toda su funcionalidad se ha diseñado un caso de estudio bastante completo en el que mediante diferentes escenarios se ha ejemplificado la funcionalidad total del sistema desarrollado. Para hacerlo más real se ha

metido el concepto de usuarios con distintos permisos aunque el desarrollo completo de este aspecto se ha quedado fuera del objetivo de la tesina.

Este trabajo proporciona por tanto una herramienta funcional preparada para integrar con la time-machine y probar su funcionamiento en un entorno real.

Por tanto, por todo lo dicho antes, se puede decir que se han cumplido los objetivos iniciales de la tesina que se trataba de crear una herramienta sencilla, una metodología para guardar las configuraciones y se ha ofrecido una demostración de cómo funcionaria la Time Machine. Pese a todo, esta aplicación es una primera aproximación y como veremos a continuación hay varias líneas de trabajo por las que se puede seguir trabajando.

8. Trabajos Futuros

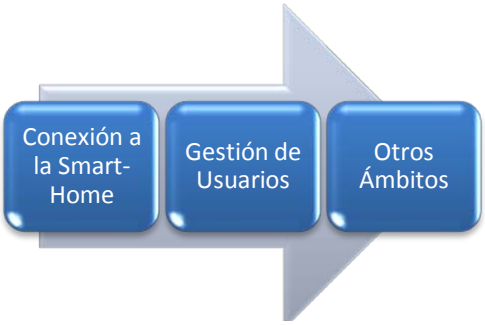

Figura 67 – Alcance del capítulo 8

Hay temas que no se han tratado dentro de la tesina o que se han tratado pero que quedan pendientes para futuros estudios ya que debidos a su complejidad su desarrollo completo quedaba fuera del ámbito de esta tesina.

Entre los trabajos futuros más destacables estarían el de realizar una gestión completa de los usuarios de la Time Machine, así como la de realizar una conexión real con la Smart-Home. Además sería interesante además de en la Smart-Home probar la time-machine en otros ámbitos.

De todo ello sería interesante realizar un trabajo y un estudio futuro para poder desarrollarlo y completar el trabajo que se ha seguido en esta tesina para desarrollar la Time Machine.

8.1. Conexión Real con la Smart-Home

Para el desarrollo de la tesina nos hemos centrado en la parte visual y en definir las funcionalidades que debía tener la Time Machine. Para el caso de estudio se han simulado los datos sin necesidad de conectarlo con la Smart-Home, únicamente mostrando una demo de lo que en realidad haría la aplicación. Uno de los trabajos futuros más importantes será el de conectar la Time Machine con la Smart-Home para que se integre con ella y tenga un funcionamiento completamente real.

De este modo se adaptaría la Time Machine para que a la hora de realizar un cambio o más en una configuración esta llamase al sistema de reconfiguraciones de la Smart-Home, aplicando los cambias al entorno real. Además cada vez que la Smart-Home realice un cambio deberá notificárselo a la Time Machine para que esta modifique también los datos que almacena y muestra al usuario.

Para ello en capítulos anteriores se ha explicado el funcionamiento completo de la Time Machine, así como la forma en la que se han simulado los datos y las interfaces que se han creado para su futura conexión a la Smart-Home real.

8.2. Gestión de Usuarios

Debido a que, como en cualquier aplicación, la Time Machine se puede utilizar por múltiples usuarios es de lógica que tenga una gestión de usuarios eficiente. Para el desarrollo de la tesina se ha hecho una simulación para el caso de estudio, pero en trabajos futuros sería interesante que se trabajase de forma más profunda en este apartado realizando una gestión eficiente de los usuarios y de las distintas opciones que puedan tener cada uno.

En el caso de estudio se ha presentado una propuesta de tipos de usuario y de permisos de cada uno pero en hipotética futura gestión de usuarios puede ir más allá e incluso crear un grupo de permisos y de usuarios configurables.

Esta gestión deberá asociar las configuraciones a los distintos usuarios de forma que dependiendo del usuario que esté accediendo a la aplicación pueda visualizar o modificar unas u otras configuraciones. Esto hará que una configuración además de a una fecha este asociada a un usuario (como hemos simulado para el caso de estudio). Además la Time Machine deberá de ser capaz de saber con qué usuario está tratando y saber que configuraciones y opciones debe mostrarles y cuáles no.

8.3. Otras aplicaciones de la Time Machine

Además de para la Smart-Home la Time Machine está pensado para que se pueda utilizar en cualquier otro ámbito de utilización que no sea una casa, ya sea un hotel, una oficina, un museo. Los ámbitos son diversos y cada uno puede tener sus peculiaridades.

Aunque la Time Machine se ha hecho de forma bastante genérica tanto a nivel de programación (está programada en Java y por tanto puede funcionar sobre cualquier plataforma) como a nivel de configuración (se puede cambiar el nombre de las habitaciones o

las configuraciones de una forma sencilla), sería interesante probarla en otros ámbitos y adaptarlas a otros ámbitos. Incluso hacer que fuese totalmente configurable a nivel de número de habitaciones etc.

Anexo A: Detalle de Implementación

Figura 68 – Alcance del capítulo 4.2.

En este apartado se va a explicar en detalle algunos aspectos de la implementación de la Time Machine. Para ello primero explicaremos los métodos de utilidad de la Time Machine así como la forma elegida para guardar los datos. Después se pasará a explicar en detalle los métodos relativos a las utilidades internas y la interfaz de conexión a los datos. Explicaremos también en detalle la estructura de la BD y como se ha implementado la interfaz de conexión a datos.

A.1. Utilidades internas

Para las distintas utilidades y operaciones de la Time Machine se ha diseñado una clase de utilidad que mediante una serie de métodos estáticos se realizan distintas funcionalidades útiles para el funcionamiento de la Time Machine.

Concretamente estos métodos se encargan de crear los objetos AggregateDemo que representan las configuraciones y almacenarlos en un vector mediante el método rellenaConfiguraciones o distintos métodos para el control de cambios y modificaciones (botones deshacer y rehacer) mediante los métodos getArrayModificaciones o getArrayModificacionesDeshechas. Además tenemos otros métodos que nos permite obtener la fecha actual (getFechaActual) o el número de modificaciones que hemos realizado (getModificaciones), entre otros.

Los métodos más importantes que se han diseñado para esta clase de utilidad son los siguientes:

Método	Parametros entrada	Parametros Salida	Descripción
rellenaConfiguraciones	Ninguno	void	Método que crea los gráficos de las configuraciones actuales de la casa.
addModificacion	Nombre(String)	void	Añade una modificación al Array donde guardamos las modificaciones realizadas por el usuario.
addModificacion	Ninguno	void	Incrementa el contador de modificaciones.
getModificaciones	Ninguno	int	Devuelve el contador de modificaciones.
setModificaciones	int	void	Setea el contador de modificaciones.
getArrayModificaciones	Ninguno	ArrayList<String>	Devuelve el array con las modificaciones realizadas por el usuario.
deshacerCambio	Ninguno	void	Mueve una modificación del array de modificaciones al de modificaciones deshechas.
rehacerCambio	Ninguno	Void	Mueve una modificación del array de modificaciones deshechas al de modificaciones.
getArrayModificaciones Deshechas	Ninguno	ArrayList<String>	Devuelve las modificaciones que se han deshecho.
getFechaActual	Ninguno	Date	Devuelve la fecha actual.
getConfigActual	Ninguno	Vector<Aggregate Demo>	Devuelve los objetos Aggregate Demo de la configuración actual.

Tabla 2 – Métodos de utilidad

A.2 Interfaz de conexión a los datos

Puesto que para el desarrollo de la tesis lo que se ha hecho es una simulación y no una conexión real a los datos lo que se ha hecho es crear una interfaz que hemos implementado para obtener los datos, tanto los estáticos (nombre de las habitaciones, opciones) como los dinámicos (fechas, configuraciones). De esta forma se independiza el desarrollo de esta tesis con la conexión al sistema real. Para desarrollos futuros de conexión a la Smart-Home o para adaptarlo a otros sistemas se deberá implementar de nuevo esta interfaz como veremos más adelante. En la figura 68 podemos ver una imagen de la implementación de dicha interfaz.

```
 8 public interface DatosTime {
 9
10     Vector<String> getHabitaciones();
11     Vector<Date> getFechas();
12     Hashtable<String, String> getOpciones();
13     boolean isFromComputer(Date fecha);
14     ArrayList<String[]> getDispositivos(Date fecha, String sala);
15     ArrayList<String[]> getCanales(Date fecha, String sala);
16     ArrayList<String[]> getServicios(Date fecha, String sala);
17     ArrayList<String[]> getServiciosForDisp(String disp);
18     void borrarConfig(Date fecha);
19     void reconfigurarHabitación(ArrayList<String> desactivar,
20             ArrayList<String> activar, Date fecha, String habitación);
21 }
```

Figura 69 – Interfaz DatosTime

La interfaz cuenta con los siguientes métodos que independientemente del sistema elegido para su implementación, la funcionalidad y los datos que se obtendrán serán los mismos.

Método	Entrada	Salida	Descripción
getDispositivos	Fecha (Date) Habitación(String)	ArrayList<String[]>	Este método obtiene los dispositivos activos en una habitación para la fecha indicada. Si la fecha es nula se obtienen los actuales. Devuelve un ArrayList de tuplas [nombre, imagen] de los dispositivos
getFechas	Ninguno	Vector<Date>	Obtiene todas las fechas de las que tenemos configuraciones guardadas que no pertenezcan a la configuración actual.

getHabitaciones	Ninguno	Vector<String>	Obtiene el nombre de las habitaciones de la casa.
getOpciones	Ninguno	Hashtable<String, String>	Obtiene todas las opciones de la Time Machine devolviéndolas en un Hashtable donde el key es el nombre de la opción y el valor la imagen correspondiente.
getServicios	Fecha (Date) Habitación(String)	ArrayList<String[]>	Este método obtiene los servicios activos en una habitación para la fecha indicada. Si la fecha es nula se obtienen los actuales. Devuelve un ArrayList de tuplas [nombre, imagen] de los servicios.
getCanales	Fecha (Date) Habitación(String)	ArrayList<String[]>	Este método obtiene los canales activos en una habitación para la fecha indicada. Si la fecha es nula se obtienen los actuales. Devuelve un ArrayList de tuplas [nombreNodo2, nombreNodo1] de los canales.
getServiciosForDisp	Dispositivo(String)	ArrayList<String[]>	Este método devuelve los servicios conectados a un determinado dispositivo, identificado por su nombre.
isFromComputer	Fecha(Date)	Boolean	Dada una fecha devuelve si la configuración ha sido creada por el sistema o no.
borrarConfig	Fecha (Date)	void	Reconfigura la casa con las opciones elegidas por el usuario.
reconfiguraHabitación	Disp. a desactivar (ArrayList<String>) Disp. A activar (ArrayList<String>) Fecha (Date) Habitación(String)	Void	Reconfigura una sola habitación con las opciones elegidas por el usuario.
addConfiguracion	Fecha (Date) Habitación(String) Usuario(String)	void	Se encarga de introducir una nueva configuración en el sistema de almacenamiento de la Time Machine.

	Dispositivos(Array)
	Servicios(Array)
	Canales(Array)

Tabla 3 – Métodos para la obtención de datos

Estos métodos nos serán útiles por ejemplo a la hora de crear la componente AgreggateDemo ya que sabiendo la fecha y la habitación podemos obtener los dispositivos y servicios para representar los gráficos de la componente y obtener los canales para representar los ejes entre dichos nodos.

Además a la hora de reconfigurar una habitación o la casa entera mediante estos métodos nos permitirán aplicar los cambios correspondientes. En nuestro caso los guardaremos en BD como veremos más adelante pero en el caso de estar conectada a la Smart-Home estos métodos serán los encargos de hacer de conexión con el sistema de la Smart-Home y de aplicar las reconfiguraciones a la vivienda.

A.3. Estructura de la BD

A continuación se van a explicar la estructura que se ha creado para guardar los datos de las configuraciones en una Base de Datos. Concretamente se ha utilizado una BD en MySQL ya que se trata de un sistema de código libre y sencillo de utilizar.

Evidentemente la estructura de nuestra BD gira en torno al objetivo de guardar las configuraciones. Por ello la tabla más importante y sobre la que giran todas las demás es la de CONFIGURACION. Como ya hemos indicado antes una configuración está relacionada con una habitación, con una fecha y con un usuario, por ello cada configuración estará relacionada con un usuario de la tabla USUARIO, y una habitación de la tabla HABITACION. En el caso de la fecha se tratará de un campo más de la tabla CONFIGURACION. La fecha que guardaremos constará de día, mes y año, como es lógico, pero también de hora, minutos y segundos, puesto que en un mismo día pueden guardarse más de una configuración y necesitamos distinguir entre ellas.

Puesto que la fecha que guardamos es la fecha en la que creamos o modificamos por última vez la configuración, a la hora de abrir la Time Machine no tenemos forma de saber cuál es la configuración actual, ya que no tiene por qué coincidir con la fecha actual, y menos sabiendo que para la fecha tenemos en cuenta hora, minutos y segundos. Por ello hemos creado un campo dentro de la tabla CONFIGURACION llamado ACTUAL en la que nos indica si se trata de la configuración actual.

Además una configuración está formada, como ya hemos dicho antes, de dispositivos, servicios y canales. Para ello en la Base de Datos se ha creado una tabla para almacenar cada uno de los datos. Por tanto tendremos una tabla llamada DISPOSITIVO, otra SERVICIO, y otra CANALES. Una configuración está formada por uno o más dispositivos, por uno o más servicios y por uno más canales.

Un canal, por tanto, relacionará un DISPOSITIVO con un SERVICIO o un SERVICIO con otro SERVICIO, ya que se puede dar el caso de que los servicios estén también conectados.

El gráfico que representa la BD que hemos utilizado está representado por el siguiente Modelo de Tablas:

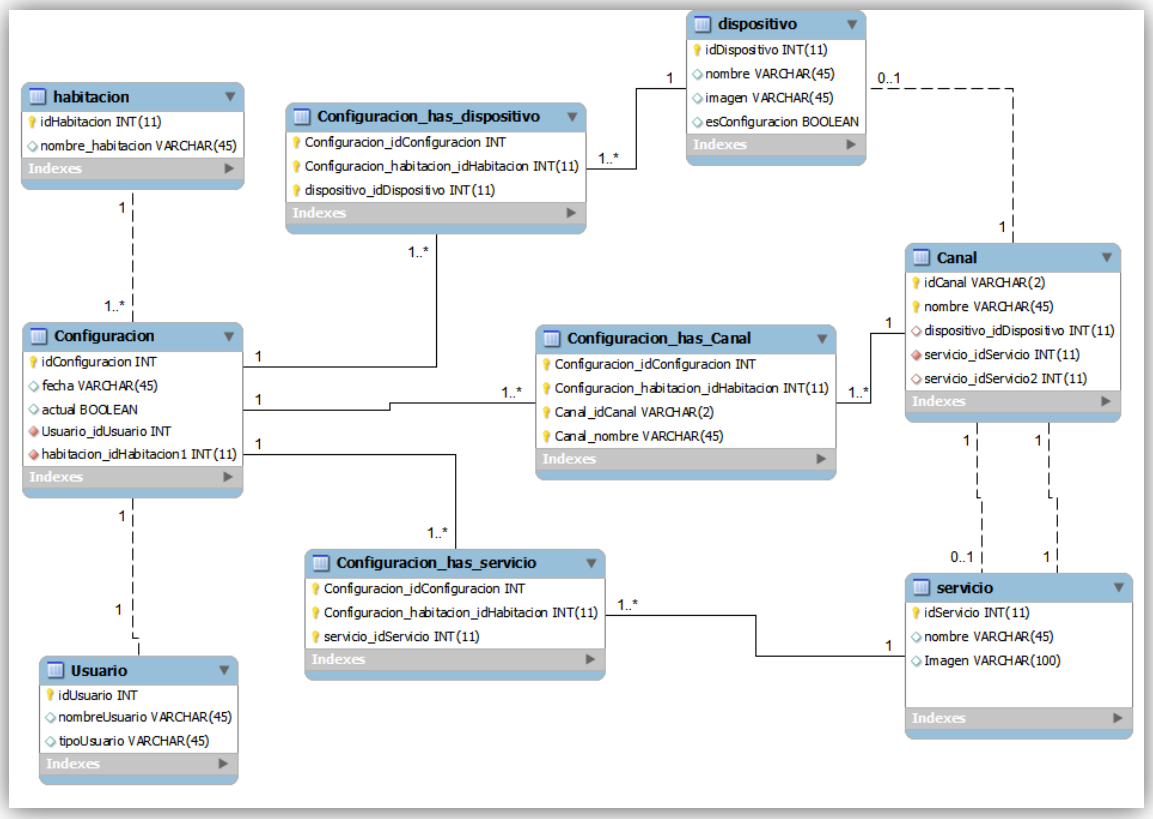

Figura 70- Estructura de la BD.

Además a continuación se muestra una descripción de las tablas y los atributos de las tablas de la BD. Se han excluido de esta tabla la descripción para las tablas de BD que hace de nexo entre tablas como Configuracion_has_servicio. Así como también las Primary Keys y las Foreign Keys, debido a que carecen de interés para el significado de la tabla.

Tabla	Atributos	Descripción
Configuración	Fecha Actual	Representa la configuración de una habitación en un instante de tiempo.
Habitación	Nombre_habitacion	Habitaciones de las que dispone la casa.
Usuario	NombreUsuario tipoUsuario	Usuario que tienen acceso a la Time-Machine.
Dispositivo	Nombre Imagen esConfiguracion	Dispositivos disponibles en el sistema de la Smart-Home. Mediante el atributo esConfiguración indicamos si queremos mostrárselo al usuario como posible configuración a aplicar.
Servicio	Nombre imagen	Servicios disponibles en la Smart-Home.
Canal	nombre	Canales de comunicación entre los servicios y los dispositivos de la Smart-Home o entre dos servicios de la Smart-Home

Tabla 4 – Descripción de las tablas de BD

A.4. Implementación de los métodos de la Interfaz DatosTime

Una vez hemos visto la funcionalidad de los métodos de la interfaz y la estructura de la BD que hemos creado, vamos a pasar a explicar cómo se han implementado estos métodos para el desarrollo de esta tesina. La mayoría de estos métodos lo único que hacen es crear una conexión entre la Time Machine y la Base de Datos.

- **getDispositivos/ getServicios**

Como podemos ver en la figura siguiente para conseguir los dispositivos y servicios de una configuración bastará con realizar una select a la BD buscando por fecha y por habitación. En caso de que la fecha sea nula, como ya hemos dicho antes, significará que queremos

obtener la configuración actual y por tanto en lugar de por fecha buscaremos por actual = 1. El código copiado es el correspondiente al método getDispositivos pero es equivalente para el método getServicios cambiando las tablas de “Dispositivos” por las de “Servicios”.

```
211 if (fecha!= null){
212     String cadenaFecha = formato.format(fecha);
213     rs = st.executeQuery("select d.nombre, d.imagen from dispositivo d, configuracion c," +
214             " configuracion_has_dispositivo cd, habitacion h where c.fecha = '"+cadenaFecha+"'" +
215             " and c.idConfiguracion = cd.Configuracion_idConfiguracion and " +
216             "cd.dispositivo_idDispositivo = d.idDispositivo and c.habitacion_idHabitacion = h.idHabitacion" +
217             " and h.nombre_habitacion = '"+habitacion+"'");
218 }else{
219     rs = st.executeQuery("select d.nombre, d.imagen from dispositivo d, configuracion c," +
220             " configuracion_has_dispositivo cd, habitacion h where c.actual = 1" +
221             " and c.idConfiguracion = cd.Configuracion_idConfiguracion and" +
222             " cd.dispositivo_idDispositivo = d.idDispositivo and c.habitacion_idHabitacion = h.idHabitacion" +
223             " and h.nombre_habitacion = '"+habitacion+"'");
224
```

Figura 71 – Select getDispositivos/getServicios

- **getCanales**

Para obtener los canales realizamos una consulta muy parecida a la de servicios y dispositivos pero en lugar del nombre y la imagen debemos coger el nombre del servicio y del dispositivo que conecta. Tendremos que repetir dos veces la consulta, una para hallar las relaciones entre dispositivos y servicios y otra para las relaciones entre servicios.

```
284 if (fecha!= null){
285     String cadenaFecha = formato.format(fecha);
286     rs = st.executeQuery("select d.nombre,s.nombre from habitacion h, configuracion cf, canal c," +
287             " configuracion_has_canal cc, dispositivo d, servicio s " +
288             "where  cf.fecha = '"+cadenaFecha+"' and cf.idConfiguracion = cc.configuracion_idConfiguracion "
289             "and cc.canal_idCanal = c.idCanal and c.dispositivo_idDispositivo = d.idDispositivo " +
290             "and c.servicio_idServicio = s.idServicio and cf.habitacion_idHabitacion = h.idHabitacion " +
291             "and h.nombre_habitacion = '"+habitacion+"'");
292 }else{
293     rs = st.executeQuery("select d.nombre,s.nombre from habitacion h, configuracion cf, canal c," +
294             " configuracion_has_canal cc, dispositivo d, servicio s " +
295             "where  cf.actual = 1 and cf.idConfiguracion = cc.configuracion_idConfiguracion " +
296             "and cc.canal_idCanal = c.idCanal and c.dispositivo_idDispositivo = d.idDispositivo" +
297             " and c.servicio_idServicio = s.idServicio and cf.habitacion_idHabitacion = h.idHabitacion " +
298             "and h.nombre_habitacion = '"+habitacion+"'");
299
300 }
```

Figura 72 – Método getCanales

- **getFechas**

Para obtener las fechas del histórico lo que hacemos es realizar una consulta simple seleccionando simplemente las que no tenga en el flag actual a 1. Es decir las que no sean la configuración actual.

```
160  ResultSet rs = st.executeQuery("select distinct fecha from configuracion" +
161          " where actual = 0 order by fecha");
```

Figura 73 – Método getFechas

- **getHabitaciones**

Para obtener las habitaciones seleccionamos todas las habitaciones que estén almacenadas en la BD en la tabla habitación.

```
189  ResultSet rs = st.executeQuery("select nombre_habitacion from habitacion");
```

Figura 74 – Método getFechas

- **getOpciones**

Para obtener las opciones de configuración que le ofrecemos al usuario hacemos una consulta a la tabla dispositivos seleccionando las que tengan el flag esConfiguracion a 1.

```
183  ResultSet rs = st.executeQuery("select nombre, imagen from dispositivo" +
184          " where esConfiguracion = 1");
```

Figura 75 – Método getOpciones

- **isFromComputer**

Para saber si una configuración es una configuración creada por la máquina, basta con hacer una consulta y obtener el nombre de usuario. Si el nombre es “Computer” se trata de un configuración creada por el sistema. En cualquier otro caso se trata de una configuración de usuario.

```
326 ResultSet rs = st.executeQuery("select nombreUsuario from usuario u, configuracion c" +
327         " where c.fecha = '"+cadenaFecha+"' and u.idUsuario = c.Usuario_idUsuario ");
```

Figura 76 – Método isFromComputer

- **getServiciosForDisp**

Este metodo nos devolverá un array de tuplas <nombre, imagen> de los servicios a los que está conectado determinado dispositivo. Mediante esta select obtendremos dicho resultado:

```
380 rs = st.executeQuery("select s.nombre, s.imagen from servicio s, canal c, dispositivo d " +
381         "where s.idServicio = c.servicio_idServicio and c.dispositivo_idDispositivo = d.idDispositivo" +
382         " and d.nombre = '"+nombre+"'");
```

Figura 77- Método getServiciosForDisp

- **reconfiguraHabitación**

Para reconfigurar una habitación necesitaremos la fecha, el nombre de la habitación y los elementos a activar y desactivar. Este método realmente es el que conectará con la Smart-Home para que se apliquen los cambios. En la implementación actual no tiene sentido ya que lo que hacemos nosotros es guardar una nueva configuración. En un entorno real de actuación cuando el usuario realice cambios se deberá guardar una entrada en el histórico de configuraciones de la Time Machine y llamar a este método para que reconfigure la Smart-Home.

- **borrarConfig**

Para borrar una configuración lo que haremos será dada una fecha, borrar primero de las tablas configuracion_has_canal, configuracion_has_dispositivo, configuracion_has_servicio aquellas filas cuyo configuracion_idConfiguracion coincida con el de la fecha que le pasamos. Por último borraremos las filas de la tabla configuración.

```
452 rs = st.executeQuery("delete  from configuracion_has_canal  where configuracion_idconfiguracion in " +
453         "(select idConfiguracion from configuracion where fecha = '"+cadenaFecha+"');");
454 rs = st.executeQuery("delete  from configuracion_has_dispositivo  where  configuracion_idconfiguracion in" +
455         " (select idConfiguracion from configuracion where fecha = '"+cadenaFecha+"');");
456 rs = st.executeQuery("delete  from configuracion_has_servicio  where configuracion_idconfiguracion in" +
457         " (select idConfiguracion from configuracion where fecha = '"+cadenaFecha+"');");
458 rs = st.executeQuery("delete  from configuracion  where fecha = '"+cadenaFecha+"');");
```

Figura 78 – Método borrarConfig

Bibliografía

[1] C. Cetina, P. Giner, J. Fons, V. Pelechano. Autonomic Computing through Reuse of Variability Models at Runtime: The Case of Smart Homes. IEEE Computer. vol. 42, no. 10, pp. 37-43, Oct. 2009.

[2] J. D. McGregor. Software Product Lines. Chair of Software Engineering Vol. 3, No. 3, March-April 2004.

[3] Douglas C. Schmidt. Model- Driven Engineering. IEEE Computer Society February 2006.

[4] R. France, B. Rumpe. Model-driven Development of Complex Software: A Research Roadmap. Future of Software Engineering(FOSE'07), 2007.

[5] Colin Atkinson, Thomas Kühne. Model-Driven Development: A Metamodeling Foundation. IEEE SOFTWARE, 2003.

[6] Bran Selic. The Pragmatics of Model-Driven Development. IEEE Computer Society, 2003.

[7] Jeffrey O. Kephart, David M.Chess. The Vision of Autonomic Computing. IEEE Computer Society, enero 2003

[8] MARKUS C. HUEBSCHER and JULIE A. McCANN. A survey of Autonomic Computing—Degrees, Models, and Applications. ACM Computing Surveys, Vol. 40, No. 3, Article 7, Publication date: August 2008.

[9] Roy Sterritt. Autonomic computing. Innovations Syst Softw Eng (2005) 1: 79–88.

[10] Carlos Cetina. Achieving Autonomic Computing through the Use of Variability Models at Run-time. Enero, 2010.

[11] S. Ibiza. Desarrollo de una Herramienta para el Diseño de Reconfiguraciones Seguras en Sistemas de Computación Autónoma. Septiembre de 2010.

[12] Prefuse – Information Visualization toolkit. http://Prefuse.org/.

[13] P. Schobbens, J. C. Trigaux, P. Heymans and Y. Bontemps. Generic semantics of feature diagrams. Comput. Netw., 51(2):456–479, 2007.

[14]D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated reasoning on feature models. LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, 3520:491–503, 2005.

[15]A. Agrawal, T. Levendovszky, J. Sprinkle, F. Shi, and G. Karsai. Generative programming via graph transformations in the model-driven architecture. En, In OOPSLA 2002 Workshop in Generative Techniques in the context of Model Driven Architecture, 2002.

[16]J. Coplien, D. Hoffman, and D. Weiss. Commonality and variability in software engineering. Software, IEEE, 15(6):37–45, Nov/Dec 1998.

[17]F. Roos-Frantz, D. Benavides, A. Ruiz-Cortés. Feature Model to Orthogonal Variability Model Transformations. A First Step. Actas de los Talleres de las Jornadas de Ing. del Software y BBDD, Vol. 3, No. 2, 2009.

[18]C. Cetina, P. Giner, J. Fons, V. Pelechano. Using Variability Models for Developing Self-con_guring Pervasive Systems. Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 2, No. 5, 2008.

[19]Birgit Korherr and Beate List. A UML 2 Profile for Variability Models and their Dependency to Business Processes. 2002.

[20]S. Hallsteinsen, M. Hinchey, Sooyong Park, and K. Schmid. Dynamic software product lines. Computer, 41(4):93–95, April 2008.

[21]D. F. Bantz, C. Bisdikian, D. Challener, J. P. Karidis, S. Mastrianni, A. Mohindra, D. G. Shea, and M. Vanover. Autonomic personal computing. IBM Syst. J., 42(1):165–176, 2003.

[22]T. Dinkelaker, R. Mitschke, K. Fetzer, M. Mezini. A Dynamic Software Product Line Approach using Aspect Models at Runtime. First Workshop on Composition and Variability'10 Rennes, France collocated with AOSD'2010.

[23]G. Banavar and A. Bernstein. Software infrastructure and design challenges for ubiquitous computing applications. Commun. ACM, 45(12):92–96, 2002.

[24]J. P. Sousa and D. Garlan. Aura: An architectural framework for user mobility in ubiquitous computing environments. En In Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture, pages 29–43. Kluwer Academic Publishers, 2002.

[25]Jeffrey Heer, Stuart K. Card y James a. Landay (2005). "Prefuse: un kit de herramientas de visualización de información interactiva". En: actas de la Conferencia SIGCHI sobre factores humanos en sistemas informáticos: 421-430, Portland, Oregón, Estados Unidos: ACM.