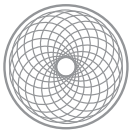
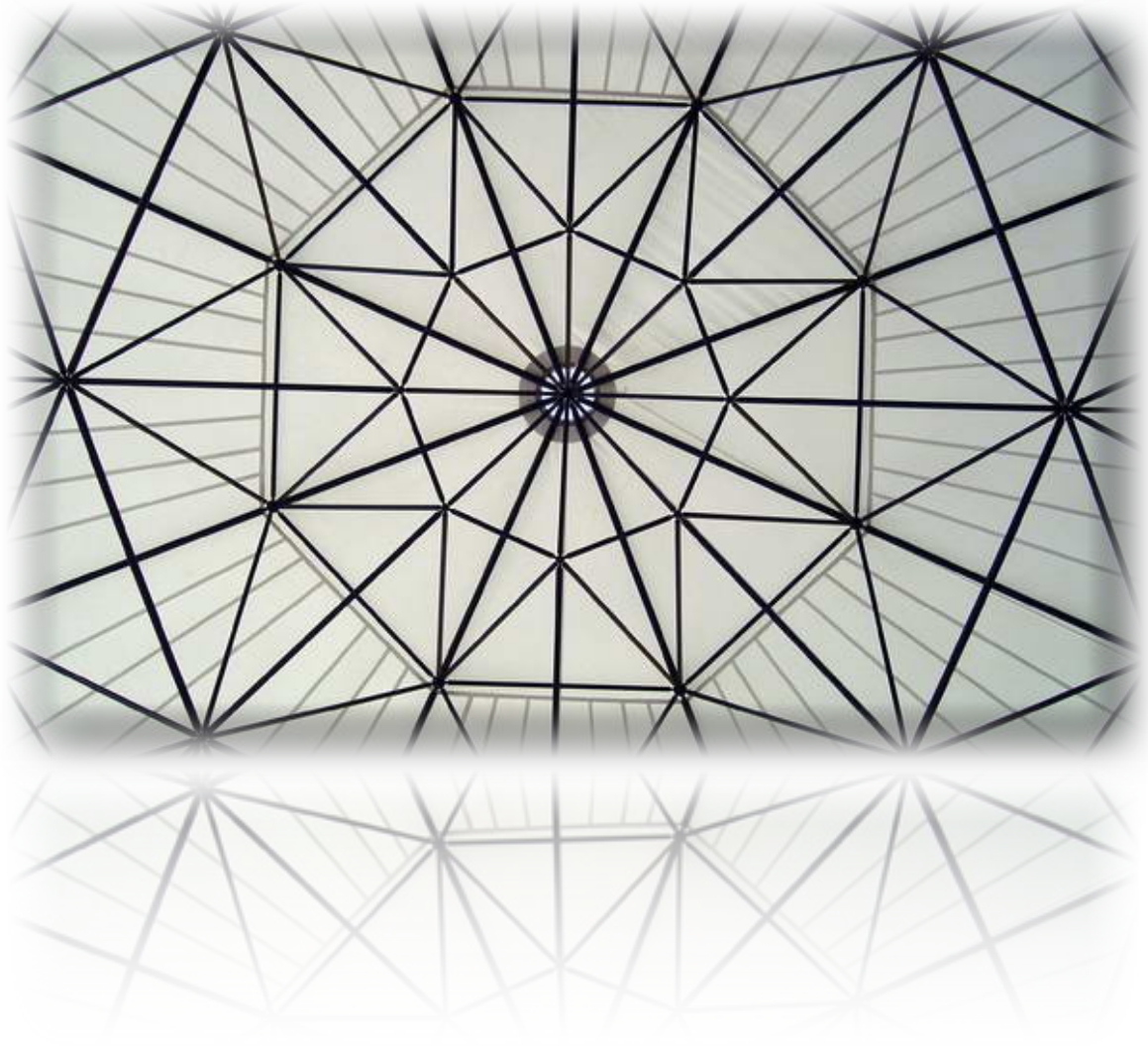


Empresa

Memoria

Monitorización de estructuras y gestión de edificios
Building Management

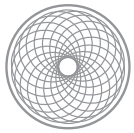


Ingeniería Técnica de Informática en Sistemas

Consultor: Marc Domingo

Alumno: Antonio Martin Moreno

11 de junio de 2012



Indice

I. Introducción

1. Contexto

Actualmente nos encontramos con la necesidad de controlar el estado de los edificios para poder evitar posibles problemas causados por el deterioro y accidentes: incendios, inundaciones, desastres naturales, seismos, etc. Con el fin de evitar desastres económicos, se necesita poder actuar de una forma rápida y ordenada como consecuencia de la destrucción del edificio o su deterioro natural, o bien para evitar tragedias de índole humano de las personas que se encuentran dentro o en los alrededores del edificio, si este se desplomara o sufriera algún daño en su estructura.

2. Objetivo

El objetivo del proyecto es realizar un sistema de monitorizacion de estructuras y gestión de edificios para controlar en todo momento el estado de los mismos. Se controlará cualquier aspecto que pueda afectar al edificio como es la temperatura, la humedad, la oscilación del edificio; ademas de otros aspectos que ayuden a que el edificio se mantenga seguro y en perfecto estado como alarmas de seguridad, luminosidad, etc.

3. El sistema

Se desarrollará un sistema que constará de un conjunto de sensores instalados en los edificios, un servidor para almacenar los datos de forma centralizada, y un conjunto de sistemas móviles donde se instalará la aplicación, a modo cliente, que consulta y gestiona los datos recibidos desde el servidor central.

El conjunto de sensores instalados en los edificios captaran los siguientes aspectos: el movimiento del edificio (vibraciones por minuto), la temperatura, la presión de las paredes, la luminosidad, la humedad, el humo, etc. Estos sensores realizaran un informe con la información precisa captada en cada momento que se enviara al servidor central. Los sensores tendrán la capacidad de gestionar algunos aspectos anteriormente comentados.

Desde la aplicación móvil se podrá consultar en todo momento estado del edificio y se podrán tomar las medidas necesarias y oportunas para gestionarlo, como sería enviar un técnico al edificio para reparar una avería, o gestionar la temperatura, alumbrado y seguridad del edificio de forma telemática por medio de la aplicación móvil.

Nota.- Debido a que no se disponen de sensores para poder implementar el sistema completo, no entrará dentro del alcance del proyecto el conocimiento del uso de la tecnología de comunicación entre los sensores y el servidor central, ni su implementación. Simplemente nos centraremos en la realización de la aplicación web y la aplicación móvil, y su interacción. Daremos por hecho que existen tales sensores, y que la aplicación web se comunica con ellos. Está ultima gestionará la información que se supone ha recibido de los sensores para que la aplicación móvil sea funcional.

4. Funcionalidades del sistema

La aplicación móvil realizará las siguientes funciones:

- Listado de edificios: la aplicación muestra una lista con los edificios monitorizados en el sistema.
- Consultar estado de un edificio: la aplicación muestra la información en detalle con del estado edificio.
- Búsqueda o filtrado de edificios: la aplicación permite buscar un edificio en concreto para ser consultado.
- Gestionar aspectos del edificio: la aplicación permite gestionar las alarmas y servicios del sistema.
- Comunicación de incidencia o avería de un edificio: la aplicación permite la comunicación con el personal responsable del mantenimiento y gestión del edificio.

El servidor central expondrá los siguientes servicios para dotar a la aplicación móvil de la funcionalidad anterior requerida:

- Listado de edificios: el servidor provee la lista de edificios monitorizados por el sistema
- Consultar estado de un edificio: el servidor provee la información en detalle con el estado edificio.
- Búsqueda o filtrado de edificios: el servidor provee la información de un edificio en concreto para ser consultado.
- Gestionar aspectos del edificio: el servidor ejecuta las acciones sobre las alarmas y servicios del sistema.

Se supondrá la existencia de los sensores instalados en los edificios que se comunican con el servidor, por lo que esta funcionalidad no se detallará. Queda fuera del objetivo del proyecto.

5. Planificación del proyecto

El calendario vendrá condicionado por las fechas de los dos principales hitos requeridos por la asignatura. A saber:

- Entrega de PEC2: 09 de abril del 2012
- Entrega de PEC3: 21 de mayo de 2012
- Entrega final: 12 de junio de 2012

Principalmente se dividen las tareas en bloques entre las fechas de los hitos.

La composición de las tareas en el calendario resulta de la siguiente forma:

Desde la 12 de marzo hasta el 09 de abril

- Diseño técnico de la aplicación web en el servidor.
- Diseño técnico de la aplicación móvil.
- Desarrollo e implementación de la aplicación web.

Desde el 10 de abril hasta el 21 de mayo

- Desarrollo e implementación de la interfaz de la aplicación móvil.
- Instalación, configuración y pruebas unitarias de la aplicación web en servidor.
- Instalación, configuración y pruebas unitarias de la aplicación móvil.
- Pruebas integración y uso de la aplicación móvil + aplicación web.
- Entrega del proyecto finalizado: aplicación móvil + aplicación web.

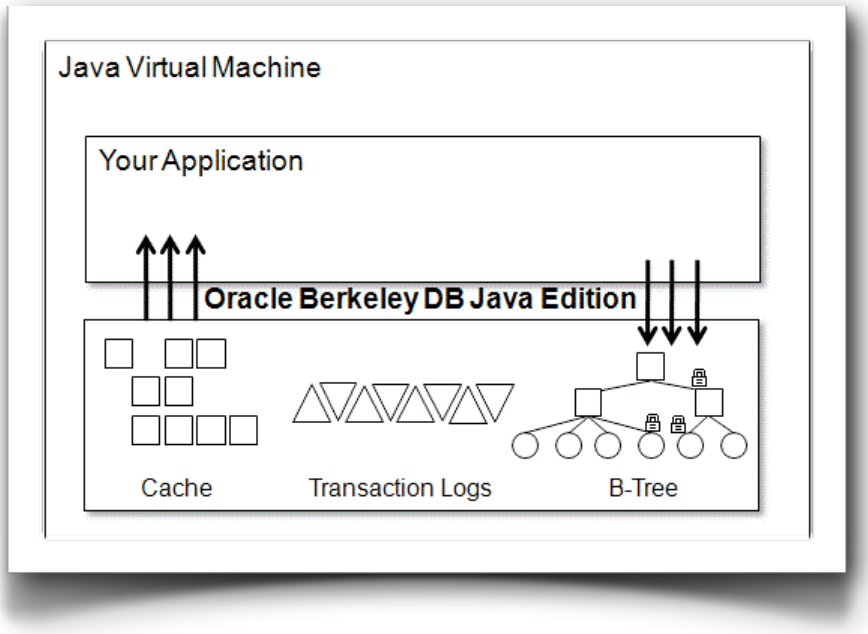
Desde el 22 de mayo hasta el 12 de junio de 2012

- Memoria.
- Presentación proyecto en formato video.

6. Tecnología del sistema: Aplicación web

El servidor central, será un servidor que albergará una aplicación web que sirva la información a la aplicación móvil tal y como se ha expuesto en el apartado de funcionalidades del sistema. Este servidor será un servidor Tomcat 7.0 que cumple con la especificación 3.0 de Java Servlets para poder publicar servicios. Al igual que la aplicación móvil, la aplicación web se desarrollará con Java, en su versión Java Standard Edition v1.6.

El datastore utilizado en la parte del servidor es la solución Oracle Berkeley-java DB de libre distribución. Se trata de una base de datos incrustada orientada a persistir objetos (no es relacional), siendo tal, que para su despliegue simplemente necesitaremos incluir el JAR que la constituye en nuestro classpath. Se utilizará la versión 5.0.34. No tiene dependencias de otras librerías, siendo su integración limpia en nuestro proyecto. El rendimiento es bueno, y su funcionalidad excelente, al ser un sistema ORM (Object Relational Mapping) que gestiona nuestras clases persistentes. Tiene su propío API de consulta, que como era de esperar y siendo un ORM, se consulta por el tipo de clase y el ID del elemento de la clase que se quiere gestionar. (http://www.oracle.com/technetwork/products/berkeleydb/overview/index.html)



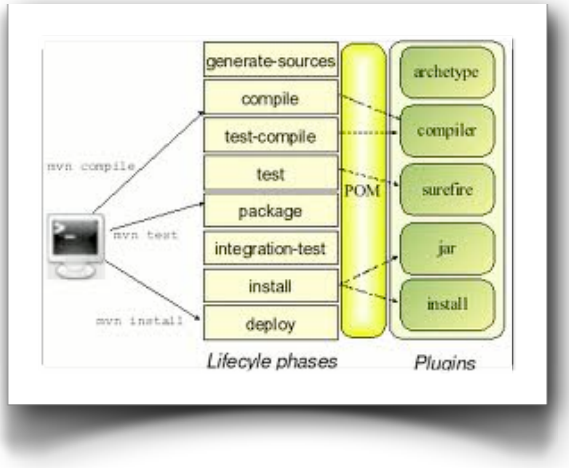
Oracle Berkeley DB Java Edition

Para la carga inicial del datastore se utilizará XStream de código libre. Esta librería simplemente serializa objetos a formato XML y viceversa. La versión que se usará es al 1.3. No tiene dependencias de otras librerías, siendo su integración limpia en nuestro proyecto. El rendimiento es bueno, y su funcionalidad excelente. Simplemente necesitaremos incluir el JAR que la constituye en nuestro classpath. (http://xstream.codehaus.org/)

Para la tarea de logging de la aplicación se ha utilizado log4j en su versión 1.2.15. Al igual que en los casos anteriores se ha buscado que no tuviera dependencias y que su integración fuera limpia, aportando un rendimiento bueno y una funcionalidad excelente. (http://logging.apache.org/log4j/1.2/)

Se utilizará el IDE Eclipse para desarrollo web con integración para servidores Tomcat para poder desplegar la aplicación y realizar pruebas. En este caso se usara la versión Eclipse Java EE IDE for Web Developers. (http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigosr2)

Finalmente comentar que no se ha utilizado la herramienta de gestión de proyectos Maven, puesto que el proyecto no tenia las dimensiones suficientes y hubiera complicado mas el desarrollo debido a su instalación y mantenimiento. En proyectos de mas envergadura, Maven no solo es deseable, sino que es indispensable. Generar entregables: JAR, WAR y EAR's, desplegar proyectos en servidores, gestionar dependencias de librerías, e incluso pasar una batería de pruebas es posible con Maven. La contrapartida es el tiempo que se tiene que utilizar para que Maven pueda gestionarlas. (http://maven.apache.org/)



Arquitectura Maven

7. Tecnología del sistema: Aplicación móvil

La aplicación se desarrollará para el sistema operativo Android, en su versión mas reciente 4.0.2., usando las API's disponibles para dicha versión (v.15). A su vez, el lenguaje utilizado para usar las API's y desarrollar la aplicación móvil será Java Standard Edition v1.6.

La aplicación se desarrollará para dispositivos Android con pantalla WVGA800, densidad LCD 240, max VM application heap size 48, memoria ram del dispositivo 512. Estas características en si comportan un riesgo debido a la diversificación inherente de terminales con sistema operativo Android.

Se utilizará el IDE eclipse para desarrollo de aplicaciones móviles con emulador de terminal móvil para poder instalar la aplicación y poder realizar pruebas. En este caso se usara la versión Eclipse for RCP and RAP Developers. (http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigosr2)

8. Entregables

La entrega final constará de un fichero del tipo apk con la aplicación Android, otro fichero del tipo war con la aplicación web:

- buildingmanagement.apk
- buildingmanagement.war

9. Instalación

La instalación del sistema final constará del despliegue de la aplicación web sobre el servidor Tomcat antes mencionado (no importa la plataforma: windows, linux o unix), y de la aplicación móvil instalada en un teléfono móvil compatible, o en su defecto en un emulador Android que haga las veces de terminal móvil.

10. Riesgos

Los riegos principales que encontramos son:

- Riesgo en el uso de tecnologías desconocidas: desconocimiento en el desarrollo de aplicaciones móviles en Android con el API suministrado.
- Riesgo de Integración de tecnologías de aplicaciones móviles con aplicaciones web. Esto es consecuencia del punto anterior. Si se desconoce el desarrollo de aplicaciones móviles en Android, también se desconoce su integración con desarrollos web.
- Riesgo en la instalación de la aplicación móvil en el terminal final. Que las características con las que se han desarrollado la aplicación no sean compatibles con las características del terminal final donde se va a instalar la aplicación.
- Riesgo de que no funcione la aplicación móvil como se espera en el terminal final debido a particularidades del terminal.

11. Contingencias

Debido a los riesgos inherentes en el desarrollo de la aplicación, en caso de que no se pueda continuar con el desarrollo por cualquier problema ocurrido, se optará por continuar con el desarrollo aplicando la solución mas factible y que guíe a la finalización del proyecto. En ese caso, se documentará tal circunstancia con los cambios aplicados a lo largo del seguimiento del proyecto.

II. Diseño técnico

1. Introducción

Se hará un breve repaso de los casos de uso implementados por el sistema y por los diagramas de clase y despliegue del sistema.

2. Casos de uso

Los casos de la aplicación son los siguientes:

- Buscar alarmas: se podrá realizar un búsqueda de las alarmas por nombre, población, provincia o código postal.
- Consultar alarma: se seleccionara una alarma de la lista para poder consultar en detalle la alarma: su nombre, ubicación, temperatura, luminosidad y estado.
- Gestionar alarma: se podrá gestionar la alarma como cambiar su estado, cambiar la temperatura y la iluminación.
- Gestionar comentarios: se podrán crear, actualizar y eliminar comentarios para cada alarma de tal forma que el usuario pueda hacer un seguimiento de la alarma.
- Visualización de localización de alarma: se podrá ver la ubicación de la alarma en un mapa a fin de saber la localización y tener una idea clara de donde está ubicada.
- Comunicación con el instalador: se podrá enviar un correo al responsable de mantenimiento de la alarma.

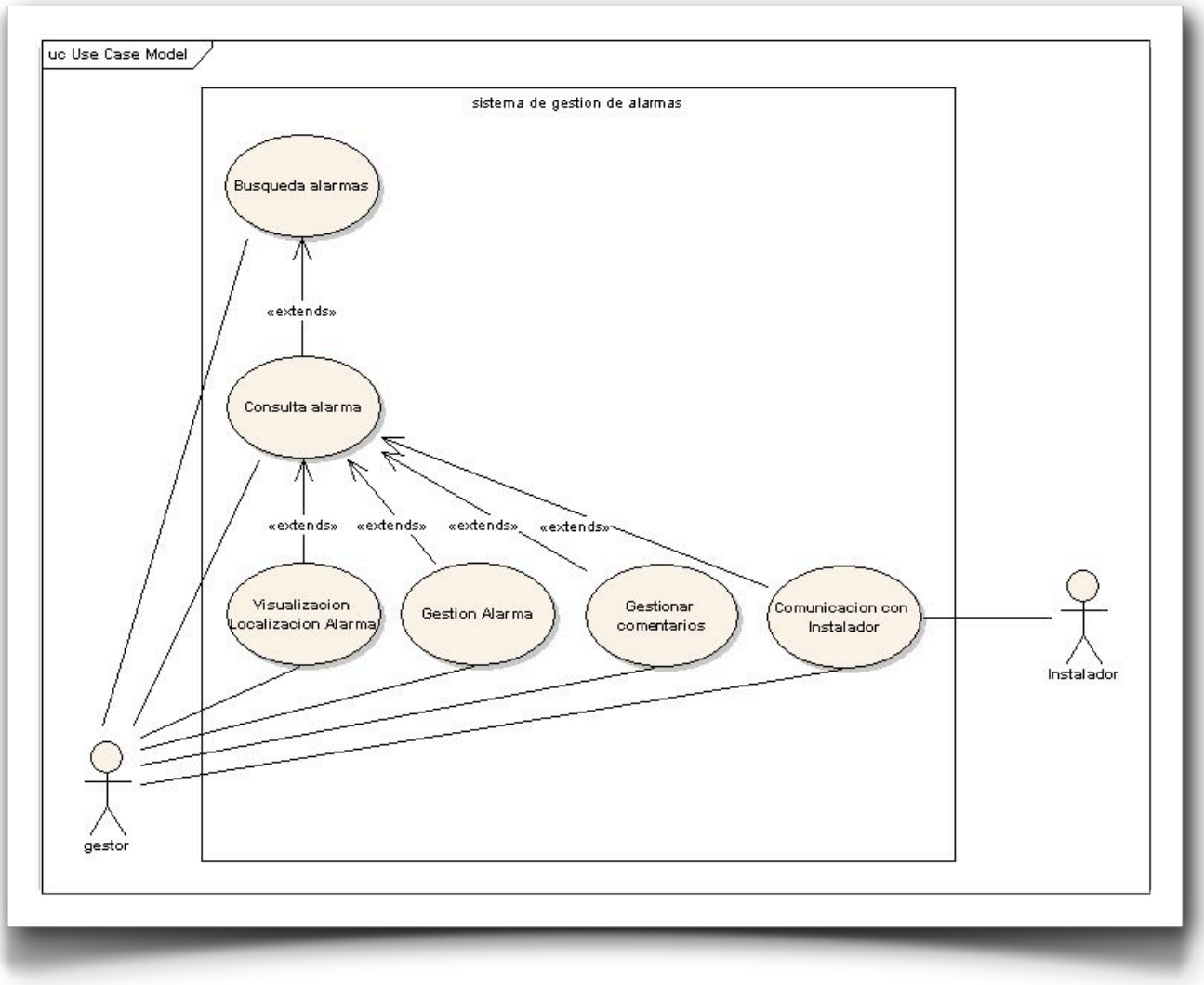
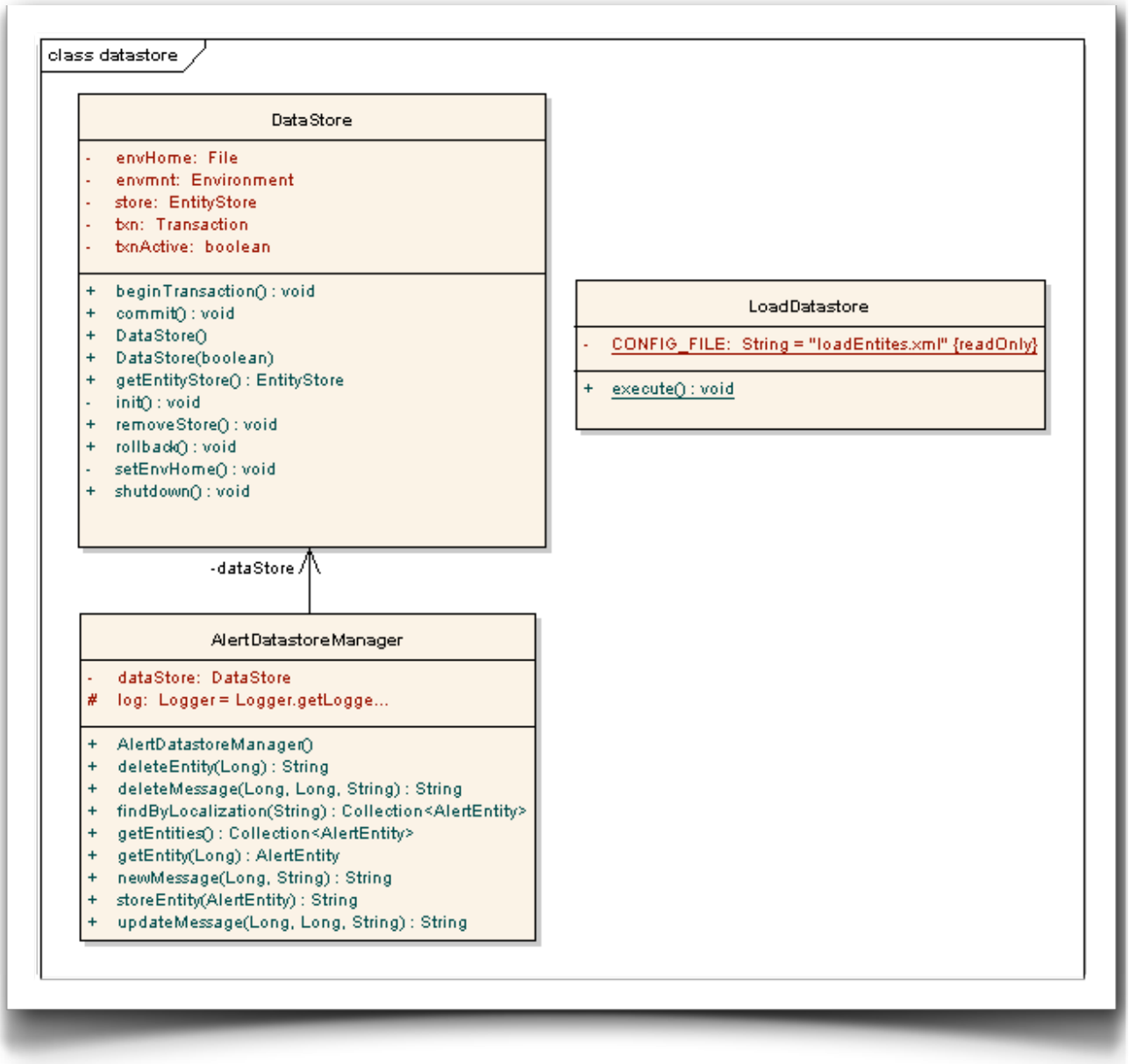


diagrama de casos de uso

3. Diagramas de clases: Aplicación web

Se ha diseñado la aplicación web que actúa como la parte servidora del sistema y que gestiona y almacena los datos principalmente. Se ha divido la aplicación en tres capas que se corresponden con los siguientes paquetes:

- Paquete uoc.ftc.amm.datastore. Capa que gestiona el datastore o almacén de datos. El datastore utilizado para almacenar datos es la solución Oracle Berkeley-java DB como se comentó anteriormente. Las clases implementadas en esta capa gestionaran la creación y el acceso al datastore, lo que viene siendo el DAO (Data Access Object) del sistema. Haciendo un repaso por clases y su funcionalidad:
  - uoc.ftc.amm.datastore.DataStore: clase encarga de administrar el acceso al almacén de datos de la aplicación. Crea el datastore y gestiona las transacciones.
  - uoc.ftc.amm.datastore.AlertDataStoreManager: clase encargada de gestionar el acceso al datastore por medio de la aplicación. Es la interfaz con la aplicación que expone los servicios que puede realizar contra el datastore.
  - uoc.ftc.amm.datastore.LoadDataStore: clase encargada de inicializar el datastore para que contenga los datos iniciales con los que se va a trabajar.

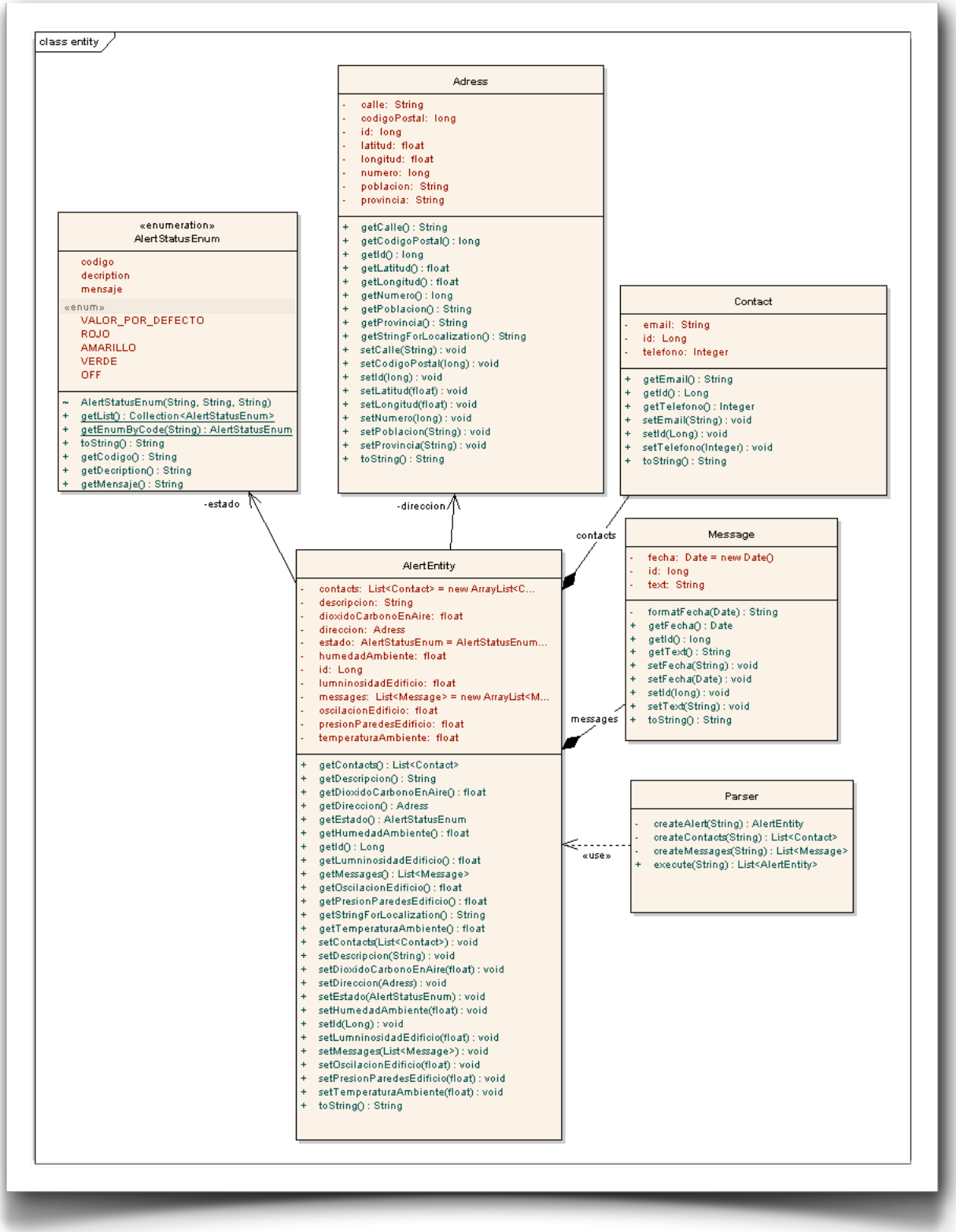


Package uoc.ftc.amm.datastore

- Paquete uoc.ftc.amm.entity. Capa que gestiona las entidades que contienen la información. Principalmente se han modelado las clases necesarias para contener la información básica de las alarmas, como son el estado, la dirección, la temperatura, los comentarios, geolocalización, etc. Haciendo un repaso por clases y su funcionalidad:
    - uoc.ftc.amm.entity.AlertEntity: clase encarga de almacenar la información sobre las alarmas: temperatura del edificio, luminosidad del edificio, dirección del edificio, estado del edificio, etc.
    - uoc.ftc.amm.entity.Adress: clase encarga de almacenar la dirección del edificio.
    - uoc.ftc.amm.entity.Contact: clase encargada de almacenar los contactos de la alarma del edificio.
    - uoc.ftc.amm.entity.AlertStatusEnum: clase encargada de almacenar el estado del edificio. Son cuatro estados: apagado, problemas, revisar y ok.
    - uoc.ftc.amm.entity.Message: clase encarga de almacenar la información sobre las mensajes de las alarmas del edificio para llegar un control del edificio.
    - uoc.ftc.amm.entity.Parser: clase encarga de transformar las tramas de texto en objetos.

Las clases modeladas en esta capa son las responsables de generar la estructura de la información que va a viajar a la aplicación cliente. Generan una trama con la información que almacenan utilizando:

- separadores “|” para separar la de información de cada valor del atributo
- separador “[“ y “]” para separar la información cuando son varios elementos tipo lista
- sepador “,” y “$” para separar entidades.



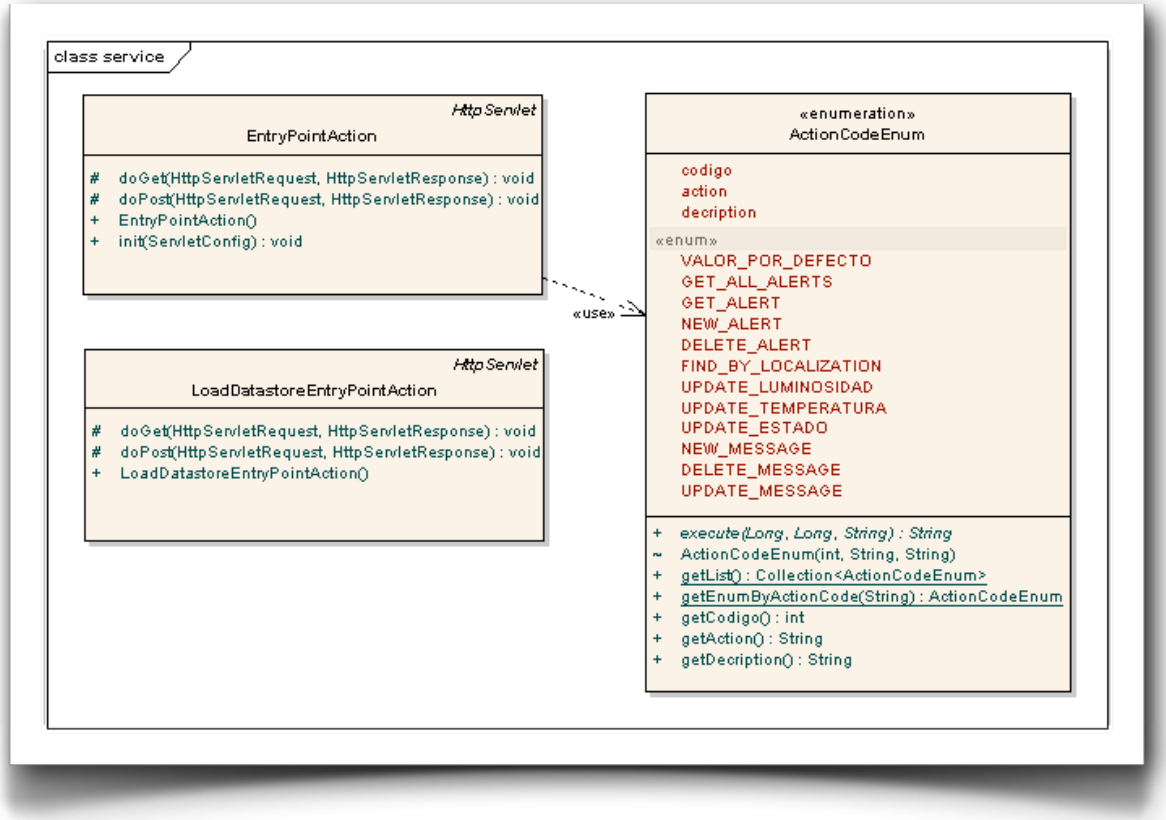
Package uoc.ftc.amm.entity

- Paquete uoc.ftc.amm.service. Capa que controla las peticiones de acceso al sistema para servir los datos. Se han generado un par de servlets: uno para inicializar el sistema y otro para gestionar las peticiones de datos del sistema. Haciendo un repaso por clases y su funcionalidad:

  - uoc.ftc.amm.service.EntryPointAction: clase encargada de controlar los accesos al sistema. Dirige las peticiones a la acción adecuada.

  - uoc.ftc.amm.service.ActionCodeEnum: clase encargada de gestionar las acciones del sistema. Es la encargada de llamar al DAO para realizar la gestion de la información con el datastore.

  - uoc.ftc.amm.service.LoadDatastoreEntryPointAction: clase encargada de inicializar el datastore con los datos



Package uoc.ftc.amm.entity

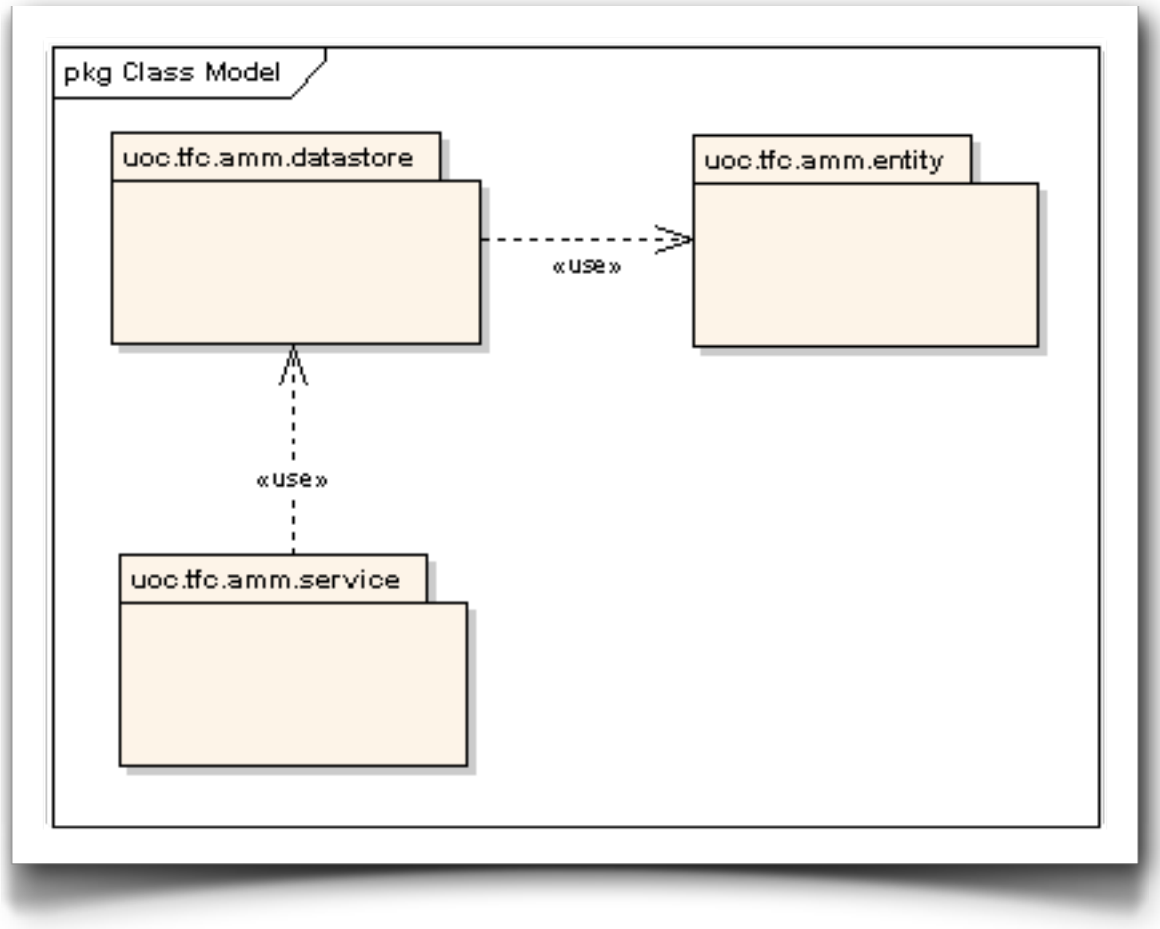


diagrama de dependencias entre paquetes

Para finalizar indicar que se ha realizado una pequeña capa de presentación para testar los servicios, con los cual nos queda un sistema implementado de forma que sigue el paradigma MVC (Model-View-Controller).

index.jsp aplicación web, acceso a los servicios

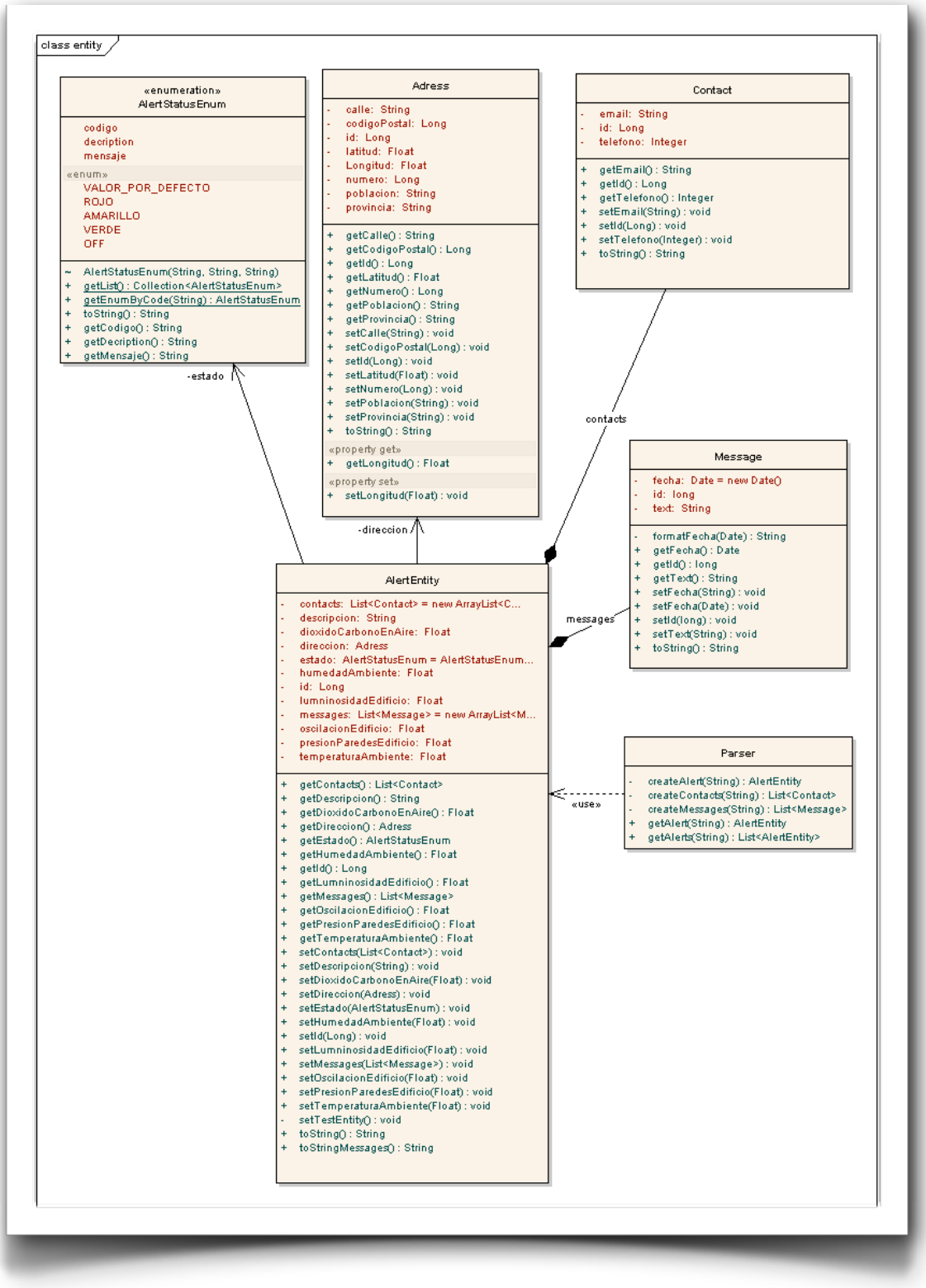
4. Diagramas de clases: aplicación móvil

Se ha diseñado la aplicación móvil que actúa como la parte cliente del sistema y que realiza las peticiones al servidor para saber como está el sistema. Se ha divido la aplicación en tres capas, que se corresponden con los siguientes paquetes:

- Paquete uoc.ftc.amm.entity. Capa que gestiona las entidades que contienen la información. Se corresponde con el modelo de la aplicación. Este paquete es similar al empleado en la parte del servidor, como cabria de esperar. Haciendo un repaso por clases y su funcionalidad:
  - uoc.ftc.amm.entity.AlertEntity: clase encarga de almacenar la información sobre las alarmas: temperatura del edificio, luminosidad del edificio, dirección del edificio, estado del edificio, etc.
  - uoc.ftc.amm.entity.Adress: clase encarga de almacenar la dirección del edificio.
  - uoc.ftc.amm.entity.Contact: clase encargada de almacenar los contactos de la alarma del edificio.
  - uoc.ftc.amm.entity.AlertStatusEnum: clase encargada de almacenar el estado del edificio. Son cuatro estados: apagado, problemas, revisar y ok.
  - uoc.ftc.amm.entity.Message: clase encarga de almacenar la información sobre las mensajes de las alarmas del edificio para llegar un control del edificio.
  - uoc.ftc.amm.entity.Parser: clase encarga de transformar las tramas de texto en objetos.

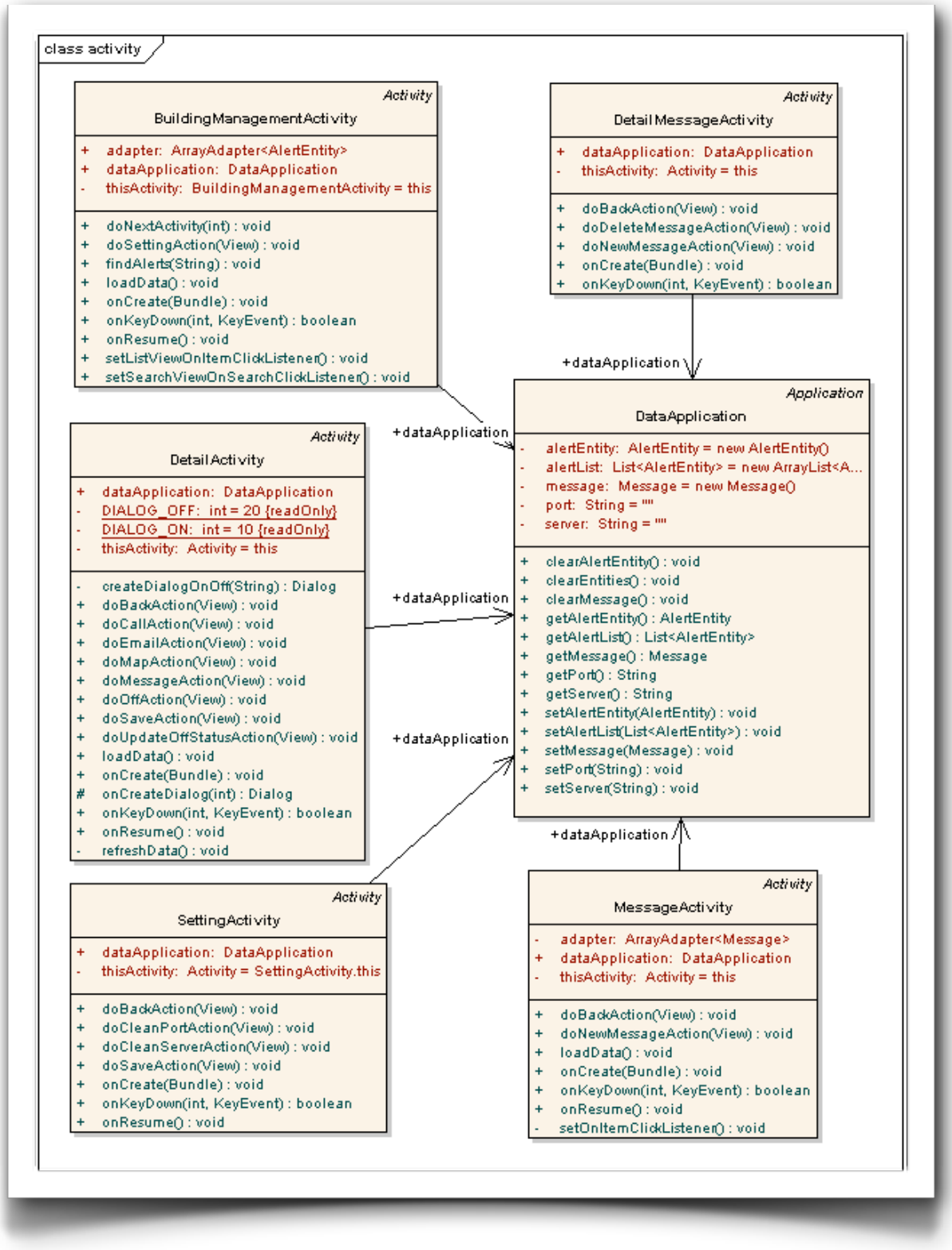
Las clases modeladas en esta capa son las responsables de generar la estructura de la información que va a viajar a la aplicación cliente. Generan una trama con la información que almacenan utilizando:

- separadores “|” para separar la de información de cada valor del atributo
- separador “[“ y “]” para separar la información cuando son varios elementos tipo lista
- sepador “,” y “$” para separar entidades.



package uoc.ftc.amm.entity

- Paquete uoc.ftc.amm.activity. Capa que gestiona el interfaz de usuario. Estas clases con las encargadas de generar la parte gráfica de la aplicación, la vista. Ademas actúan a modo de controladores de las peticiones del usuario. Haciendo un repaso por clases y su funcionalidad:
  - uoc.ftc.amm.entity.BuildingManagementActivity: actividad principal de la aplicación. Es la clase encargada de cargar la primera pantalla de la aplicación. Muestra una lista con las alarmas instaladas. Como todas las actividades, se apoyará en fichero de configuración para crear la vista o pantalla. En este caso será el main.xml.
  - uoc.ftc.amm.entity.DetailActivity: Es la clase encargada de mostrar el detalle de la alarma seleccionada. Se apoyará en fichero de configuración detail_view.xml para crear la vista.
  - uoc.ftc.amm.entity.MessageActivity: Es la clase encargada de mostrar los mensajes de la alarma seleccionada. Se apoyará en fichero de configuración message_view.xml para crear la vista.
  - uoc.ftc.amm.entity.DetailMessageActivity: Es la clase encargada de mostrar el detalle del mensaje seleccionado de la alarma seleccionada. Se apoyará en fichero de configuración detail_message_view.xml para crear la vista.
  - uoc.ftc.amm.entity.SettingActivity: Es la clase encargada de gestionar la configuración de la aplicación. En este caso se debe introducir la IP y el puerto del servidor HTTP al que nos vamos a conectar. Se apoyará en fichero de configuración settings_view.xml para crear la vista.
  - uoc.ftc.amm.entity.DataApplication: Es la clase encargada de gestionar los objetos que se comparten en las actividades de la aplicación, es decir, gestiona el estado de la aplicación. No tiene parte visual.



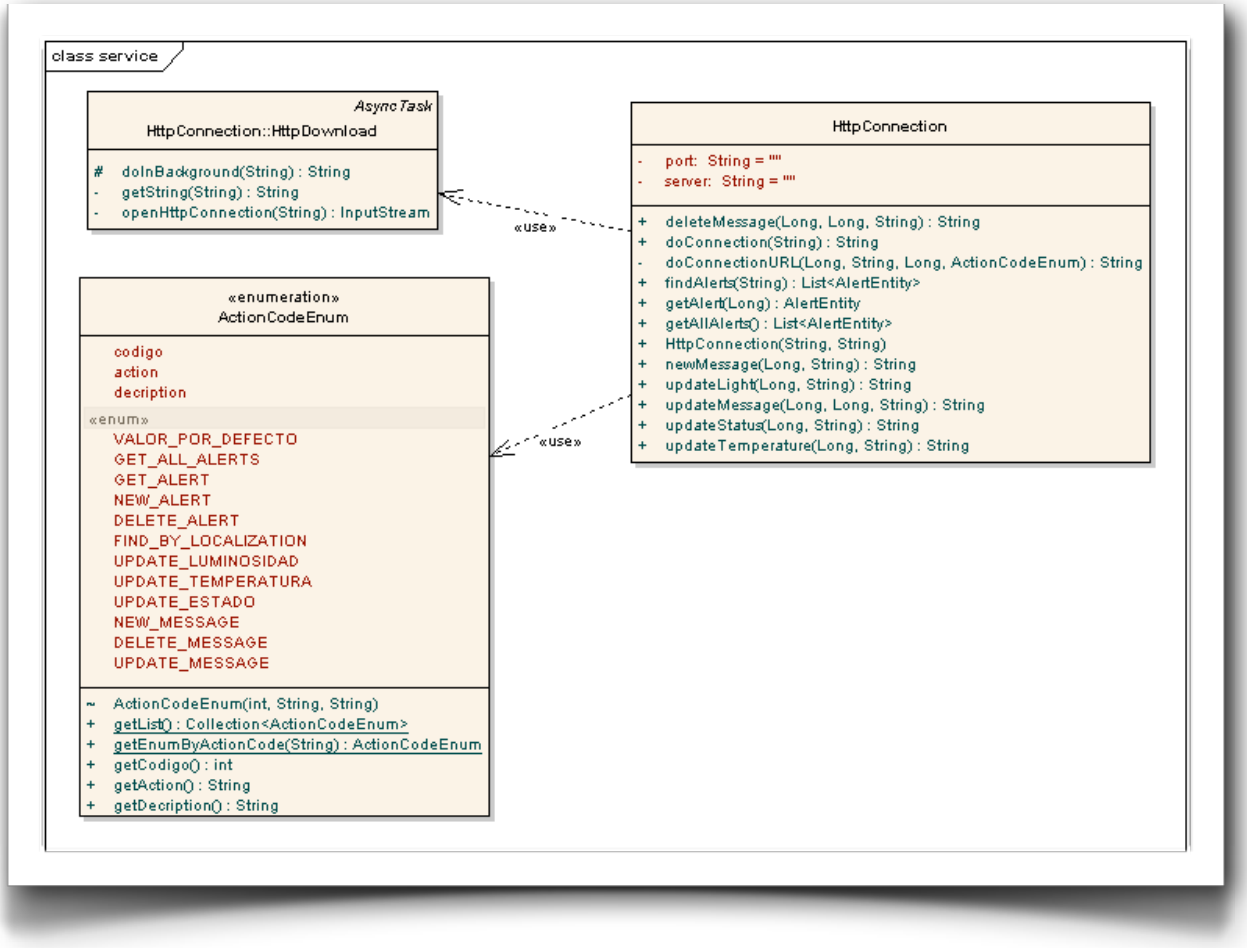
package uoc.ftc.amm.activity

- Paquete uoc.ftc.amm.service. Capa que gestiona la comunicación con el servidor. Se ha creado una clase que encapsula la comunicación con el servidor para abstraer de dicha lógica al resto del sistema. Haciendo un repaso por clases y su funcionalidad:

  - uoc.ftc.amm.service.HttpConnection: clase encargada de gestionar las peticiones hacia el servidor HTTP.

  - uoc.ftc.amm.service.HttpConnection.HttpDownload: clase encargada de realizar la conexión hacia el servidor HTTP.

  - uoc.ftc.amm.service.ActionCodeEnum: clase que contiene los códigos usados en la queryString que se usa para llamar al servidor. Se corresponden con los códigos que se usan en la parte servidora para identificar las acciones a realizar.



package uoc.ftc.amm.service

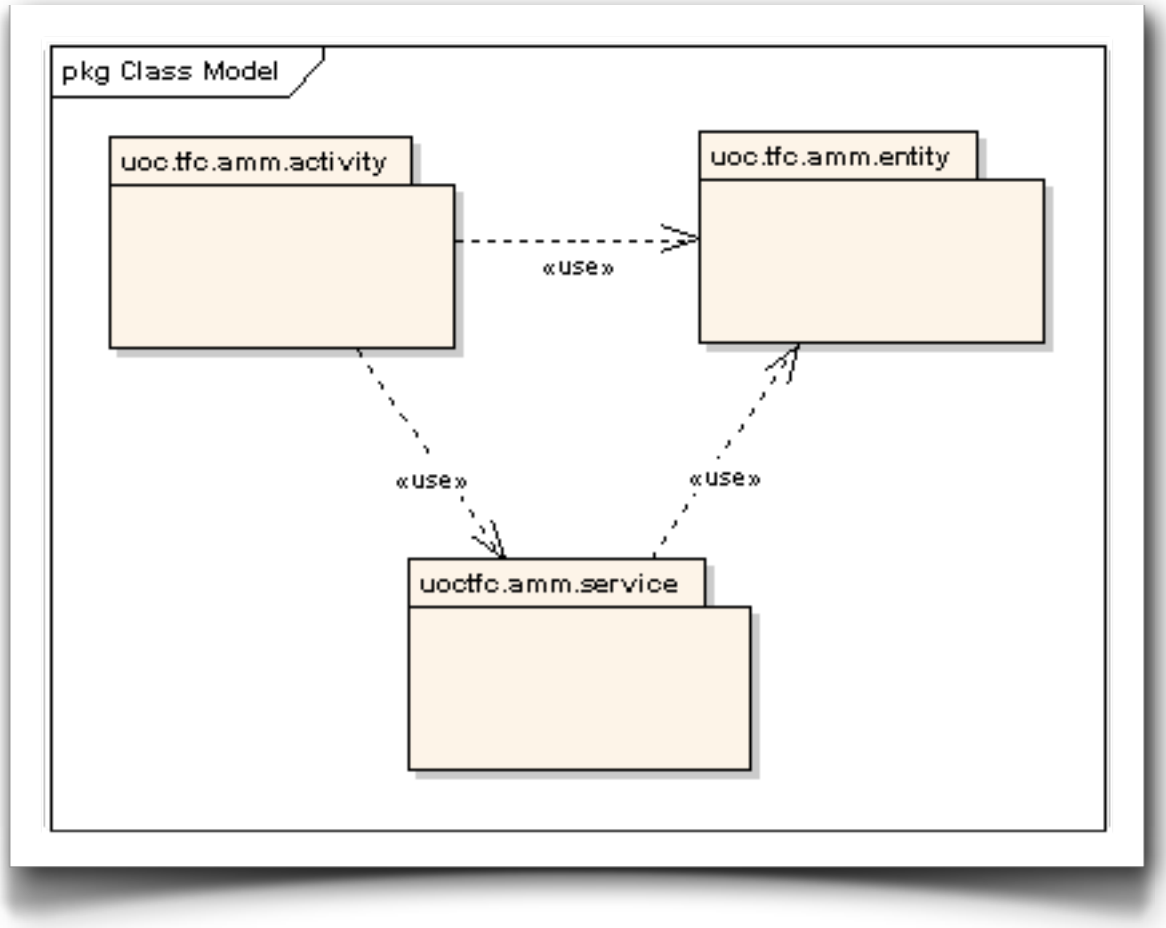


diagrama de dependencias entre paquetes

5. Diagrama de despliegue

El despliegue del sistema final quedara de la siguiente forma:

- Una aplicación móvil cliente instalada en un teléfono con SO android.
- Una aplicación web instalada en un servidor tomcat.

Los entregables en este caso serán un archivo apk y un archivo war respectivamente.

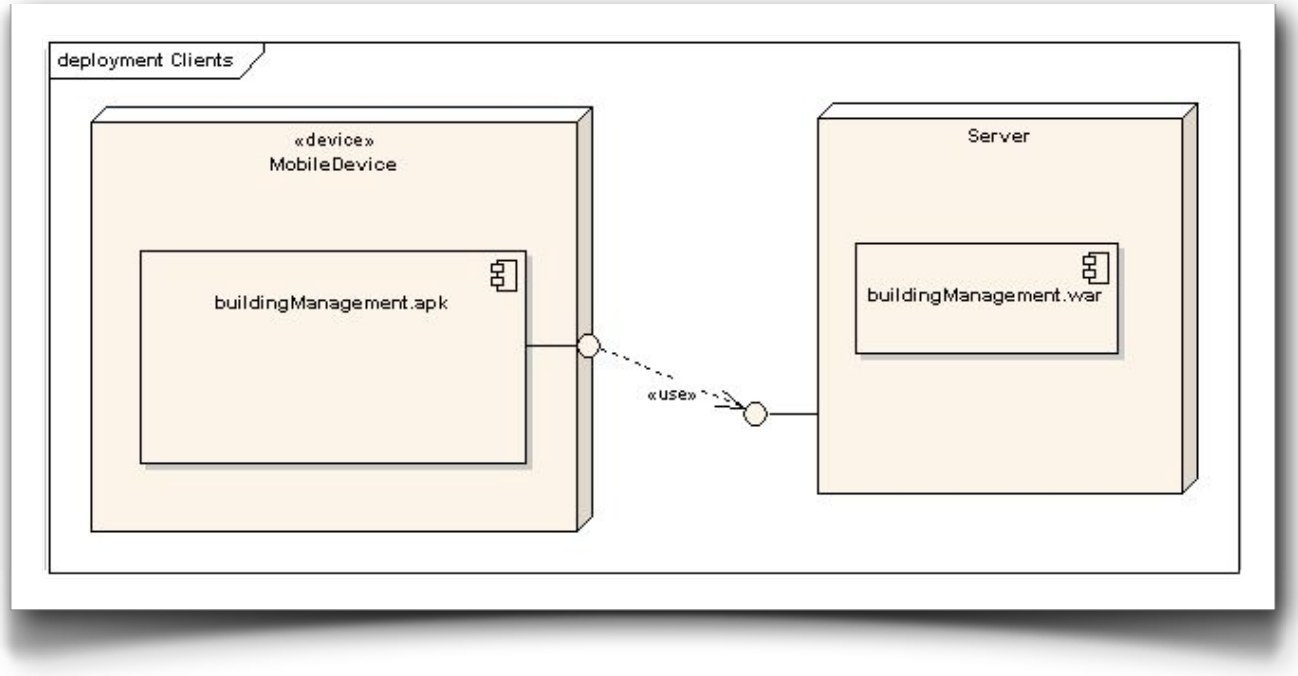


diagrama de despliegue

A tener en cuenta, en la parte del servidor se debe indicar un sitio donde ubicar el datastore que almacena los datos de la aplicación. Para ello se debe de indicar en el arranque del servidor el siguiente VM argument:

-Ddatastore="path donde almacenar el datastore"

por ejemplo:

-Ddatastore="/Users/antmarmo/Documents/workspace/tfc_war"

En caso de no indicar este parámetro la aplicación web no funcionará correctamente.

III. Instalación

1. Instalación de las aplicaciones

Primeramente se deberán instalar los entregables, tanto en el servidor TOMCAT v7 como en el teléfono móvil inteligente la versión de android: v4.0.2.

La instalación de la aplicación web se supone sencilla en un servidor web tomcat. En el servidor se debe crear un usuario con permisos para instalar la aplicación con rol de administrador, y seguidamente se puede instalar la aplicación usando la consola de administración de tomcat . Se debe tener en cuenta que en la configuración de arranque de la aplicación se debe indicar el VM argument -Ddatastore = “path”. Donde el path es la ruta donde se ubicará el almacén de datos de la aplicación. En caso de no configurar este parámetro la aplicación no funcionará correctamente. El servidor deberá tener permisos de lectura y escritura en esta ubicación, en caso contrario la aplicación no funcionará.

La instalación de la aplicación móvil puede hacerse incluyendo la aplicación en la tarjeta de móvil, seleccionando el fichero desde un explorador de archivos del móvil e indicar que se desea instalar. Se deberán seguir todos los pasos oportunos, permitiendo el acceso a los recursos del teléfono, como acceso a internet, acceso a agenda, etc. Se supone que el móvil ya tiene configurado la cuenta de correo electrónico por defecto con la cual se enviarán emails al instalador.

2. Inicialización dataStore en el servidor web

Una vez instalado el servidor web se debe inicializar el datastore para que contenga los datos de los edificios que se van a gestionar. Para ello basta con arrancar el servidor y en la pagina de inicio de la aplicación web, se debe pulsar al enlace de inicialización de base de datos, al final de la lista de enlaces. Los demás enlaces están pensados para hacer pruebas de comunicación con el servidor. No se deberían de utilizar.

Pagina de inicio de la aplicación:

http://localhost:8080/buildingManagement/index.jsp

index.jsp aplicación web, acceso a los servicios

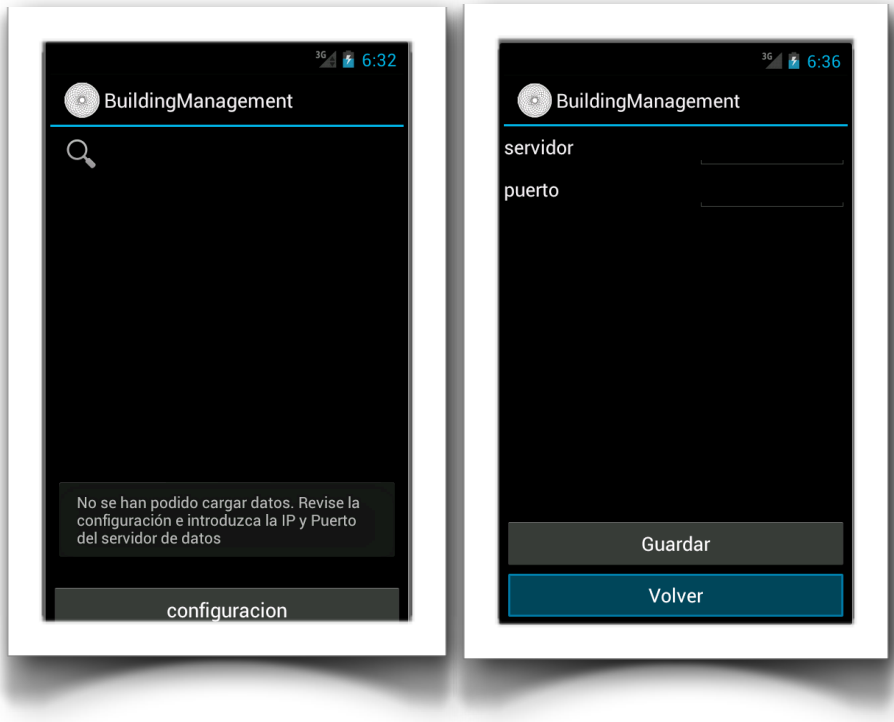
URL para inicializar el datastore:

http://localhost:8080/buildingManagement/LoadDatastoreEntryPointAction

3. Aplicación móvil

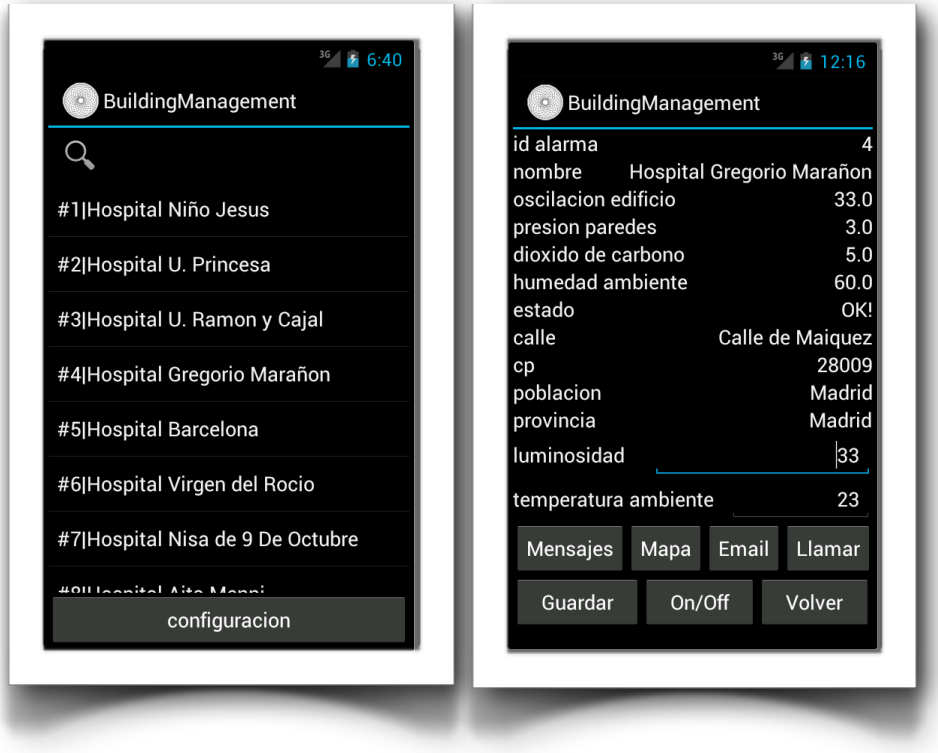
Una vez instalada la aplicación móvil, aparecerá en la lista de aplicaciones instaladas como BuildingManagement:

Al entrar aparecerá un mensaje indicando que se debe configurar la aplicación indicando la IP y puerto del servidor al que se debe conectar la aplicación. Pulsando en el botón configuración se accederá a la ventana donde se podrá indicar esta información.



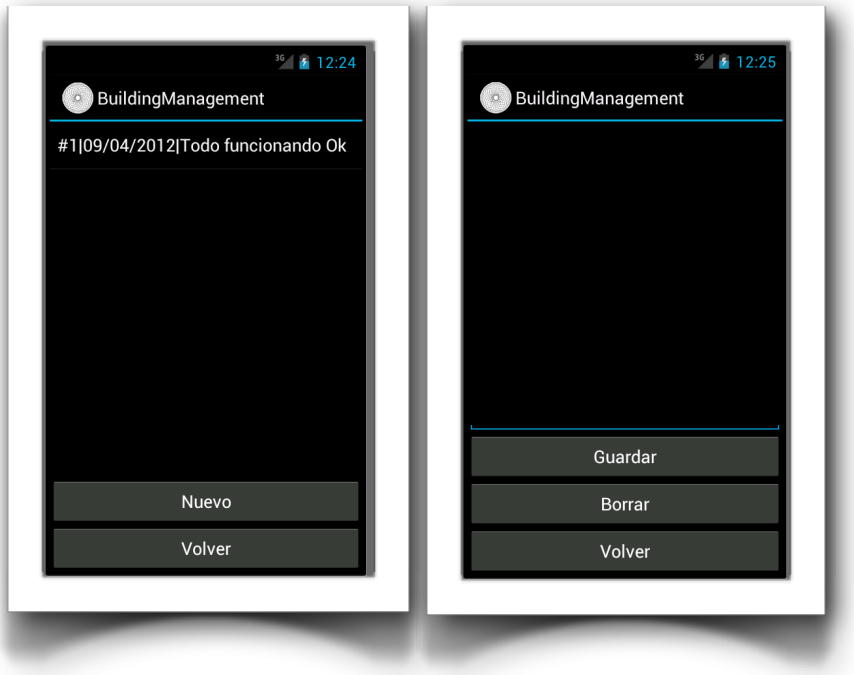
Se rellenan los campos servidor y puerto, y seguidamente se pulsa guardar. La aplicación intentará conectarse al servidor para probar la conexión. En el caso de que no se pueda conectar se indicará con un mensaje emergente. Si todo va bien, también se indicará con un mensaje y ya podremos proceder a volver a la ventana principal.

Una vez configurados los parámetros correctamente se pulsará volver para aparecer en la ventana principal, y se verán los las alarmas. Se seleccionará un registro y la aplicación nos llevará al detalle del edificio seleccionado donde está ubicado el sensor.

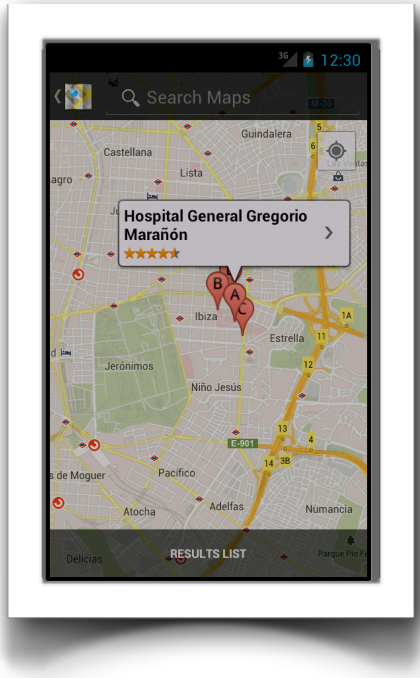


Desde aquí se puede ver el estado del sensor instalado en el edificio y las mediciones realizadas. Las acciones que se pueden realizar, a parte de la consulta, están indicadas en los botones al final de la ventana. A saber:

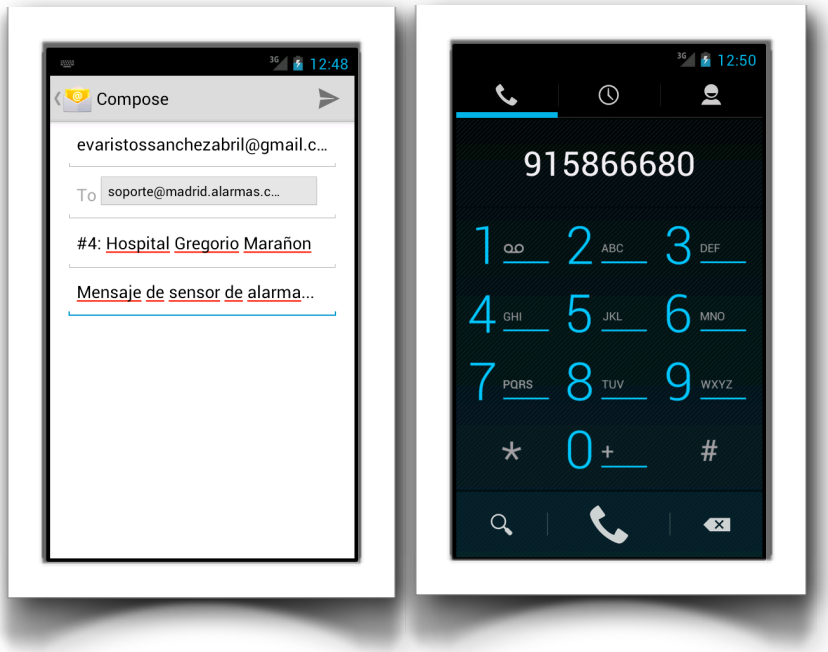
- Mensajes: se puede acceder al histórico de mensajes sobre el sensor actual, donde se va haciendo el seguimiento del edificio, y enviar nuevos mensajes de seguimiento.



- Mapa: se puede ver la ubicación actual del edificio en un mapa a modo orientativo.



- Email: se puede enviar un email al responsable del sensor en el edificio para comunicarse con este.
- LLamar: se puede realizar una llamada al responsable del sensor en el edificio.



- Guardar: Se podrá gestionar la temperatura y la luminosidad del edificio indicando los valores adecuados en los campos habilitados a tal propósito. Seguidamente habrá que guardar los cambios pulsando el botón guardar.
- Volver: se podrá volver a la pagina de inicio para continuar seleccionando registros.

Una vez aquí, se puede concluir indicando que se han implementado todas las funcionalidades indicadas al inicio del proyecto. Ya lo que queda es navegar entre las ventanas presentadas y gestionar la información.

IV.Conclusiones

El sistema implementado consta de dos partes con dos tecnologías claramente diferenciadas: J2EE y Android. De J2EE, poco mas se puede decir que es una plataforma totalmente madura, con cientos de productos en la comunidad y el mercado. Tiene apoyo a nivel mundial de los mejores expertos en software para evolucionarlo. Sus especificaciones y las referencias hacen que cada vez todo sea mas sencillo. Esto se nota a la hora de desarrollar, en general está todo muy atado y fino. Desarrollar en J2EE es cada vez mas rápido, lo mismo que desplegar sus aplicaciones. Por contra nos encontramos con Android, una plataforma relativamente nueva. A favor cuenta con una amplia comunidad de desarrolladores que mejoran el ecosistema cada día. En contra, que todavía no está suficientemente madura. Esto se nota en el desarrollo, donde el uso de una versión de un API a otro puede cambiar el desarrollo por completo. También se pueden encontrar multitud de foros donde se explican buenas practicas de programación en android para que las aplicaciones rindan mejor o se ahorre memoria, cuando estos aspectos deberían ser inherentes a la plataforma y transparentes para el desarrollador. En general, creo que debería de madurar todavía y mas, simplificar las cosas, y ser mas eficiente.

En otro orden de cosas, llegar hasta aquí nunca fue fácil: compaginar otras asignaturas con esta, la tecnología, diseñar las aplicaciones, aprender por mi cuenta a desarrollar aplicaciones con Android, instalar las herramientas, investigar sobre las API's que debía de utilizar en el proyecto, implementar las aplicaciones, comunicar las aplicaciones, codificar/decodificar la información que viaja por la red en UTF-8, parsear las tramas para mensajes y actuar en consecuencia, realizar las pruebas unitarias, realizar las pruebas de comunicación, realizar la documentación del proyecto, refrescar UML,...

Personalmente tenia la incertidumbre de poder realizar una aplicación con Android. Partía de cero, no conocía absolutamente nada de este ecosistema. Mi reto era poder realizar la aplicación. Poco a poco y a base de investigar, logré despegar hasta llegar a la finalización de la aplicación que hoy presento. Siempre he tratado de hacer todo el sistema lo mas sencillo tanto de implementar como de entender: pocas clases, pocas dependencias, y poca complejidad. Finalmente creo que finalmente lo he conseguido.

Me siento satisfecho del trabajo realizado, de la experiencia y del conocimiento adquirido, y creo que el haber realizado este proyecto es un punto de partida interesante para poder abordar próximos retos en esta tecnología.

V.Glosario de términos

Android.- Sistema operativo para móviles.

API.- Application Programming Interface

APK.- Android aplication package file

DAO.- Data Access Object

EAR.- Enterprise Archive

HTTP.- Hypertext Transfer Protocol

JAR.- Java Archive

MVC.- Model View Controller

ORM.- Object-Relational Mapping

POJO.- Plain Old Java Object

Tomcat.- Servidor de aplicaciones web.

UML.- Unified Modeling Language

WAR.- Web aplication archive

VI.Bibliografía

Material didáctico UOC: Ingeniería del software

Material didáctico UOC: Programación Orientada a Objetos

Material didáctico UOC: Redes

Material didáctico UOC: Estructura de redes de computadores

Material didáctico UOC: Seguridad en redes de computadores

Material didáctico UOC: Administración de redes y sistemas operativos

Android guías, referencia, recursos y API's: http://developer.android.com/

Java 1.6 API: http://docs.oracle.com/javase/6/docs/api/

Tutorial de Java: http://docs.oracle.com/javase/tutorial/

Tutorial de Android en español: http://www.sgoliver.net/blog/?p=1313

Tutorial de Android en ingles: http://www.xtensivearts.com/topics/tutorials/

XStream: http://xstream.codehaus.org/

Maven: http://maven.apache.org/

Berkeley DB Java: http://www.oracle.com/technetwork/products/berkeleydb/overview/index.html

Tomcat: http://tomcat.apache.org/

Eclipse: http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigosr2

Log4j: http://logging.apache.org/log4j/1.2/

MVC: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

DAO: http://es.wikipedia.org/wiki/Data_Access_Object

POJO: http://es.wikipedia.org/wiki/POJO

ORM: http://es.wikipedia.org/wiki/ORM