Desarrollo de Procesos de Negocio Móviles Adaptados a la Obtrusividad

Rocío Carolina Martínez Arenas

Supervisor:
Dr. Vicente Pelechano Ferragud

UNIVERSIDAD
POLITECNICA
DE VALENCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

PROS
Centro de Investigación en Métodos de Producción de Software

Diciembre 2009

A mi esposo, Hernán Darío

A mi familia

A Dios

Resumen

Los procesos de negocio pueden beneficiarse de una mayor integración de los elementos físicos, mediante el uso de tecnologías de identificación automática. Un ejemplo de ella es el uso de la Identificación por Radio Frecuencia (RFID), con la cual se consigue interconectar objetos del mundo real, con servicios de información digital, en lo que se conoce como Internet de las Cosas (Internet of Things).

En este trabajo se aborda la construcción de sistemas de soporte a procesos de negocio en el contexto de la Internet de las Cosas, partiendo de la descripción del proceso y de la obtrusividad requerida en el desarrollo de sus tareas, para obtener una solución informática que dé el soporte adecuado al proceso.

Esta tesis propone el uso de un marco conceptual para clasificar los tipos de interacción requeridos para completar las tareas del proceso de negocio, y así obtener el nivel de obtrusividad adecuado para el sistema. En particular, se aborda el desarrollo de estos sistemas aprovechando las capacidades avanzadas para la detección de objetos del mundo real que poseen los dispositivos móviles actuales.

Finalmente, se definió un caso de estudio para probar la aplicabilidad de la propuesta y su implementación en entornos móviles.

Abstract

The Business Processes can benefit from greater integration of physical objects, through the use of automatic identification technologies. An example of it, is the use of Radio Frequency Identification (RFID), which is achieved with interconnecting real world objects with digital information services, in what is known as the Internet of Things.

The present work addresses the construction of systems to support business processes in the context of the Internet of Things, based on the description of the process and obtrusiveness required in performing their tasks, to obtain a software solution to give adequate support to the process.

This thesis proposes the use of a conceptual framework to classify the types of interaction required to complete the tasks of business process, and obtain the appropriate level of obtrusiveness for the system. In particular, it addresses the development of these systems by exploiting the advanced capabilities to detect real-world objects, that have existing mobile devices.

Finally, it defines a study case to test the applicability of the proposal and its implementation in mobile environments

Agradecimientos

Este trabajo no hubiera sido posible sin el apoyo de las siguientes personas:

Vicente Pelechano Ferragud, mi director de tesis y mi apoyo durante el proceso.

Pau Giner, gran persona e investigador, orientador de este trabajo.

A todos mis profesores del Máster, quienes compartieron conmigo a través de sus clases, sus profundos conocimientos y experiencias.

A mis compañeros, quienes me acogieron y me brindaron su amistad.

A mi familia, apoyo constante en cada etapa de mi vida.

Tabla de Contenido

Lista de figuras

Lista de Tablas

1. Introducción

Los actuales sistemas de información tienen la capacidad de manejar de manera automática información digital asociada a objetos del mundo real - productos en un supermercado, recursos en una biblioteca-. Esto le permite a los objetos físicos participar activamente en los procesos de negocio reduciendo así la brecha entre los mundos físico y virtual. Esta es la visión propuesta por la Internet de las Cosas (Internet of Things[1]), la cual plantea alcanzar la integración de estos dos mundos, aumentando los elementos del mundo real con una identidad digital.

Pero para conseguir la fluidez que requieren los procesos de negocio que involucran objetos del mundo real en el desarrollo de sus tareas, es importante controlar hasta qué punto los servicios digitales asociados a los objetos pueden demandar la atención del usuario. El nivel de dicha "obtrusividad" dependerá de la tarea en particular que se esté realizando. Las capacidades avanzadas para la detección de objetos del mundo real que poseen los dispositivos móviles actuales –la mayoría con cámara digital integrada-, impulsan el desarrollo de aplicaciones móviles para dar soporte a procesos de negocio en el contexto de la Internet de las Cosas.

1.1 Motivación

El uso de dispositivos móviles está ampliamente difundido en la vida moderna. Mientras se procura reducir su tamaño en pro de una mayor portabilidad, su funcionalidad aumenta, convirtiéndolos en dispositivos inteligentes con capacidad de cómputo añadido, siendo sólo limitada por las aplicaciones que en ellos se instalen. Lo habitual es que este tipo de dispositivos incorporen de fábrica las aplicaciones más comunes: agenda de contactos, calendario, calculadora, notas, gestor de correo electrónico, conexión a Internet, etc. Pudiendo posteriormente instalar sobre ellos nuevas aplicaciones que resuelven una necesidad específica.

De esta manera el uso de dispositivos móviles permite gestionar la información que consideramos más esencial para realizar nuestras actividades, con la ventaja de tenerla siempre a mano, en cualquier lugar y a cualquier hora. La extensa funcionalidad que ofrece este tipo de

[1] http://en.wikipedia.org/wiki/Internet_of_Things

dispositivos ha sido aprovechada también por el sector comercial que ve en la utilización de dispositivos móviles una herramienta para aumentar la productividad, modificando sus procesos de negocio a través de la incorporación de tareas para ser desarrolladas por aquellos empleados que no se encuentran físicamente en las dependencias corporativas (teletrabajo, personal de ventas, consultores desplazados, etc.). Permitiendo el procesamiento en tiempo real de la información y en ocasiones, dependiendo de la infraestructura de comunicación utilizada, el acceso a las bases de datos y aplicaciones de la empresa.

Pero paralelamente a todas las ventajas que se pueden encontrar en los dispositivos móviles también se encuentran desventajas, algunas técnicas y otras funcionales, entre ellas las siguientes:

- Pantallas y dispositivos de entrada pequeños o lentos.
- Dificultades tecnológicas, entre ellas, fuente de energía limitada y menor velocidad de procesamiento.
- Dificultades en la conexión de red: tiempos de latencia muy prolongados, y ancho de banda altamente variable, por infinidad de factores como el cambio de celda, condiciones de tráfico, etc.

Adicionalmente, las aplicaciones que se desarrollan para ejecutarse sobre tales dispositivos frecuentemente compiten por la atención del usuario: demandando la constante entrada de información (cajas de diálogo, formularios, confirmaciones, etc.) o anunciando la ocurrencia de eventos (alertas, notificaciones, llamadas entrantes, etc.). Provocando interrupciones inoportunas que originan complicaciones en las actividades de la vida diaria. Como Neil Gershenfeld observa, "Hay un verdadero sentido en el cual las cosas alrededor nuestro están infringiendo una nueva clase de derecho del que no necesitábamos protección hasta ahora. Estamos gastando más y más tiempo respondiendo a las demandas de las máquinas." [22]

Conscientes de estos problemas, se están llevado a cabo diferentes investigaciones tanto en la academia como en la industria, buscando desarrollar técnicas que permitan a los dispositivos computarizados estimar la carga cognitiva del usuario y el foco de su atención, otorgándoles de esta forma un comportamiento más considerado, menos egocéntrico y obtrusivo. Este amplio campo de investigación es conocido como "*Considerate Computing*" [8]. Esta área de investigación extiende la tradicional Human-Computer Interaction HCI -que comprende el paradigma de la interfaz de usuario gráfica o basada en comandos, requiriendo la explícita entrada o salida de información-. Con nuevas técnicas que permitan a los dispositivos actuar de manera más proactiva y menos obtrusiva, sin requerir la orden explícita del usuario y en muchas ocasiones obrando fuera de su plano atencional, en lo conocido como interacción implícita.

Para permitir a los investigadores y a los diseñadores en el área de HCI, comprender los modos en los cuales las interacciones implícitas son diferentes de las interacciones explícitas, y proporcionar una guía sobre cuándo diferentes tipos de interacciones implícitas son útiles. Wendy Ju y Larry Leifer [3], han desarrollado un framework para clasificar los tipos de interacción y sus particularidades.

El diseño adecuado de la interacción con el usuario determina la usabilidad y aceptación que tenga la aplicación, en el campo para el cual fue desarrollada. Por tanto, las aplicaciones destinadas a ejecutarse sobre dispositivos móviles deben tener muy bien definido el tipo de interacción y el nivel de obtrusividad que ofrecerán a sus usuarios. Sólo de esta manera se explotará todo el potencial tecnológico que cada día se integra a este tipo de dispositivos, permitiendo que las personas y las corporaciones se beneficien más ampliamente de su uso.

1.2 Establecimiento del problema

El despliegue de procesos de negocio sobre dispositivos móviles es actualmente un campo de investigación contemplado en el área de la gestión de procesos de negocio (Business Process Management - BPM), la cual se enfoca en administrar y optimizar de forma continua las actividades y los procesos de negocio de una organización.

Los procesos de negocio que involucran objetos físicos para la realización de sus tareas requieren de acuerdo al tipo de información proporcionada por el objeto, el diseño de una apropiada interacción de usuario permitiendo conseguir el grado adecuado de fluidez en el proceso. Especificando para cada tarea en particular, la manera cómo se debe presentar y pedir la información al usuario, si debe hacerse de manera explícita o si por el contrario es mejor hacerlo implícitamente.

El trabajo que se ha desarrollado en la presente tesis, realiza una propuesta para el diseño de la interacción con el usuario en los procesos de negocio que integran elementos del mundo físico en la ejecución de sus tareas, utilizando las capacidades para la detección de objetos del mundo real que poseen los dispositivos móviles actuales. Para lo cual se plantea la siguiente pregunta:

- ¿Cómo se definen las acciones que debe realizar el usuario para completar una tarea, presentando la adecuada interactividad, y consiguiendo la agilidad que requieren los procesos que se ejecutan sobre ambientes de movilidad?

La pregunta es analizada y respondida en la siguiente sección.

1.3 Solución Propuesta

Para responder a la pregunta planteada en la sección anterior, esta tesis tiene como objetivo principal aplicar el concepto de obtrusividad al diseño de procesos de negocio móviles. Mediante la utilización del framework de interacciones implícitas introducido por Wendy Ju y Larry Leifer [3], a través del cual se pueden clasificar las interacciones requeridas para realizar una tarea en explícitas e implícitas, basándose en la iniciativa (quién dispara la interacción) y la atención (qué nivel de conciencia tiene el usuario respecto a la interacción).

Este trabajo propone realizar el diseño de cada tarea dentro del flujo de trabajo considerado, definiendo la misma funcionalidad pero a niveles distintos de obtrusividad. Para esto, se debe guiar el diseño de las acciones necesarias para completar la tarea haciendo uso del framework conceptual de interacciones implícitas. El cual ofrece un método claro para determinar cuándo es oportuno que el sistema haga preguntas al usuario, cuándo debe suministrar feedback de sus acciones, cuándo es más conveniente que el sistema actúe de manera independiente o proactiva.

Para llevar a la práctica la propuesta planteada, se desarrolló un caso de estudio que contempla tareas a realizar en un proceso de negocio móvil. Para implementar dicho caso de estudio se utilizaron los siguientes medios:

1. El proceso de negocio se despliega utilizando Presto [1], una plataforma que apoya el desarrollo de procesos de negocio móviles en el contexto de Internet de las Cosas. Permitiendo recuperar información digital asociada a un objeto físico, facilitando su integración a las actividades involucradas en el flujo de trabajo.
2. Se extendió Presto para implementarlo sobre la plataforma móvil de código abierto Android, debido a que este sistema operativo presenta una arquitectura que se adapta a las necesidades de Presto, y goza de buena aceptación entre los usuario móviles.
3. Cada tarea involucrada en el proceso de negocio se describió utilizando la notación BPMN (Business Process Modeling Notation).
4. La funcionalidad de cada tarea que constituye el proceso, se diseñó a diferentes niveles de obtrusividad, ubicándola en la región del framework de referencia que corresponde al tipo de interacción deseada.

El resultado es una solución software que expone al usuario las tareas a realizar para completar un flujo de trabajo desde diferentes perspectivas, clasificadas de acuerdo a la región del framework conceptual de referencia a la cual corresponde el tipo de interacción considerada.

1.3 Contexto de la Tesis

Esta tesis de Máster fue desarrollada bajo el contexto del centro de investigación *Centro de Investigación en Métodos de Producción de Software* de la *Universidad Politécnica de Valencia*.

El trabajo que ha hecho posible el desarrollo de esta tesis está en el contexto del siguiente proyecto de investigación:
PRESTO[2]. Proyecto orientado a dar soporte a procesos de negocio que integran elementos del mundo real, estudiando tanto métodos para el desarrollo de éstos como arquitecturas que los apoyen.

Relación con las asignaturas impartidas en el Máster en *Ingeniería del Software, Métodos Formales y Sistemas de Información*

Se aplicaron de manera directa o indirecta, en el desarrollo del presente trabajo de tesis, los contenidos de las siguientes asignaturas:

- Desarrollo de aplicaciones en Java. El contenido de esta asignatura fue de especial relevancia durante la implementación, en la plataforma móvil Android, de la aplicación desarrollada para soportar el caso de estudio propuesto en la tesis. En razón a que las aplicaciones que se despliegan sobre Android deben ser programadas Java.
- Modelado, diseño e implementación de Servicios Web. Los conceptos y principios a tener en cuenta para modelar, diseñar e implementar Servicios Web, se aplicaron durante la implementación del servicio web desarrollado para dar soporte al caso de estudio propuesto en la tesis.
- Tecnología software para ambientes Web. Los principios de Human-Computer Interaction HCI, las nociones de Business Process Management BPM y de Business Process Modeling Notation BPMN. Se aplicaron en la definición de las tareas involucradas en el proceso de negocio del caso de estudio desarrollado, y en el diseño de las interacciones de la aplicación con el usuario.
- Integración Semántica de Datos. Durante el desarrollo de la aplicación que soporta el caso de estudio propuesto, se trató con información estructurada en diferentes formatos: XML, JSON, entre otros. También se aplicaron algunos principios conceptuales relativos al tema de estudio de la asignatura para definir el modelo de datos considerado en la aplicación.

[2] http://www.pros.upv.es/labs/projects/presto

- Gestión de Modelos. Los conceptos impartidos sobre Model Driven Engineering MDE me permitieron comprender el metamodelo de Presto, y las especificaciones que deben satisfacer las soluciones basadas en Presto.
- Calidad de Sistemas de Información. Los principios de diseño aplicados al modelado de la interfaz de usuario en los sistemas de información web, pueden ser aplicados en el diseño de la interfaz de usuario de las aplicaciones móviles.

1.4 Estructura de la Tesis

El documento está organizado en seis capítulos. El capítulo 2 presenta una descripción general de algunos conceptos presentados por las disciplinas en las cuales se fundamenta este trabajo: Business Process Management, Internet de las Cosas y enfatiza sobre el área de investigación Considerate Computing. El capítulo 3 presenta una descripción general de los principios arquitecturales de Presto, especificando el mapping tecnológico de Presto sobre Android y las pautas que se deben seguir para lograr la integración Presto-Android. El capítulo 4 presenta la definición del framework conceptual de interacciones implícitas, sobre el cual se basa el desarrollo propuesto en este trabajo, y se introduce el caso de estudio sobre el cual se aplicarán dichas interacciones. Así mismo describe la implementación de cada tarea involucrada en el caso de estudio sobre Presto de acuerdo a la obtrusividad requerida. El capítulo 5 presenta las conclusiones, contribuciones realizadas por este trabajo y algunas visiones de trabajos futuros. Finalmente, el capítulo 6 presenta el listado con las referencias a los trabajos consultados durante la realización de esta tesis.

2. Contexto Tecnológico

En este capitulo se describen las tecnologías y los conceptos usados en el desarrollo del trabajo propuesto, revisando el estado del arte de las diferentes disciplinas involucradas. La Sección 2.1 introduce la definición de procesos de negocio y el lenguaje de modelado de éstos. La Sección 2.2 hace una breve presentación de la ejecución de procesos en dispositivos móviles, resaltando en esta sección el sistema operativo Android, el cual se utilizó para implementar los conceptos que el presente trabajo define. Finalmente, la Sección 2.3 presenta la visión de la Internet de las Cosas y el soporte existente en términos de tecnologías. Mencionando un campo de aplicación actual: Considerating Computing, que permite observar el comportamiento obtrusivo, la mayor parte de las veces, de los sistemas computacionales que rodean la vida moderna.

2.1. Procesos de Negocio

Un proceso se define como una serie de actividades que siguen un orden lógico establecido para obtener un fin determinado. En el ámbito empresarial, del que posteriormente se establecen modelos informáticos, en un proceso de negocio (Business Process, en inglés) se obtiene un resultado de valor real y palpable a las organizaciones, en el contexto de su negocio.

La gestión de los procesos de negocio, Business Process Management (BPM), es una disciplina que se enfoca en administrar y optimizar de forma continua las actividades y los procesos de negocio de una organización. BPM incluye prácticas y políticas de gestión; métricas y procesos de control, así como herramientas informáticas para diseñar, ejecutar, administrar y monitorizar procesos de negocio.

Para el contexto del presente trabajo, se hace especial relevancia al modelado y ejecución de procesos de negocio en dispositivos móviles, estos temas se desarollarán en las siguientes sub-secciones.

2.1.1 Modelado de Procesos de Negocio

En el modelado de procesos de negocio, han sido usadas diferentes notaciones, entre las que se destacan cronológicamente: IDEF [24]; diagramas de actividades UML [23]; ebXML BPSS

[25] y Business Process Modeling Notation (BPMN) [26]. Esta última (actualmente la más extendida) proviene del consorcio Business Process Management Initiative (BPMI), del cual participaron en su tiempo importantes empresas (Adobe Systems, IBM Corporation, SAP AG, entre otras) y cuyo objetivo era la definición de una notación que fuera fácilmente comprendida por los distintos participantes en el modelado de procesos (analistas, desarrolladores y clientes). Posteriormente, en junio de 2005 BPMI es adoptada por OMG como notación estándar en el modelado de procesos de negocio, unificando y consolidando de esta manera, diferentes notaciones y diferentes puntos de vista.

BPMN proporciona cuatro diferentes categorías de elementos de modelado, con las cuales es posible representar un proceso, éstas son:

- Elementos de flujo
- Elementos de conexión
- Swimlane (Franjas de responsabilidad), y
- Artefactos.

Elementos de flujo

Se usan para definir cómo trabaja el proceso de negocio (comportamiento). Existen tres tipos de elementos de flujo:

- **Evento**, sirve para representar algo que ocurre en el proceso. Este 'algo' tiene una causa (disparador de alguna acción) y un efecto (un resultado). Existen tres tipologías de eventos, según su representación en el proceso:
  - Eventos de inicio (Start), para representar flujos de entrada.
  - Eventos intermedios (Intermediate), para representar flujos tanto de entrada como de salida.
  - Eventos de fin (End), para representar flujos de salida.

La Figura 2.1 muestra los eventos completos que están definidos en la notación:

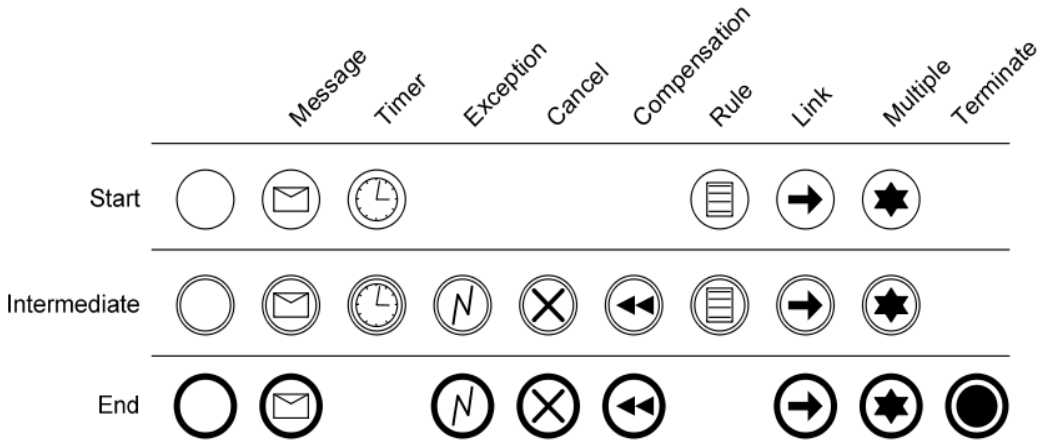


Fig. 2.1. Notación BPMN – Elementos de flujo: Eventos

- **Actividad**, es un término genérico para denominar una acción hecha por un participante de un proceso. Las Actividades pueden ser atómicas (task) o no atómicas (sub-process). Además, se incluyen atributos en la notación para indicar si la actividad se realiza una sola vez o se repite, detallando si las repeticiones se hacen de manera secuencial (looping) o en paralelo (instancias múltiples). La Figura 2.2 muestra los tipos de Actividades definidas:

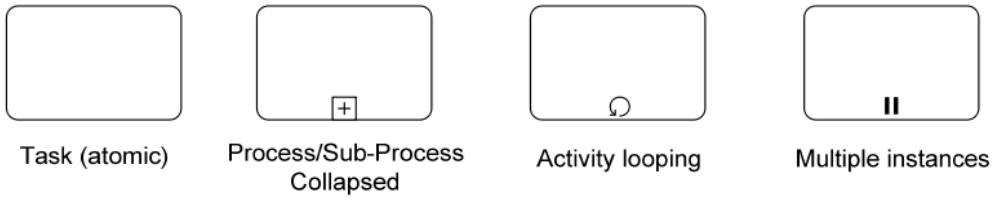


Fig. 2.2. Notación BPMN – Elementos de flujo: Actividades

- **Bifurcaciones (Gateways)**, es usado para controlar la convergencia y/o divergencia en la secuencia del flujo de proceso. Estas bifurcaciones determinan divisiones, uniones y combinaciones. La Figura 2.3 muestra los tipos de Bifurcaciones definidas:

Fig. 2.3. Notación BPMN – Elementos de flujo: Bifurcaciones

Elementos de conexión

Son usados para conectar entre sí, elementos de flujo, o elementos de flujo con otros elementos BPMN. En la notación BPMN, se definen los siguientes tipos de elementos de conexión:

- **Secuencias**, indican el orden en el cual las actividades son ejecutadas en el proceso. Este tipo de elemento se especializa en: Normal, Conditional y Default.
- **Mensajes**, indican el flujo de los mensajes que intervienen entre dos participantes.
- **Asociaciones**, son usados para indicar información de asociación (en texto o gráfica) a los elementos de flujo.

La Figura 2.4 muestra los diferentes tipos de elementos de conexión.

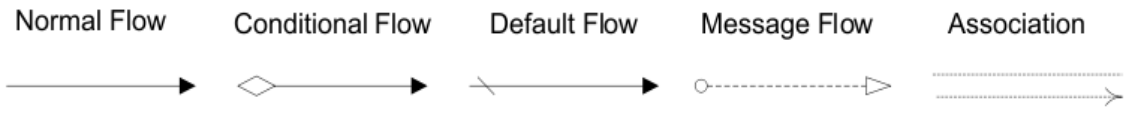


Fig. 2.4. Notación BPMN – Elementos de conexión

Swimlanes

Estos elementos representan la responsabilidad en los diagramas: "quién hace qué". Son rectángulos que definen el proceso y los participantes. Hay dos tipos de elementos de Swimlanes:

- Pool, representa un participante dentro del proceso.
- Lane, subdivide los elementos Pool en entidades más lógicas. Por ejemplo, un Pool puede ser una empresa y los Lanes puede ser los departamentos en que se divide.

La Figura 2.5 muestra las franjas de responsabilidad:

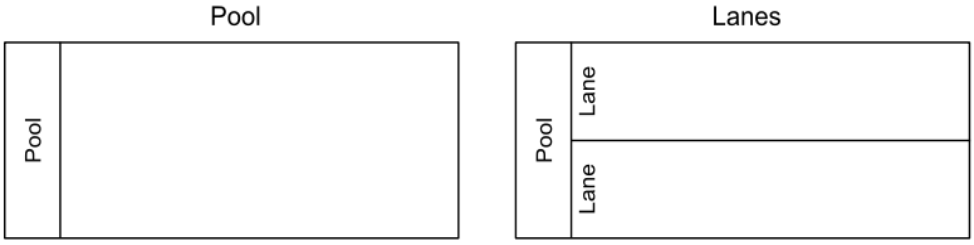


Fig. 2.5. Notación BPMN – Swimlanes

Artefactos

Son elementos que no influyen en el cambio del flujo de ejecución, pero su objetivo es mejorar la comprensión del proceso. La notación BPMN define los siguientes artefactos:

- **Grupo**, permite agrupar actividades, como propósito de documentación.
- **Objeto de datos**, representa la información que una actividad requiere o su resultado.
- **Anotaciones**, permite adicionar información, para clarificar la lectura del diagrama.

La representación de estos elementos es mostrada a continuación, en la Figura 2.6:

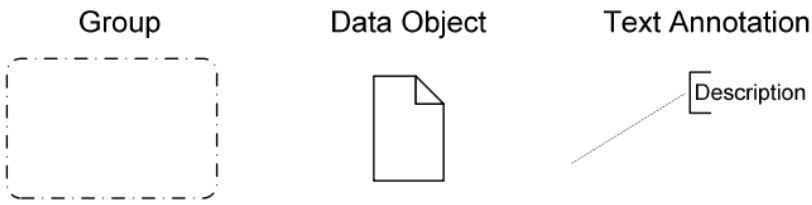


Fig. 2.6. Notación BPMN – Artefactos

Además de estas cuatro categorías, la notación BPMN maneja conceptos avanzados de modelado tales como: manejo de excepciones, transacciones y compensación.

2.1.2 Ejecución de procesos de negocio móviles

Muchos procesos de negocio tradicionales -tales como la aprobación de un préstamo, el proceso para una reclamación de seguros, entre otros- pueden ser naturalmente modelados como flujos de trabajo (workflows). Los sistemas workflow coordinan y monitorean la ejecución de tareas mediante múltiples agentes activos (personas o servicios software) hacia la realización de un objetivo común. Esto ha motivado el desarrollo de estándares Web, tales como Business Process Execution Language (BPEL), el cual describe estos procesos usando un lenguaje común. Cada tarea en un proceso BPEL es representada como un servicio que es invocado usando el Simple Object Access Protocol [18] (SOAP). Un servidor BPEL centralizado compone estos servicios dentro de procesos complejos ejecutando una serie ordenada de invocaciones de acuerdo a las especificaciones del usuario. BPEL ha tenido gran aceptación en la definición de procesos de negocio debido a que está construido sobre estándares como XML y SOAP los cuales están ya ampliamente desplegados [19].

Por otra parte, la ubicuidad de los dispositivos móviles con propiedades computacionales integradas, como las PDAs y los teléfonos móviles, ofrecen una nueva y creciente plataforma para el despliegue y ejecución de aplicaciones colaborativas. Aunque cada dispositivo es individualmente mucho menos potente que un servidor independiente, su potencial de computación y comunicación agregado es notable. Los siguientes ejemplos corresponden a aplicaciones que podrían incorporar las ventajas de tales dispositivos [19]:

- Un trabajo de construcción complejo requiere que los empleados lleven a cabo ciertas tareas en un orden específico. El trabajo es modelado como un workflow, el cual está divido en una serie de procesos distribuidos. Cada proceso es distribuido a la correspondiente PDA del empleado, cuando cada proceso se ejecuta, ésta le dice al trabajador la siguiente tarea que debe ser completada.
- Un banco realiza una serie de comprobaciones y evaluaciones durante la aplicación de su proceso de préstamo, y puede requerir que el administrador de su aprobación para algunos créditos. Mientras el administrador está fuera en un viaje de negocios, su teléfono móvil está equipado con un servicio el cual le pide que apruebe o desapruebe ciertos créditos. Este servicio es invocado como parte del workflow de la aplicación de préstamo.

Aplicaciones móviles como las descritas anteriormente pueden beneficiarse grandemente de estándares Web como BPEL y SOAP. Definiendo un lenguaje común para interacciones entre servicios y flujo de datos, estos estándares permiten la composición de servicios simples dentro de poderosas aplicaciones distribuidas. Desafortunadamente, las aplicaciones móviles plantean varios desafíos importantes que no reúne el estado actual del arte en los sistemas SOAP y BPEL:

1. Los dispositivos móviles típicos tienen su hardware severamente restringido por lo cual requieren una infraestructura muy ligera de software.
2. En ausencia de una conexión estable a Internet, puede ser imposible, impracticable, o muy costoso que los dispositivos móviles contacten servidores centralizados.
3. Los enlaces a redes inalámbricas en los dispositivos móviles pueden ser interrumpidos frecuentemente e impredecibles.

Por tanto se necesita un sistema middleware basado en servicios, ligero y descentralizado, el cual pueda ejecutar sobre la marcha replaneación, reasignación, y/o reconfiguración para enfrentar los fallos de la red.

Adicionalmente, para que los procesos móviles soporten estándares en un ambiente de ejecución móvil, es necesario que las máquinas de workflow móvil cumplan ciertos requisitos [21]:

- Protocolos de comunicación (interna y externa): SMS, MMS, Bluetooth, email, y HTTP
- Lenguajes de ejecución de procesos: WS-BPEL, WSDL, XML, y XPath
- Formatos de mensaje: SOAP y SOAP con adjuntos
- Lenguajes de interfaz de usuario: tecnologías web tales como XHTML
- Integración a aplicaciones nativas del teléfono: mensajería, navegador, mapas, lanzador, agenda y calendario
- Integración al ambiente móvil de la ejecución: tiempo e información de la zona horaria del usuario, cobertura de la red, nivel de batería, localización, velocidad, distancia, temperatura, y otros sensores externos y datos ambientales
- Garantías de calidad: limitaciones en la memoria y ancho de banda de la red.

Teniendo en cuenta los requisitos que implica la ejecución de procesos en un ambiente móvil, diferentes soluciones han surgido para dar soporte a la implementación de máquinas de ejecución de procesos de negocio sobre dispositivos móviles. A continuación se mencionan algunas de ellas:

Magi[20]. Es un proyecto de código abierto y se puede descargar libremente en: http://magi.endeavors.org. Es un framework arquitectural que explícitamente dirige la coordinación de mensajería e-business y la despliega sobre un amplio rango de plataformas de computación. Particularmente Magi hace más fácil la configuración y construcción de aplicaciones distribuidas XML y Java destinadas a cualquier PC de escritorio, laptop, o palmtop que corran un servidor Magi.

La arquitectura de Magi tiene dos componentes principales:

- Un servidor micro-Apache HTTP. Una versión ligera de Apache HTTP que proporciona interoperabilidad a lo largo de un amplio rango de dispositivos inteligentes con acceso a Internet inalámbrico, tales como PDAs, o smartphones.
- Una interfaz genérica extensible. La arquitectura modular de Apache y la extensibilidad de HTTP permiten la fácil personalización de servicios.

Magi, es esencialmente un servidor Apache HTTP que usa HTTP como su protocolo de comunicación central. Implementa otras definiciones de protocolos como módulos Apache o extensiones HTTP que pueden ser cargadas por demanda. En la Figura 2.7 se aprecia la configuración y arquitectura de componentes de Magi. Los servicios Magi pueden ser

agregados en la marcha, permitiendo así que otros servidores o clientes Magi descubran si un cierto servidor Magi puede ayudar a responder la solicitud particular.

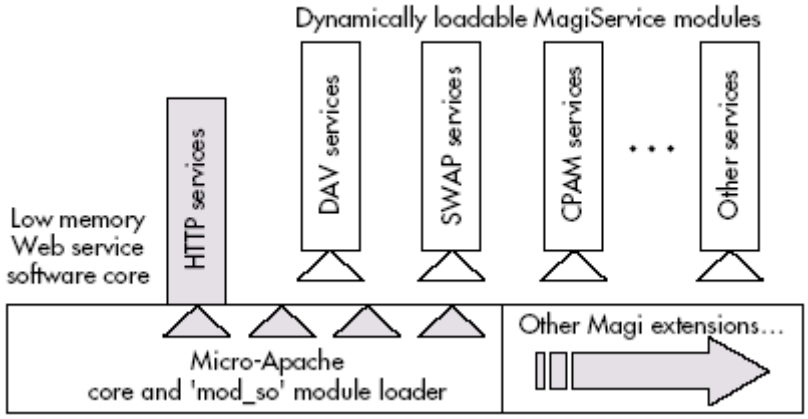


Fig. 2.7. Configuración y arquitectura de componentes de Magi

Sliver[19]. Es una máquina de ejecución de procesos de negocio móviles basada en BPEL, que soporta una amplia variedad de dispositivos desde teléfonos móviles hasta PC's de escritorio. Sliver se basa en tres principios arquitecturales:

1. Una clara separación entre la comunicación y el proceso. Los componentes de la comunicación pueden ser intercambiados sin afectar a los componentes del proceso.
2. Solamente depende de dos ligeras librerías externas, las cuales fueron diseñadas pensando en los dispositivos móviles. Esto asegura que Sliver puede ser desplegado sobre un amplio rango de dispositivos.
3. Los componentes SOAP de Sliver no dependen de sus componentes BPEL. Los desarrolladores pueden desplegar servicios SOAP sobre dispositivos móviles sin necesitar el motor BPEL completo de Sliver.

La Figura 2.8 ilustra los componentes arquitecturales de Sliver.

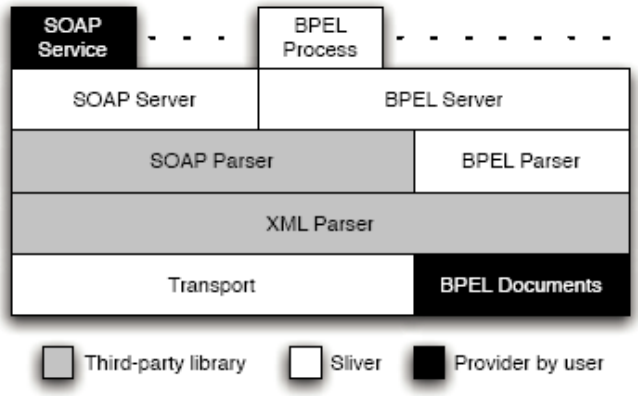


Fig. 2.8. Arquitectura del motor de ejecución Sliver

El desarrollo de máquinas de ejecución de procesos de negocio como Magi o Slider, son un importante paso hacia el objetivo a largo plazo de crear aplicaciones que soporten el despliegue de actividades colaborativas sobre dispositivos móviles. Los cuales tienen el potencial suficiente para albergar sofisticadas aplicaciones que facilitan la comunicación entre las personas y los sistemas externos.

2.2. Dispositivos Móviles

El mercado de teléfonos inteligentes (smartphones) crece anualmente, los fabricantes que lideran el sector son Nokia, RIM - fabricante de BlackBerry, y Apple - fabricante de Iphone[3]. Conscientes de las ventajas que trae consigo la movilidad para el sector corporativo, tanto fabricantes como desarrolladores de sistemas operativos móviles, se esfuerzan por incorporar en sus plataformas aplicaciones orientadas a apoyar a las empresas en la ejecución de procesos de negocio en un ambiente de movilidad.

Actualmente lidera el campo de los smartphones corporativos la plataforma BlackBerry[4], creada por la compañía canadiense RIM (Research In Motion), su principal atractivo consiste en su sistema de gestión del correo electrónico, que permite que los mensajes se reciban en tiempo real en el terminal. Según la opinión de expertos, continua siendo la mejor solución de este tipo en el mercado, aunque en los últimos tiempos su competencia directa, como es el caso de Apple o Nokia, han incluido en sus plataformas servicios que realizan la misma función en sus teléfonos.

En EEUU la mayoría de los empresarios utilizan en su trabajo diario el iPhone, de tal forma que las grandes empresas de software profesional están adaptando utilidades para el terminal de Apple. La compañía de Redmond ha apoyado activamente el giro del iPhone -inicialmente orientado al público final- hacia el mundo empresarial, incluyendo por ejemplo compatibilidad con servidores Exchange hasta la versión 2007 de estos, una característica indispensable para los entornos corporativos desde las pequeñas y medianas empresas hasta las grandes corporaciones.

[3] http://www.noticiasdot.com/wp2/2009/11/06/crece-en-un-42-la-venta-mundial-de-smartphones/#more-21141

[4] http://www.noticiasdot.com/wp2/2009/10/01/blackberry-se-juega-su-futuro-iphone-android-y-windows-aprietan-fuerte/

Por su parte, Nokia sigue siendo un contendiente fuerte en todos los sentidos, haciendo tender a Symbian al entorno corporativo con funcionalidades como el soporte para servidores Exchange y terminales con un formato parecido a los teléfonos BlackBerry.

Por otra parte, Android de Google, continúa atrayendo a diversos fabricantes, entre ellos HTC y Samsung, y se prevé una auténtica explosión de terminales con este sistema operativo para el 2010. Debido a que la implementación desarrollada en este trabajo se realizó utilizando esta plaforma, en la siguiente sección se proporcionan detalles específicos sobre Android y sus componentes.

2.2.1 Android

Android es un sistema operativo móvil de código abierto iniciado por Google pero ahora administrado por el Open Handset Alliance. Este consorcio incluye prestigiosos fabricantes tales como HTC, Motorola, Samsung o Sony Ericsson y grandes portadoras tales como Sprint, T-Mobile o Vodaphone.

Android es un sistema operativo basado en Java que corre sobre el kernel Linux 2.6. El sistema es muy ligero y está completamente equipado. Las aplicaciones Android se desarrollan usando el lenguaje de programación Java y pueden ser llevadas fácilmente a una nueva plataforma.

Características[5]

- Framework de aplicación que permite reemplazar y reusar componentes
- Máquina virtual Dalvik optimizada para dispositivos móviles
- Browser integrado basado en el motor de código abierto WebKit
- Gráficos optimizados potenciados por una librería gráfica 2D particular, gráficos 3D basados en la especificación OpenGL ES 1.0.
- Soporte para almacenamiento de datos estructurados con la base de datos SQLite
- Soporte para audio, video, y múltiples formatos de imágenes.
- Telefonía GSM (depende del hardware)
- Bluetooth, EDGE, 3G, y WiFi (depende del hardware)
- Cámara, GPS, compás, y acelerómetro (depende del hardware)
- Rico ambiente de desarrollo que incluye un emulador del dispositivo, herramientas para hacer debugging, un plugin para el Eclipse IDE, entre otros.

[5] http://developer.android.com/guide/basics/what-is-android.html

Muchas de las características listadas anteriormente también están disponibles en otros SDKs móviles. Pero las siguientes son algunas de las características que lo hacen único[6]:

- Aplicaciones Google Map. Google Map para móviles es tremendamente popular, y Android ofrece un control Google Map atómico y reusable para utilizarlo en las aplicaciones propias. El widget MapView permite desplegar, anotar y manipular un Google Map dentro de sus componentes Activity para construir aplicaciones basadas en mapas usando la interfaz familiar de Google Maps.
- Servicios en background. Los servicios en background permiten crear aplicaciones que usan un modelo conducido por eventos, trabajando silenciosamente mientras otras aplicaciones están siendo usadas o mientras el móvil permanece ignorado hasta que suena o vibra obteniendo la atención del usuario.
- Datos compartidos y comunicación interproceso. Usando Intents y Content Providers, Android permite que las aplicaciones intercambien mensajes, ejecuten procesos, y compartan datos, incluso con las aplicaciones nativas. Restringiéndose a un completo mecanismo de seguridad basado en permisos.
- Todas las aplicaciones son creadas iguales. Android no diferencia entre aplicaciones nativas y aquellas desarrolladas por terceras partes. Esto da a los consumidores un poder sin precedentes permitiéndoles reemplazar cada aplicación nativa con una desarrollada por terceras partes que tiene acceso a los mismos datos y al hardware.
- Aplicaciones de mensajería P2P Interdevice. Android ofrece mensajería peer-to-peer que soporta presencia, mensajes instantáneos, y comunicación interaplicación/interdispositivo.

2.2.1.1 Arquitectura

La Figura 2.9 muestra los principales componentes del sistema operativo Android. Cada sección se describe en más detalle a continuación.

- **Linux Kernel**

  El núcleo de servicios (incluyendo drivers hardware, administración de procesos y memoria, seguridad, red, etc.) son manejados por el kernel Linux 2.6. El kernel también proporciona una capa de abstracción entre el hardware y el resto de la pila.

- **Librerías**

  Corriendo encima del kernel, Android incluye varias librerías en C/C++ tales como SSL, librerías gráficas que incluyen SGL y OpenGL para gráficos 2D y 3D, SQLite para soporte a bases de datos, entre otras.

- **Android Run Time**

  El Android run time es el motor que potencia las aplicaciones y junto con las librerías forma las bases para el framework de la aplicación.

  Librerías Core. Mientras el desarrollo en Android es realizado en Java, Dalvik no es una Java VM. El núcleo de librerías de Android proporciona la mayoría de la funcionalidad disponible en el núcleo de librerías de Java.

  Dalvik Virtual Machine. Dalvik es una máquina virtual que ha sido optimizada para asegurar que un dispositivo pueda correr múltiples instancias eficientemente.

- **Framework de Aplicación**

  El Framework de Aplicación proporciona las clases usadas para crear aplicaciones Android. También proporciona una abstracción genérica para el acceso a hardware y maneja la interfaz de usuario y los recursos de la aplicación.

- **La Capa de Aplicación**

  Todas las aplicaciones, nativas y de terceras partes, están construidas sobre la capa de aplicación usando el mismo API de librerías. La capa de aplicación corre dentro del Android run time usando las clases y servicios disponibles desde el framework de aplicación.

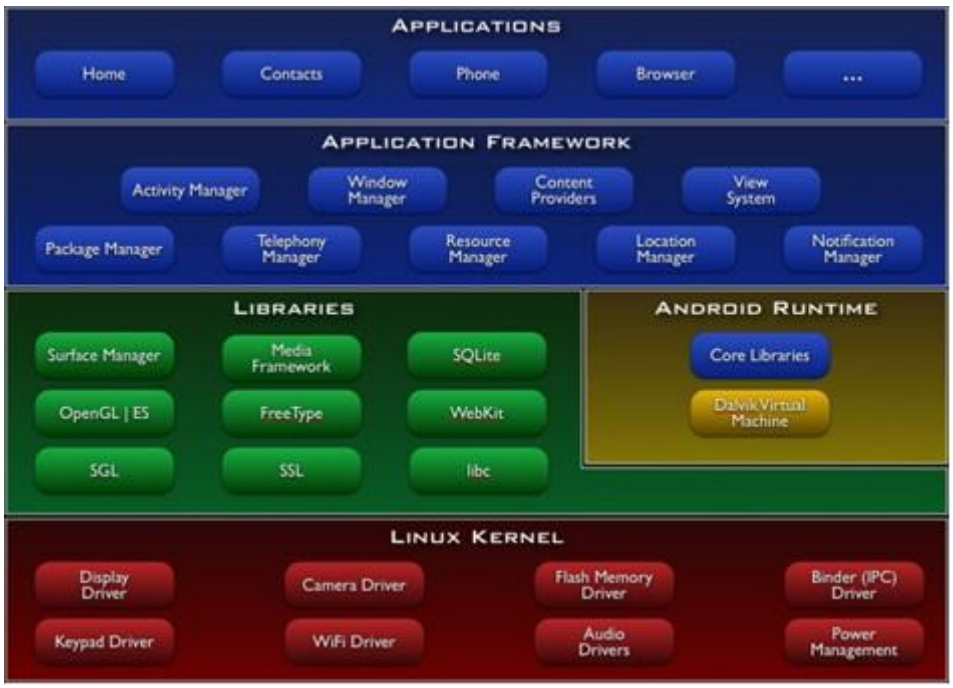


Fig. 2.9. Componentes del sistema operativo Android

2.2.1.2 Fundamentos de las Aplicaciones

Las aplicaciones Android son escritas en el lenguaje de programación Java. El código Java compilado -junto con cualquier dato y archivo de recurso requerido por la aplicación- es fusionado por la herramienta *appt* dentro de un *paquete Android*, el cual es un archivo marcado

por el sufijo *.apk*. Este archivo es el vehículo para distribuir la aplicación e instalarla en los dispositivos móviles, es este el archivo que los usuarios descargan a sus dispositivos. Todo el código en un único archivo *.apk* es considerado como una aplicación.

2.2.1.3 Componentes de una Aplicación

Una característica central de Android es que una aplicación puede hacer uso de elementos de otras aplicaciones (obteniendo los permisos requeridos para ello). Favoreciendo de este modo el reuso de componentes. Por lo tanto, a diferencia de las aplicaciones en la mayoría de los otros sistemas, las aplicaciones Android no tienen un único punto de inicio (no hay función main(), por ejemplo). En lugar de eso, tiene *componentes* esenciales que el sistema puede instanciar y correr cuando los necesite. Hay cuatro tipos de componentes:

Actividades (Activity)

Una actividad (*Activity)* presenta la interfaz de usuario visual y los medios de interacción necesarios para que el usuario desarrolle una tarea. Por ejemplo, una *Activity* puede presentar un menú con una lista de ítems que el usuario puede escoger, o puede desplegar fotografías con sus respectivas etiquetas, etc. Una actividad puede llamar a una segunda actividad para ofrecer la funcionalidad requerida por la aplicación. Aunque las actividades trabajan juntas para formar una interfaz de usuario coherente, cada actividad es independiente de las otras. Cada una está implementada como una subclase de la clase base Activity.

Servicios (Services)

Un servicio no tiene interfaz de usuario visual, sino que corre en el background durante un periodo indefinido de tiempo. Por ejemplo, un servicio puede tocar música de fondo mientras que el usuario realiza otras actividades con su dispositivo, o puede buscar datos en la red y proporcionar los resultados a las actividades que lo necesiten. Cada servicio extiende la clase base Service.

Broadcast receivers

Un Broadcast receiver es un componente que recibe y reacciona a los avisos de broadcast. Muchos broadcasts se originan en el código del sistema, por ejemplo, avisos anunciando que la zona horaria ha cambiado, que la batería está baja, que el usuario cambió una preferencia, entre otros. Las aplicaciones también pueden iniciar broadcast, por ejemplo, dejar saber a otras aplicaciones que algunos datos han sido descargados al dispositivo y están disponibles para que ellas los utilicen. Una aplicación puede tener cualquier número

de broadcast receivers para responder a cualquier anuncio que se considere importante. Todos los receivers extienden la clase base BroadcastReceiver.

Los Broadcast receiver no despliegan interfaz de usuario. Sin embargo, ellos pueden iniciar una actividad en respuesta a la información que reciben, o pueden hacer uso del NotificationManager para alertar al usuario. Las notificaciones pueden atraer la atención del usuario de varias maneras, haciendo que el dispositivo se ilumine, que vibre, que suene, etc. Las notificaciones típicamente ponen un icono en la barra de estatus, el cual el usuario puede abrir para obtener el mensaje.

Content providers

Un content provider hace un conjunto específico de datos de una aplicación disponible para las otras aplicaciones. Los datos pueden ser almacenados en archivos del sistema, en una base de datos SQLite, o en cualquier otro medio de almacenamiento. El content provider extiende la clase base ContentProvider para implementar un conjunto estándar de métodos que permiten a otras aplicaciones recuperar y almacenar datos del tipo que éste controla.

Cuando hay una solicitud que debe ser manejada por algún componente en particular, Android asegura que el proceso vinculado al componente de la aplicación está corriendo, iniciándolo en caso de ser necesario, y que la instancia apropiada del componente está disponible, creándola si es necesario.

2.2.1.4 Activando componentes: intents[6]

Las actividades, los servicios y los broadcast receivers, son activados por mensajes asíncronos llamados intents. Un intent es un objeto Intent que alberga el contenido del mensaje, es una estructura de datos pasiva que mantiene una descripción abstracta de una operación que debe ser ejecutada. El mensaje puede contener por ejemplo, una solicitud para que una actividad presente una imagen al usuario o le permita editar algún texto.

Principalmente un objeto Intent puede contener, entre otros, los siguientes parámetros:

Parámetro	Descripción
Component name	El nombre del componente que debe manejar el intent, debe corresponder al nombre completo de la clase que implementa el componente incluyendo el paquete en el que se encuentra. Este parámetro es opcional,

[6] http://developer.android.com/guide/topics/intents/intents-filters.html

	si está establecido, el objeto Intent es enviado a una instancia de la clase designada. Si no es establecido, Android usa otra información del objeto Intent para encontrar un destinatario apropiado.
Action	Una cadena de texto nombrando la acción a ser ejecutada. La clase Intent define un número de acciones constantes, incluyendo las siguientes: ACTION_CALL. Inicia una llamada telefónica. ACTION_TIMEZONE_CHANGED. Anuncia que la zona horaria ha cambiado. Pero el desarrollador de la aplicación puede definir sus propios parámetros action para activar los componentes en la aplicación.
Category	Una cadena de texto que contiene información adicional sobre la clase de componente que debe manejar el intent. Cualquier número de descripciones de categorías puede ser puesto en un objeto Intent. Como para las acciones, la clase Intent define varias categorías contantes incluyendo las siguientes: CATEGORY_HOME. La actividad se despliega en la home screen, la primera pantalla que el usuario ve cuando el dispositivo se enciende o cuando la tecla HOME es presionada. CATEGORY_LAUNCHER. La actividad puede ser la actividad inicial de una tarea y es lanzada en el nivel superior de la aplicación. Al igual que con las actions un desarrollador puede definir sus propios parámetros category para los componentes de su aplicación.
Extras	Parejas clave-valor que contienen información adicional que debe ser entregada al componente que maneja el intent. El objeto Intent tiene una serie de métodos put_() para insertar varios tipos de datos extra y un conjunto similar de métodos get_() para leer los datos.

Tabla 2.1. Parámetros objeto Intent

Intent filters

Las actividades, los servicios, y los broadcast receivers pueden tener uno o más intent filters para informar al sistema cuáles intents ellos pueden manejar. Cada filter describe una capacidad del componente, un conjunto de intents que el componente está preparado para recibir y manejar. Un componente tiene filters separados para cada trabajo que puede hacer, para cada cara que puede presentar al usuario.

Un intent filter es una instancia de la clase IntentFilter. Sin embargo, dado que el sistema Android debe conocer las capacidades de un componente antes que pueda lanzarlo, los intent filters son generalmente establecidos en el archivo manifest de la aplicación (AndroidManifest.xml) como elementos <intent-filter>.

Archivo Manifest

Antes que Android pueda iniciar un componente de la aplicación, debe conocer qué componentes existen. Por lo tanto, las aplicaciones declaran sus componentes en un archivo manifest que se encuentra dentro del paquete Android, siendo incorporado al archivo .apk que también mantiene el código de la aplicación, los archivos y recursos.

El manifest es un archivo XML estructurado y siempre es llamado AndroidManifest.xml para todas las aplicaciones. Tiene elementos adicionales a la declaración de los componentes de la aplicación, tales como el nombre de cualquier librería que la aplicación necesite (aparte de las librerías Android), y los permisos que la aplicación espera obtener.

Pero la tarea principal del manifest es informar a Android sobre los componentes de la aplicación. Por ejemplo, una actividad puede ser declarada como sigue:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest . . . >
   <application . . . >
      <activity android:name="com.example.project.FreneticActivity"
              android:icon="@drawable/small_pic.png"
              android:label="@string/freneticLabel"
              . . . >
      </activity>
      . . .
   </application>
</manifest>
```

El atributo name del elemento <activity> nombra a la subclase de Activity que implementa la actividad. Los atributos icon y label apuntan a archivos de recursos que contienen un ícono y una etiqueta que pueden ser desplegados a los usuarios para representar la actividad.

Los otros componentes son declarados de una forma similar: elementos <service> para servicios, elementos <receiver> para broadcast receivers, y elementos <provider> para content providers. Las actividades, los servicios y los content providers que no están declarados en el manifest son invisibles para el sistema y en consecuencia nunca son ejecutados. Sin embargo,

los broadcast receivers pueden alternativamente ser creados en código (como objetos BroadcastReceiver) y registrados en el sistema llamando el método Context.registerReceiver().

2.3. La Internet de las Cosas

La visión de la Internet de las cosas propone aumentar los elementos del mundo real con una identidad digital, logrando así la integración de los mundos real y virtual. En la Internet de las Cosas, los elementos del mundo real no requieren ser aumentados con complejas capacidades computacionales sino solamente ser etiquetados con un único identificador para hacerlos reconocibles a los sistemas computacionales. Con una identidad digital los servicios que los Sistemas de Información ofrecen pueden alcanzar el mundo real [28]. Esta idea fue bien ilustrada por Bruce Sterling en su charla en The Emerging Technology Conference en 2006:

> "No estamos hablando de un objeto inteligente con grandes capacidades de cálculo. Sino del objeto diario, el más barato, la cosa más obvia que uno puede comprar o usar. Excepto que tiene una identidad digital única, de manera que llega a ser rastreado, clasificado, organizado, en el espacio y en el tiempo."

Los avances en las tecnologías de Identificación Automática (Auto-ID) han ayudado a iniciar el movimiento de la visión de la Internet de las Cosas hacia la realidad. Auto-ID permite que objetos del mundo real sean considerados automáticamente por un sistema software, haciendo que los objetos no dependan ya de los humanos. Gracias a la Auto-ID, la gente, los lugares y las cosas pueden ser identificados en una variedad de diferentes modos. Identificación por Radio Frecuencia (RFID), Smartcards, códigos de barras, bandas magnéticas, y memorias de contacto, son algunas de las tecnologías Auto-ID disponibles.

Los objetos automáticamente identificables reciben diferentes nombres, tales como Spimes (objetos que son rastreables en espacio y tiempo), UFOs (Ubiquitous Findable Objects) o EKOs (Evocative Knowledge Objects). Esta heterogeneidad en terminología muestra que la Internet de las Cosas está todavía bajo construcción.

Desde las diferentes aplicaciones emergentes para la Internet de las Cosas, el presente trabajo está particularmente interesado en la integración de objetos del mundo real en los procesos de negocio. El paradigma de la Internet de las Cosas puede proporcionar numerosos beneficios en este campo, conduciendo a interesantes desafíos y oportunidades en diferentes áreas de negocio tales como mantenimiento y reparación, seguridad y responsabilidad, marketing, entre otras.

Hay un creciente interés en las tecnologías de la Internet de las Cosas desde la academia y la industria. Algunos prototipos han sido desarrollados para experimentar sus beneficios en el contexto de ventas al por menor en tiendas de comestibles, mantenimiento de aviones, entre otros. Desarrollos experimentales son llevados a cabo por diferentes compañías para mejorar sus procesos, por ejemplo, *Infineon* usa Auto-ID en la administración de la cadena de frío en orden a controlar la temperatura durante el transporte de productos químicos. [28]

El incremento en el nivel de madurez de la Internet de las Cosas se demuestra por la presencia de Auto-ID en muchos sistemas en producción real. Auto-ID está presente en las tarjetas de los empleados, en los tickets para controlar el acceso a algún evento, en las tarjetas de transporte para los respectivos sistemas de Hong Kong y Londres, por mencionar algunos de los ejemplos basados en tecnologías Auto-ID.

Entre las tecnologías usadas para apoyar la visión de la Internet de las Cosas, se contempla el uso de sensores para capturar la información asociada con los objetos físicos, y es así como surge una nueva disciplina conocida como *Considerate Computing*, la cual propone dotar a los dispositivos electrónicos que rodean nuestra cotidianidad con sensores, micrófonos o cualquier otro medio, que los haga en cierto modo conscientes de las situaciones y del mundo real que los rodea, dándoles a su vez un comportamiento menos obtrusivo y egocéntrico. Los principios comprendidos por la Considerate Computing son enunciados a continuación.

2.3.1 Considerate Computing

La sociedad se encuentra entrelazada por aproximadamente tres billones de teléfonos conectados a una red, ordenadores, luces de tráfico, incluso refrigeradores, porque estos artículos hacen la vida más conveniente y nos mantienen disponibles para quienes nos interesan. Pero los artefactos digitales que rodean la vida moderna demandan cada vez más de nuestra atención, llegando incluso a producir situaciones de agobio y frustración con sus inoportunas interrupciones. Aunque simplemente se pueden apagar los teléfonos, cerrar los programas de e-mail, apagar las alarmas, para evitar ser interrumpidos en un momento inadecuado, usualmente no se hace. Simplemente se sufren las consecuencias [8].

Las personas se esfuerzan por mantener en mente una lista de cosas por hacer. Una interrupción de sólo 15 segundos causa que la mayoría de la gente pierda parte de esa lista de deberes, concluyen experimentos realizados por Gilles O. Einstein de Furman University [9]. Numerosos estudios han demostrado que cuando la gente es interrumpida inesperadamente, ellos no sólo trabajan de manera menos eficiente sino que también cometen más errores. Este

no es simplemente un problema de productividad; para doctores, soldados, conductores y pilotos, errores de inatención pueden ser muy peligrosos.

Actualmente un grupo de investigadores están tratando de enseñar a los ordenadores, teléfonos, coches y otros artefactos a comportarse menos egocéntricamente y a parecer más como colegas considerados. Eric Horvitz de Microsoft Research, dice: "Si nosotros pudiéramos darle a nuestros ordenadores y teléfonos algún entendimiento de los límites de la atención y la memoria humana, esto los haría parecer mucho más reflexivos y gentiles."

Para hacer esto, las máquinas necesitan tres clases de nuevas habilidades: detección, razonamiento y comunicación. Primero un sistema debe inferir o detectar dónde está su dueño y lo que está haciendo. Luego debe pesar el valor de los mensajes que quiere transportar contra el costo de la interrupción. Posteriormente tiene que elegir el mejor modo y hora para entregarlos.

Cada uno de estos aspectos trasciende los límites de la informática para penetrar en temas relacionados con la privacidad, complejidad o confiabilidad. Sin embargo, los sistemas de computación "atentos" han comenzado a aparecer en los Volvos más recientes, e IBM ha introducido el software de comunicación Websphere con un sentido básico de ocupabilidad. También Microsoft ha estado corriendo extensivas pruebas internas, desde el 2003, de un sistema mucho más sofisticado, para que dentro de pocos años, las compañías puedan ofrecer a sus empleados una versión software de una recepcionista personal. Pero se debe tener en cuenta, que por definición, un sistema atento es aquel que siempre está vigilando. Ese computador considerado puede llegar a conocer más sobre nuestros hábitos de trabajo que nosotros mismos.

Estudios Realizados

Varios estudios, entre ellos los realizados por James Fogarty y Scott E. Hudson [10] de la Carnegie Mellon University, por Jennifer Lai [12] de IBM Research, y por Horvitz [13] de Microsoft Research. Concluyen que aproximadamente el 65% del día la persona se encuentra en un estado de baja atención, tolerando así las interrupciones. De acuerdo a lo anterior, los teléfonos y ordenadores, quienes ingenuamente asumen que el usuario nunca está ocupado para recibir una llamada, leer un e-mail, o hacer click a un botón "OK", están probablemente en lo correcto las dos terceras partes del tiempo. Por lo cual, para que los sistemas considerados sean útiles deberán tener una precisión mayor al 65% detectando cuándo sus usuarios están cerca de sus límites cognoscitivos.

La precisión de tales sistemas puede aumentarse ampliando su software para que controle nuevos dispositivos hardware. Como lo demostraron Fogarty [11] y sus colaboradores elevando la precisión del sistema al 76%, cuando usaron un micrófono para detectar si alguien estaba sosteniendo una conversación. Cuando el grupo de Fogarty mejoró el software para detectar también movimientos del mouse, actividad del teclado y aplicaciones corriendo sobre las máquinas, la precisión del sistema alcanzó en promedio el 81%.

De otra parte, Bestcom/Telefonía ampliada, un prototipo basado en el trabajo de Horvitz, indaga más profundo dentro de la computadora de cada usuario para encontrar pistas sobre lo que ellos tienen programado hacer y lo que están haciendo. Microsoft lanzó una prueba beta interna de este sistema a mediados de 2003. La plataforma realiza una clasificación de las llamadas y los mensajes de correo entrantes tal como lo haría una recepcionista. El sistema, el cual corre sobre un ordenador central, analiza al emisor de la llamada contrastándolo con la lista de contactos, el directorio de la compañía, o el historial de llamadas recientes de la persona que recibe la llamada. Triangulando estas fuentes el sistema trata de deducir su relación: miembros de la familia, supervisores o gente a quien ha llamado durante el día, y decide si deja pasar la llamada o no, si decide no pasar la llamada, le ofrece a quien la realiza, dos opciones: dejar un mensaje de voz o re-programar la llamada para otro momento en que la agenda del emisor y del receptor estén libres (consultando para ello las agendas de los dos involucrados). Los mensajes de e-mail tienen un tratamiento similar. Cuando el empleado está fuera de la oficina, Bestcom automáticamente ofrece re-dirigir los mensajes de los emisores prioritarios a su teléfono móvil, a menos que su agenda y otra evidencia indiquen que se encuentra en una reunión. La plataforma de notificación de e-mail, decide cuándo y cómo notificar al receptor de la llegada de un nuevo mensaje. Para ello está divida conceptualmente en tres partes: la primera parte, estima el valor del mensaje y cómo éste decae con el tiempo, la segunda, usa sensores y modelos estadísticos para adivinar el foco y la intensidad de la atención del usuario, cuáles dispositivos está usando, cómo probablemente verá el mensaje sin una alerta y cómo estas variables cambiarán en el futuro. La tercera parte decide qué clase de alerta usar.

Implantar este tipo de tecnología en las empresas debería ser fácil, ya que muchas de las grandes empresas ya usan software para administrar los contactos, sistemas telefónicos y calendarios estándar computarizados. Sin embargo, no a todos los empleados les agrada la idea de tener micrófonos todo el tiempo en su oficina, o exponer su agenda a un programa que en últimas no controlan. Los investigadores conocen sobre estos riesgos. Hudson [15] argumenta que un sistema atento no debe grabar audio, golpes de teclado y similares, sino simplemente analizar el flujo de datos y descartarlos cuando se registran “conversaciones en progreso”, “mecanografía detectada”, etc.

Algunos expertos en el campo permanecen escépticos con respecto a la precisión que puede alcanzar un sistema considerado, porque los usuarios pueden tener muy baja tolerancia para un sistema que erróneamente suprime una de cada diez llamadas importantes. "Las cosas más 'atentas' llegan a ser, las más impredecibles" advierte Ben Shneiderman [17] de la Universidad de Maryland. "Tenemos una historia en esta comunidad de crear dispositivos 'inteligentes' que la gente no usa porque no puede entender cómo operan".

Entonces, ¿realmente la computación considerada reducirá las interrupciones y aumentará la productividad? Al menos para ciertas tareas especializadas, la respuesta es: indiscutiblemente. Como es el caso de la marina americana que cuenta con el sistema HAIL-SS (Human Alerting and Interruption Logistics-Surface Ship), el cual vigila a los marineros que están operando el sistema de armas navales Aegis y gestiona las muchas alertas que Aegis produce. En simulaciones de combate, HAIL-SS cortó el número de interrupciones de 50 a 80 por ciento, permitiendo a los marines manejar alertas críticas hasta dos veces más rápido. El software redujo la dificultad percibida y el estrés del trabajo en un 25%.

De momento los investigadores en las corporaciones y en los laboratorios académicos continuarán desarrollando y perfeccionando técnicas para permitir a las máquinas computarizadas estimar la carga cognitiva y el foco de atención de su usuario. Tecnología basada en sensores para detectar la atención y medir la distancia ha comenzado a aparecer en los coches, y esto conducirá a la aparición de más aplicaciones "consideradas" cada día.

3. Presto: una plataforma extensible para el desarrollo de procesos de negocio ubicuos

Los procesos de negocio pueden beneficiarse de una mayor integración de los elementos físicos en los sistemas de información. Con el uso de sensores, tareas como el préstamo de libros en una biblioteca no requiere más que tomar los libros de la estantería sin necesidad que intervenga un biblitecario. Consiguiendo así interconectar objetos del mundo real en lo que se conoce como Internet de las Cosas (Internet of Things). Los dispositivos móviles, dotados con tecnologías Auto-ID, desempeñan un papel clave en la reducción de la brecha entre el mundo físico y el mundo digital, su capacidad de identificación automática puede mejorar la participación del usuario en los procesos de negocio que involucran elementos físicos [2].

Sin embargo, las soluciones actualmente disponibles para conectar elementos físicos con servicios digitales no están adaptadas para apoyar tales procesos, debido a su limitada capacidad de personalización. Como respuesta a este problema se ha desarrollado Presto[7] [1], la cual es una plataforma extensible que soporta procesos de negocio en la Internet of Things, personalizable para dispositivos móviles que apoya la participación del usuario en procesos de negocio ubicuos. Su arquitectura puede ser extendida para proporcionar apoyo a nuevas tareas, tecnologías de identificación y fuentes de datos.

En muchos de los procesos en los que estamos inmersos participan objetos del mundo físico, pero con los sistemas basados en escritorio su presencia debe ser informada al sistema por parte de seres humanos, lo que es propenso a errores y poco eficiente. En este sentido, Presto le permite al usuario conectar procesos de negocio con objetos del mundo real en diferentes modos. Por un lado, Presto proporciona un administrador de tareas consciente de su contexto, es decir, le indica al usuario las tareas qué puede hacer de acuerdo a (1) su rol en el proceso, (2) el ambiente físico y (3) el estado del proceso. Por ejemplo, cuando un bibliotecario (rol de usuario) selecciona un libro (contexto físico) que ha sido retornado (estado del proceso), mecanismos de interacción son proporcionados para facilitar al usuario la ejecución de la tarea "Poner el libro en su estante". Por otro lado, Presto permite el uso de la interfaz de usuario y

[7] http://www.pros.upv.es/labs/projects/presto

elementos del mundo real para completar las tareas del proceso. Los datos demandados por una tarea se pueden proporcionar "tocando" el objeto físico cuya información asociada es requerida.

Presto está orientado a apoyar aplicaciones que soportan procesos de negocio caracterizados por una clara definición de sus actividades involucradas y su correspondiente coordinación. Está diseñado para ofrecer en todo momento la funcionalidad que el usuario necesita para su participación en el proceso de negocio ubicuo. Para conseguir esto, Presto proporciona soporte a (1) mecanismos de operación general que son útiles para cualquier proceso de negocio ubicuo, y (2) la posibilidad de personalización en orden a responder a las necesidades específicas de una aplicación particular. De la misma manera como los navegadores Web ofrecen conceptos comunes al usuario sin importar la página web que actualmente está siendo desplegada -tal como el botón Regresar o la opción de Agregar a favoritos-, Presto proporciona algunos modos de operación general y conceptos sin importar la tarea soportada en el proceso de negocio, permitiendo así la familiarización del usuario con el uso de la plataforma. Adicionalmente, para favorecer la personalización de las tareas, Presto permite a los usuarios (por ejemplo: bibliotecarios, miembros de la biblioteca, técnicos de mantenimiento, etc.) incorporar un conjunto de plug-ins a su plataforma para convertir la herramienta genérica en una específica para el cliente que apoya su perspectiva particular en el proceso de negocio.

3.1. Arquitectura independiente de la tecnología

La arquitectura de Presto está definida sobre el concepto de *componentes*. Los componentes son las piezas básicas de software que conforman el sistema. La funcionalidad de los componentes es descrita por medio de interfaces, y la comunicación entre ellos es asíncrona. Se optó por comunicación asíncrona dado que los procesos de negocio considerados son usualmente de larga ejecución e involucran participación humana.

La arquitectura de Presto está compuesta de un Task Manager, un Controller Component, y varios Task Processors, Identification Components y Data Providers. El desarrollo de una solución específica supone crear ciertos plug-ins e incorporarlos a la plataforma para extender su funcionalidad, de esta forma Presto puede ser extendido con soporte para nuevas tareas, tecnologías de identificación y fuentes de datos. La Figura 3.1 ilustra los componentes involucrados en la arquitectura de Presto y su conexión con el espacio digital y físico.

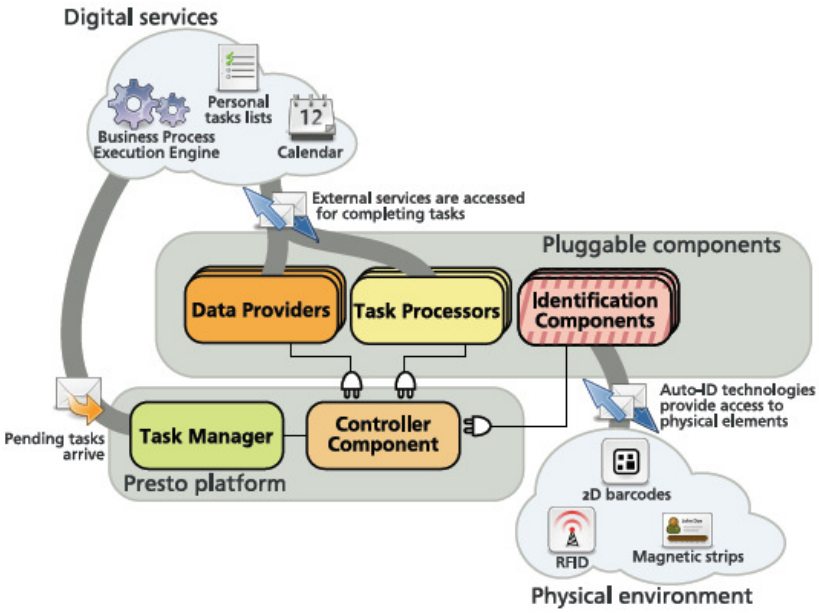


Fig. 3.1. Descripción de los componentes de la arquitectura

Task Manager. Este componente actúa como un buffer donde piezas de trabajo pendiente están esperando a ser completadas. Recibe mensajes de servicios externos tales como el sistema de información de una biblioteca y espera por los componentes necesarios para procesarlos. El Task Manager puede ser consultado en orden a recuperar mensajes de acuerdo a ciertos criterios –ejemplo: tareas de un cierto tipo, involucran un elemento físico dado o están dirigidas a un usuario específico-. El Task Manager tiene un papel central ofreciendo un vínculo flexible físico-virtual teniendo en cuenta la información manejada por este componente.

Task Processors. Estos componentes están encargados de soportar la extensibilidad de la plataforma en términos de la funcionalidad del negocio. Un Task Processor está definido para manejar un tipo particular de tarea –ej., prestar un libro de la biblioteca. Los Task Processors ofrecen a los usuarios la información requerida, los servicios y los mecanismos de interacción para completar la tarea.

Identification Components. Estos componentes están encargados de soportar la extensibilidad de la plataforma en el nivel tecnológico. Proporcionan mecanismos para accesar el entorno físico y transferir identificadores entre el espacio físico y digital por medio de alguna tecnología Auto-ID. Estos componentes ofrecen facilidades para capturar y generar identificadores con una tecnología específica.

Data Providers. Están encargados de transformar los identificadores proporcionados por los Identification Components en información que es relevante para el usuario. Los Data Providers están encargados de establecer dinámicamente la conexión entre los elementos físicos y su contraparte virtual. Así, ellos son la base para alcanzar un vínculo físico-virtual flexible. Un Data Provider está encargado de (1) determinar si un identificador corresponde a un elemento de una cierta clase, y (2) proporcionar información relativa al elemento.

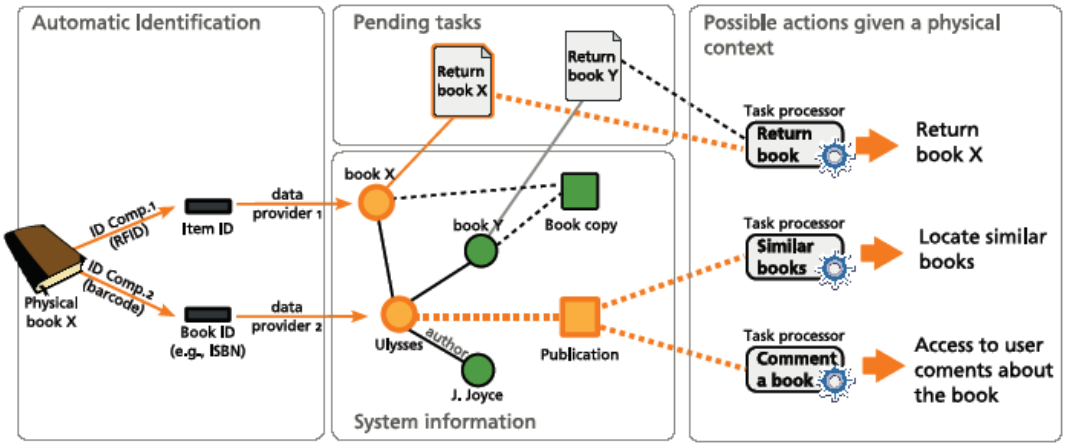


Fig. 3.2. Ejemplo del rol de identication components, data providers y task processors

La Figura 3.2 ilustra cómo los componentes introducidos interactúan cuando un elemento físico es detectado. En el ejemplo, dos Identification Components pueden proporcionar diferentes identificadores, los cuales son traducidos a diferentes contrapartes digitales usando un diferente Data Provider. El data provider A considera que el elemento físico es un libro particular –Book X- que ha sido prestado. El otro data provider considera al libro físico como la representación de un trabajo literario, al cual pertenecen libros como Book X y Book Y. Cada uno de estos objetos puede ser parte de una tarea pendiente, que disparará la activación de un Task Processor. En la figura, el libro detectado está pendiente de ser retornado, así el Task Processor encargado del retorno de libros es activado. Otros Task Processors que están asociados con la publicación digital del objeto pueden también ser activados. De este modo, podemos seleccionar la funcionalidad que es relevante para el usuario dado el contexto del proceso y el ambiente físico.

Controller Component. Está encargado de manejar la extensibilidad de la plataforma coordinando los componentes anteriormente mencionados. El Controller Component transfiere la información entre los diferentes componentes de la arquitectura. El modo en que la información es trasferida depende del modo de operación considerado. Presto soporta dos modos de operación general: modo dirigido por el objeto (object-driven mode) y modo dirigido por la tarea (task-driven mode).

En el modo dirigido por el objeto, el contexto físico es escaneado primero (sense) y las tareas relacionadas con él son presentadas al usuario. Por ejemplo: un usuario toca un libro con un dispositivo y los servicios asociados con este libro, tal como prestarlo, son presentados al usuario. De este modo, el usuario encuentra lo qué puede hacer en un contexto físico dado. En el modo dirigido por la tarea, el usuario explícitamente indica cuál tarea está ejecutando. El contexto físico en entonces accesado con el propósito de completar la tarea específica. Por ejemplo, el usuario decide prestar un libro y después selecciona el libro que quiere prestar.

Cuando se desarrolla algún sistema particular que sigue la arquitectura de Presto, algunos componentes son genéricos y pueden ser reusados, mientras que otros deben ser desarrollados para el dominio particular. En la actual arquitectura, el Task Manager y el Controller Component son genéricos, por tanto, ellos pueden ser usados en cualquier dominio. El resto de componentes (Task Processors, Identification Components y Data Providers) son definidos como plug-ins por la arquitectura, y pueden ser adicionados para personalizar la plataforma a las necesidades específicas del dominio.

En la siguiente sección se presenta la instanciación de los componentes arquitectónicos definidos sobre una plataforma concreta. En particular, se ha desarrollado Presto sobre Android.

3.2. Presto desplegado sobre Android

La plataforma móvil de Google, Android, se ha seleccionado para implementar los componentes arquitecturales de Presto, y así ofrecer una plataforma móvil que apoye procesos de negocio desarrollados en el contexto de la Internet de las Cosas. El desarrollo de Presto implica satisfacer los siguientes requisitos:

1. Ofrecer una interacción adecuada del usuario en el dispositivo móvil.
2. Definir un modelo extensible.
3. Integrar diferentes fuentes de datos.

Android permite cumplir satisfactoriamente cada requisito, por las razones dadas a continuación:

Interaccion del usuario. Android proporciona una serie de objetos para construir ricas e interactivas interfaces de usuario, dotadas de efectos visuales, animaciones, soporte para gráficos 3D usando OpenGL[8], soporte para interfaces multitáctiles, entre otros. Ofrece librerías

[8] OpenGL La librería usada para soportar gráficos 3D graphics basada en el Open GL ES 1.0 API

para manejar imágenes, video, y archivos de audio en múltiples formatos. Favoreciendo de esta manera el diseño de la interfaz gráfica de usuario y su adecuada interacción, consiguiendo de esta forma el nivel deseado de fluidez en el proceso de negocio considerado.

Extensión del modelo. La arquitectura de Android promueve el concepto de reuso de componentes, permitiendo publicar y compartir actividades, servicios y datos entre aplicaciones con las restricciones de seguridad impuestas por ellas. Cada aplicación consiste de componentes débilmente acoplados definidos usando un archivo XML denominado Manifest (AndroidManifest.xml); el cual describe la estructura, los metadatos de la aplicación, sus componentes y la forma cómo interactúan. A través de un framework de paso de mensajes el sistema operativo del dispositivo determina qué componente de qué aplicación será activado para desarrollar una determinada tarea.

Abstracción de datos. Uno de los componentes principales de Android es el Content Provider, éste es usado por la aplicación para compartir sus datos almacenados (local o remotamente) con otras aplicaciones. Lo cual significa que también nuestra aplicación puede usar los Content Providers expuestos por otras, para acceder a sus datos almacenados. De esta manera se ofrece un acceso homogéneo a los datos sin importar su fuente (XML, relacional o basada en archivos).

3.2.1 Mapping tecnológico: Presto sobre Android

En esta sub-sección se dan las guías generales que permiten implementar cada componente de Presto a través de la utilización de diferentes elementos de Android. A través de esta guía un desarrollador podrá determinar para cada componente de Presto, qué componentes de Android debe utilizar y cómo relacionarlos. La Sección 3.2.1.1 define una notación gráfica para representar los componentes de una arquitectura basada en Android. La Sección 3.2.1.2 presenta los elementos de la plataforma Android utilizados para realizar la implementación de los componentes arquitecturales de Presto.

3.2.1.1 Notación específica para Android

Para poder representar de una manera fácilmente entendible los diferentes componentes de una arquitectura basada en Android, hemos definido una notación gráfica que se usará a lo largo del trabajo. Esta notación simboliza los elementos involucrados en la implementación de la arquitectura de Presto, y los concernientes a la implementación de la aplicación que dará soporte al proceso de negocio y sus servicios. Actualmente no existe una notación específica para esta

plataforma, por lo que en aras de una mayor comprensibilidad la notación definida se basa en conceptos de otras notaciones ampliamente aceptadas (BPMN, UML).

La notación organiza los símbolos de acuerdo a las siguientes categorías: Components, Intents, Resources, y Recursos de Datos:

Components

Representa los componentes esenciales de Android:

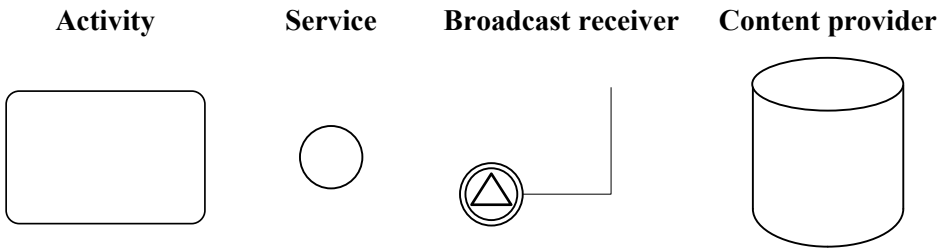


Fig. 3.3. Notación gráfica para representar los componentes Android

Intents

Representa los mensajes asíncronos, Intents e Intent filters, usados para activar a los componentes Activity, Service y Broadcast receiver:

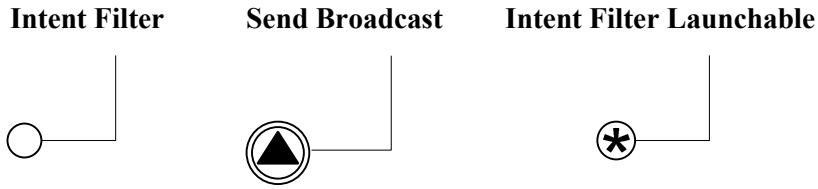


Fig. 3.4. Notación gráfica para representar los Intents Android

Resources

Representa los elementos externos que se incluyen y referencian dentro de una aplicación Android. Por ejemplo: imágenes, audio, video, cadenas de texto, layouts, temas, etc. Para simplificar la notación se agruparon los elementos en dos símbolos: Layout, que representa elementos del tipo layout, y Resource, que representa a los demás elementos.

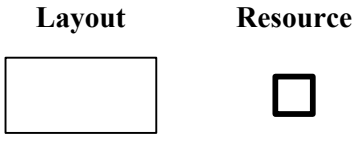


Fig. 3.5. Notación gráfica para representar los Resources Android

Recurso de Datos

Representa los contenedores físicos de información como pueden ser las bases de datos creadas en SQLite usando el Android API.

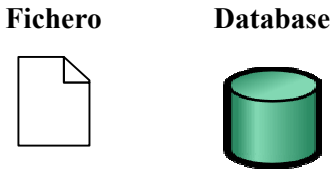


Fig. 3.6. Notación gráfica para representar los Recursos de Datos en Android

3.2.1.2 Implementación de Presto sobre Android

Como se indicó en la sección 3.1, la arquitectura de Presto está compuesta de un Task Manager, un Controller Component, y varios Task Processors, Identification Components y Data Providers. En esta sección se definen los elementos de la plataforma Android utilizados para realizar la implementación de los citados componentes. Se nombrará cada componente y a continuación los elementos de Android requeridos para su correcto despliegue.

- **Task Manager**

Es principalmente un repositorio para tareas pendientes, éstas son creadas, consultadas o actualizadas utilizando los métodos expuestos por el ContentProvider. La implementación de este componente comprende dos partes:

Interfaz de Usuario. Encargada de brindar la interfaz gráfica, proporcionando el contexto para el proceso de negocios y presentando los servicios que se ofrecen al usuario. Se implementó haciendo uso del componente Activity, el cual integra en su funcionalidad diferentes recursos (Resources –de diferente tipo: imagen, cadenas de texto, ícono, etc.- y Layouts –los archivos definidos en XML para crear la interfaz de usuario-), así como también objetos para manejar la interacción con el usuario. La Figura 3.7 ilustra gráficamente los componentes de la IU, denominada Presto View.

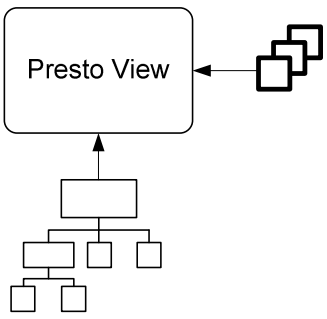


Fig. 3.7. Representación gráfica de la IU -Presto View-

Gestión de Tareas Pendientes (Pending Tasks)

La gestión de *Pendings tasks* se implementó mediante la definición de un Content Provider, el cual proporciona acceso a la base de datos desarrollada en SQLite permitiendo la administración de las tareas pendientes. Dicha base de datos es creada en el dispositivo móvil del usuario y consta de una tabla cuyos campos almacenan información relativa a la tarea. Esta información es proporcionada por el *Task Processor* que ordenó su creación.

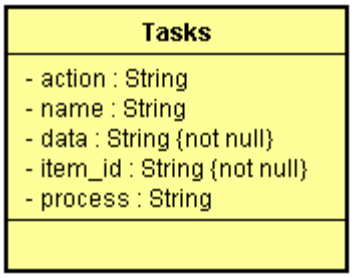


Fig. 3.8. Tabla Task de la base de datos Tasks_list.db

El Task Processor que ordene la creación de una tarea pendiente, debe enviar a Presto un mensaje (Intent) que conste de un action y de parámetros extra, los cuales están especificados en la Tabla 3.1. La columna "Valor" de esta tabla, es completada con la información suministrada por el Task Processor en el momento de creación de la tarea pendiente. En la Sección 4.2.3 se especifican los diferentes valores que de acuerdo al caso de estudio implementado se asignan a cada parámetro.

Parámetro	Tipo de dato	Descripción	Valor
action	String	Le indica a Presto que debe crear una nueva tarea pendiente (*Pending task*).	"es.upv.pros.presto.CREATE"
Extras			
presto.action	String	Contiene la llamada al Task Processor que debe ejecutar la tarea pendiente y así completarla.	
presto.name	String	Contiene el nombre de la tarea, éste será desplegado en la interfaz de usuario de Presto.	
presto.data	String	Contiene la información referente al objeto	

		identificado y sobre el cual se ejecutará la acción (presto.action).	
presto.item_id	String	Contiene el código único de identificación del objeto.	
presto.process	String	Se refiere al proceso al que pertenece el Task Processor.	

Tabla 3.1. Intent para la creación de una nueva Pending task

El conjunto de tareas es accesado por el Task Manager a través de la URI expuesta por el Content Provider: *content://es.upv.pros.provider.pending.Tasks/tasks*
Para accesar una tarea específica se usa su item_id correspondiente, por ejemplo:
*content://es.upv.pros.provider.pending.Tasks/tasks/{*item_id*}*

La Figura 3.9 ilustra gráficamente los componentes involucrados en la Gestión de Tareas Pendientes, siguiendo la notación definida para Android.

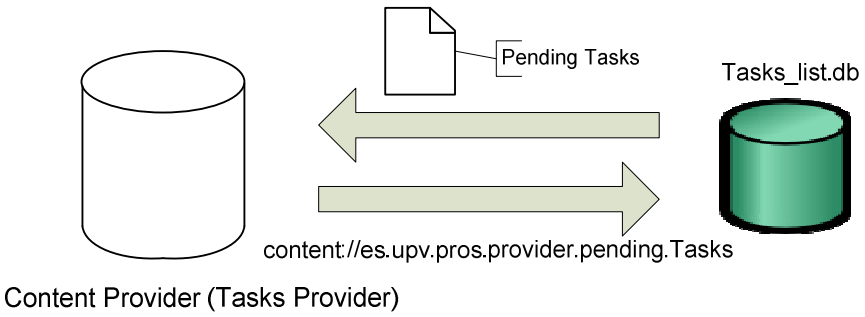


Fig. 3.9. Gestión de Tareas Pendientes (Pending Tasks)

- **Controller Component**

Encargado de manejar la extensibilidad de la plataforma coordinando los componentes: Task Processors, Identification Components y Data Providers. Todos ellos implementados a través de las aplicaciones que darán soporte al proceso de negocio.

Recordando lo mencionado en la Sección 3.1, Presto soporta dos modos de operación: modo dirigido por el objeto (object-driven mode) y modo dirigido por la tarea (task-driven mode). Para dar soporte a estos dos modos de operación, la aplicación Android sobre la cual se implementa la funcionalidad del proceso de negocio debe proporcionar la información requerida y en el formato especificado a la aplicación Android que implementa a Presto. A continuación se describe la información requerida y la forma cómo debe ser enviada:

Object-driven mode

En este modo, el contexto físico es escaneado (Sense) primero y las tareas que se pueden ejecutar sobre el objeto son presentadas al usuario. El usuario escoge este modo de operación seleccionando la opción Sense (en la Interfaz de Usuario de Presto), posteriormente realiza la identificación del objeto físico usando alguno de los *Identification Components* implementados, e inmediatamente Presto envía un mensaje en broadcast con la identificación del objeto a todas las aplicaciones que pueden escucharlo. De esta manera, la aplicación cuyo proceso de negocio tiene tareas que se pueden ejecutar sobre el objeto identificado enviará toda la información necesaria para que esas tareas puedan administrarse a través de Presto.

Fig. 3.10. Presto ejecutándose en Object-driven mode

Para que una aplicación funcione correctamente en este modo de operación debe satisfacer lo siguiente:

1. Para escuchar el broadcast enviado por Presto, debe incorporar en su archivo Manifest un Broadcast Receiver, escuchando peticiones del tipo: *"es.upv.pros.action.PROVIDERS"*. El Broadcast Receiver implementado debe extraer la identificación del objeto accesando el parámetro extra "*idElement*" del mensaje (Intent) recibido. La Figura 3.11 ilustra la sección del AndroidManifest.xml de la aplicación que implementa el Broadcast Receiver.

2. Proporcionar la información referente a la aplicación y al objeto identificado, a través de un mensaje (Intent) que conste de un parámetro action y de parámetros extra, de la siguiente manera:

Parámetro	Tipo de dato	Valor/Descripción
action	String	"es.upv.pros.presto.PROVIDER_RECEIVER"
Extras:		
name	String	Nombre de la Aplicación, corresponde a la descripción global del proceso de negocio implementado. En la Figura 3.10 se aprecia que el proceso lleva por nombre ***Pros Library***
domain	String	Corresponde al dominio de la aplicación que implementa la funcionalidad total del proceso de negocio.
exists	Boolean	Indica si el objeto recuperado (*idElement)* existe entre los objetos físicos que maneja el proceso de negocio implementado en la aplicación.
description	String	Es la descripción del objeto identificado. En la Figura 3.10 se aprecia que Pros Library encontró una correspondencia con el objeto suministrado, cuya descripción es: **Models 2009 Proceedings**
data	String	Corresponde a toda la información disponible sobre el objeto identificado.
element	ArrayList<String>	Corresponde al código único que identifica al libro. Es decir corresponde a *idElement.*

Tabla 3.2. Información suministrada por la aplicación que implementa el Proceso de Negocio

3. Enviar el mensaje (Intent) anteriormente descrito a través de un mensaje broadcast a Presto.

Fig. 3.11. Broadcast Receiver de la aplicación que implementa el proceso de negocio

Task-driven mode

En este modo de operación, el usuario primero indica la tarea que desea realizar y luego el contexto físico es accesado con el fin de completarla. La aplicación que implementa el proceso de negocio debe suministrar a Presto las tareas que el usuario puede llevar a cabo en dicho modo. Para lograr esto, la aplicación debe especificar en su archivo Manifest los componentes Activity (equivalentes a los *Task Processors* en la arquitectura de Presto) que implementarán tales tareas, agregando a la definición de la Activity los siguientes elementos:

Parámetro	**Valor/Descripción**
android:process	Se refiere al proceso al que pertenece el Task Processor.
android:taskAffinity	Se refiere al grupo de tareas a las que pertenece el Task Processor.
Intent filter	
android:label	Corresponde al nombre de la tarea (Task Processor) que será presentado al usuario en la IU de Presto.
action android:name	*"presto.pros.upv.es.task"* Indica que el Task Processor puede ejecutar esta tarea en modo Task-driven.
action android:name	Corresponde al nombre de la Activity que implementa el Task Processor, debe incluir también el paquete.
category android:name	*"presto.pros.upv.es.launchable"* Indica que el Task Processor lanzará esta tarea en modo Task-driven.
category android:name	*"android.intent.category.DEFAULT"* Presto sólo ejecuta tareas en este tipo de categoría.

Tabla 3.3. Elementos de la Activity que implementa un Task Processor en modo Task-driven

Los campos de la tabla 3.3 se resaltan en el siguiente ejemplo, correspondiente a la sección de código que ilustra la definición del Task Processor *BorrowBook,* en el archivo SmartLibrary Manifest.xml:

```
<activity android:name=".BorrowBook" android:label="@string/app_name"
      android:process="es.upv.pros.smartlib.loanprocess"
      android:taskAffinity="es.upv.pros.smartlib.book">
      <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
      <intent-filter android:label="Borrow Book">
         <action android:name="es.upv.pros.smartlib.BORROW"></action>
         <category android:name="android.intent.category.DEFAULT"></category>
      </intent-filter>
```

```
        <intent-filter android:label="Borrow Book">
            <action android:name="presto.pros.upv.es.task"></action>
            <action android:name="es.upv.pros.smartlib.BorrowBook"></action>
            <category android:name="presto.pros.upv.es.launchable"></category>
            <category android:name="android.intent.category.DEFAULT"></category>
        </intent-filter>
    </activity>
```

La Figura 3.12 presenta la estructura del archivo PrestoManifest.xml, en el que se aprecia la definición del Broadcast Receiver para escuchar los mensajes enviados por la aplicación que implementa el proceso de negocio, y el Content Provider para la gestión de las tareas pendientes (Pending tasks).

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.upv.pros.presto" android:versionCode="1"
    android:versionName="1.0">

    <application android:icon="@drawable/icon" android:label="@string/app_name">

        <provider android:name="es.upv.pros.provider.pending.TasksProvider"
            android:authorities="es.upv.pros.provider.pending.Tasks" />

        <receiver android:enabled="true" android:name="PrestoProviderReceiver">
            <intent-filter>
                <action android:name="es.upv.pros.presto.PROVIDER_RECEIVER"></action>
            </intent-filter>
        </receiver>

        <activity android:name=".Presto" android:label="@string/app_name"
            android:launchMode="singleTask">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter android:label="Complete Task">
                <category android:name="android.intent.category.DEFAULT"></category>
                <action android:name="es.upv.pros.presto.COMPLETE"></action>
            </intent-filter>
            <intent-filter android:label="Create Task">
                <category android:name="android.intent.category.DEFAULT"></category>
                <action android:name="es.upv.pros.presto.CREATE"></action>
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="3" />
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
</manifest>
```

Fig. 3.12. Presto Manifest.xml

Una vez definidos los componentes genéricos de la plataforma de Presto, se especificarán los componentes -Task Processors, Identification Components y Data Providers- que personalizan la plataforma a las necesidades específicas del dominio.

- **Task Processors**

En términos de la funcionalidad del negocio, un *Task Processor* debe definirse para manejar un tipo particular de tarea. Este componente proporciona a los usuarios la información requerida y los mecanismos de interacción necesarios para completar la tarea.

El *Task Processor* se implementa haciendo uso de componentes Activity para presentar la interfaz gráfica de la tarea, y componentes Service en caso que se necesite soportar tareas con funcionalidad ejecutándose de manera continua. Los mecanismos de interacción requeridos para completar una tarea se implementan haciendo uso del framework de mensajes (Intent e Intent Filters), que permiten la comunicación del *Task Processor* (Activity) con los demás componentes que conforman la plataforma de Presto.

La Figura 3.13 expresa gráficamente la definición de un *Task Processor* usando componentes Android.

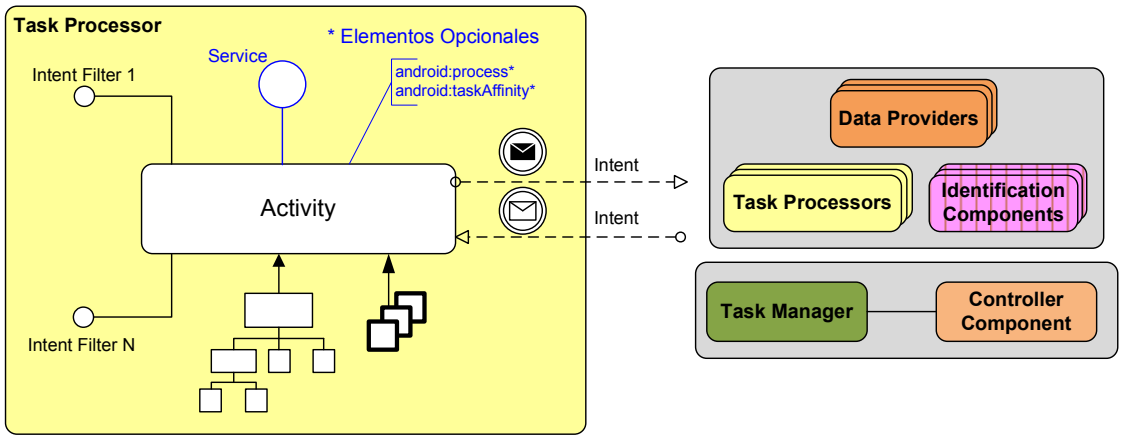


Fig. 3.13. Definición del Task Processor en Android

- **Identification Components**

Estos componentes proporcionan mecanismos para accesar el ambiente físico, y transferir identificadores de objetos entre el espacio físico y el digital por medio de alguna tecnología Auto-ID.

Cada Identification Component se desarrolla como una aplicación Android, la cual constará de una Activity para presentar información y opciones de interacción al usuario, también se puede usar en su implementación componentes Service en caso de ser necesarios. Es esencial que la Activity usada para implementar el Identification Component presente un Intent Filter cuyo elemento action sea igual a: "*es.upv.pros.SENSE*", de esta manera podrá recibir los mensajes

(Intents) enviados desde los Task Processors o desde el Controller Component, requiriendo su actuación.

Una vez identificado el objeto físico, el Identification Component devuelve su código único de identificación al componente que lo llamó, enviando de regreso el Intent con un parámetro extra:

Extra	
Parámetro	**Valor/Descripción**
es.upv.pros.interactions.elements	Contiene el código único de identificación del objeto.

Tabla 3.4. Identification Component, parámetro extra en el Intent

La Figura 3.14 expresa gráficamente la definición de un *Identification Component* usando componentes Android.

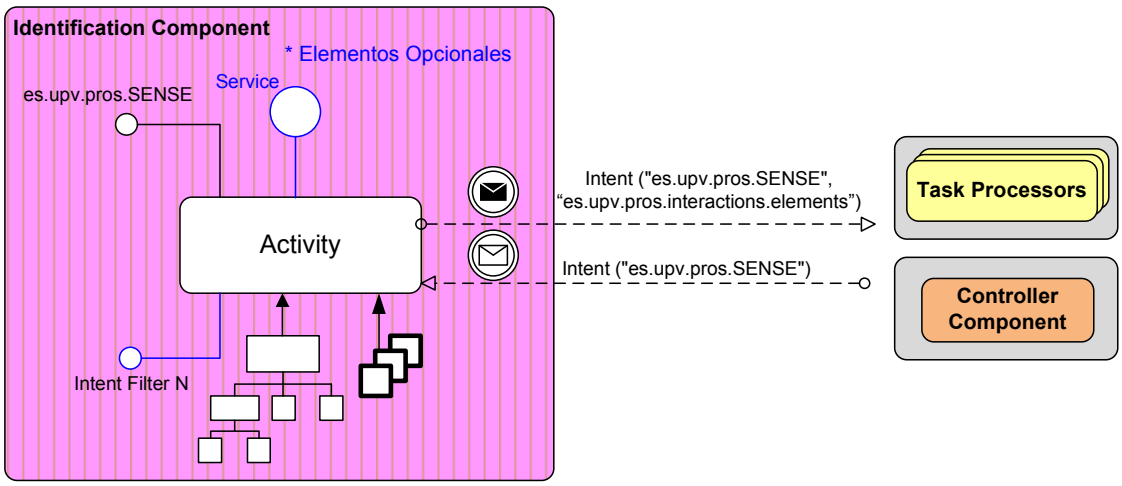


Fig. 3.14. Definición del Identification Component en Android

- **Data Providers**

Son los encargados de transformar los identificadores proporcionados por los Identification Components en información relevante para el usuario. Cada Data Provider representa una posible proyección de un elemento físico en el mundo digital, es por ello que para un mismo elemento físico se pueden definir uno o más Data Providers, los cuales brindarán información relativa al objeto de acuerdo a la perspectiva definida.

Los Data Provider se han implementado usando el componente Content Provider, el cual proporciona acceso homogéneo a la información exponiendo una URI única, dicha URI es direccionada por los Task Processors o por el Controller Component, para obtener información sobre un objeto especificado.

La Figura 3.15 expresa gráficamente la definición de un Data Provider usando componentes Android y accesando diferentes fuentes de información para establecer la relación entre el espacio digital y el físico.

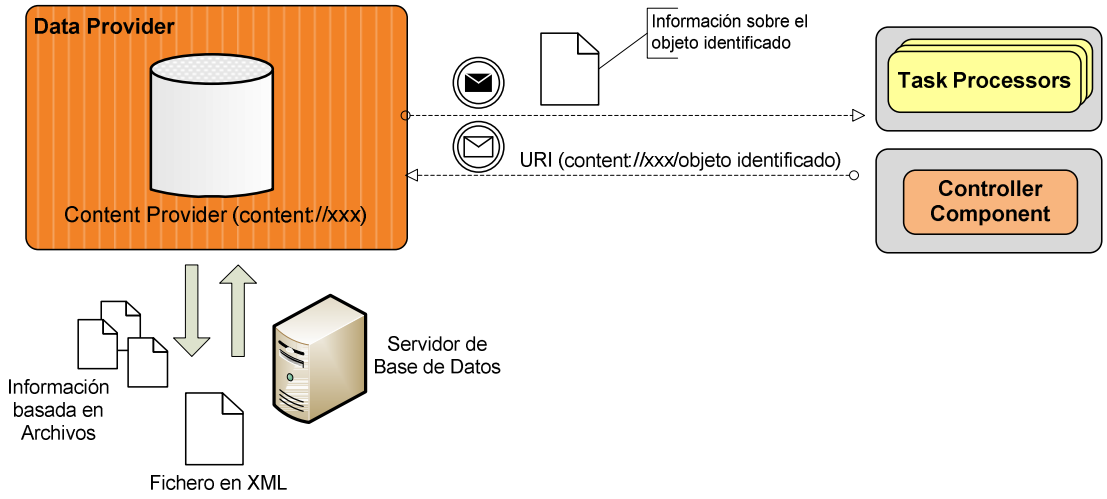


Fig. 3.15. Definición del Data Provider en Android

La Figura 3.16 ilustra gráficamente los diferentes componentes de la arquitectura de Presto traducidos a elementos de la arquitectura Android. Se puede apreciar como el Controller Component se puede implementar haciendo uso del archivo Manifest de la aplicación sobre la cual se implementa Presto, comunicándose con los demás componentes de la arquitectura a través de Intents y Broadcast Receivers. También se observa la equivalencia de los Task Processors e Identification Components, con las aplicaciones desarrolladas para soportar la respectiva funcionalidad.

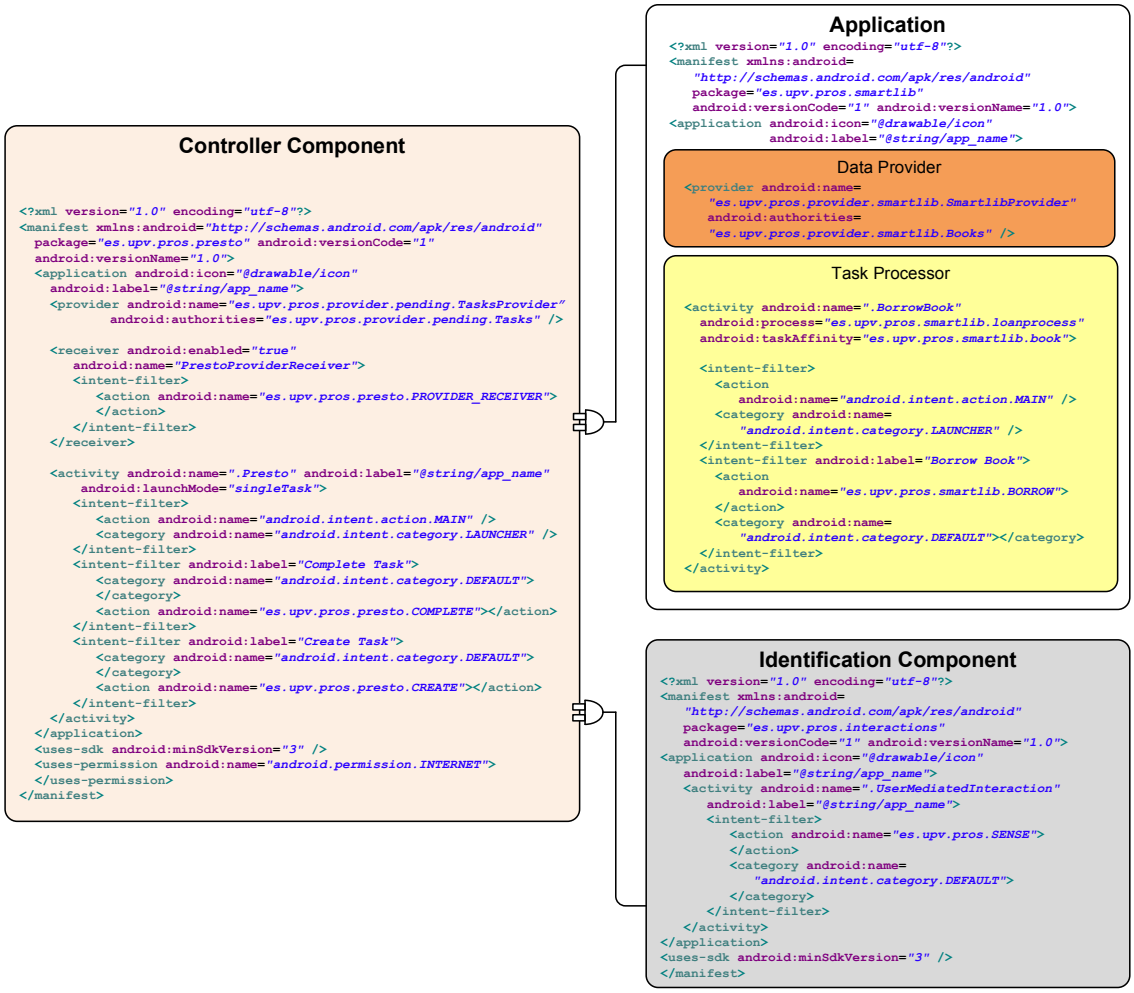


Fig. 3.16. Componentes de Presto implementados sobre Android

4. Desarrollo de procesos de negocio móviles adaptados a la Obtrusividad

Cada día los sectores comercial y académico invierten mayores recursos económicos en tecnología móvil que les permita realizar sus procesos de negocio de manera rápida y segura. Según el Institute for Business Value de IBM[9], la cantidad de usuarios de Internet móvil en el mundo se acerca a 1000 millones, y aproximadamente el 67% de todos los trabajadores hoy son usuarios de computación inalámbrica y móvil, esta situación impulsa el desarrollo de nuevo software y servicios para dispositivos móviles con características antes sólo disponibles en PCs. Las aplicaciones móviles que se desarrollan para apoyar los procesos de negocio se centran en proporcionar herramientas para agilizar la toma decisiones de negocio y al mismo tiempo, conectar a amigos, colegas de trabajo y equipos más allá de lo que era posible antes.

El éxito en el uso de las aplicaciones móviles depende, entre otros, de factores como el crecimiento en el número de dispositivos que pueden soportar aplicaciones de Internet móvil, el incremento en la velocidad de conexión de banda ancha inalámbrica, la mejora de interfaces de usuario (incluyendo mayor tamaño de pantalla), y de precios más atractivos para los servicios de Internet móvil. De la misma manera, el uso de aplicaciones móviles para apoyar procesos de negocio dependerá en gran medida del diseño adecuado de la interacción entre el usuario y el proceso, para conseguir la fluidez y la agilidad deseadas en este tipo de operaciones.

Siendo Presto una plataforma que apoya el desarrollo de procesos de negocio en dispositivos móviles a través de la utilización de tecnologías de identificación automática de objetos, es adecuado plantearse en orden a ofrecer una adecuada fluidez en el proceso, si la información se debe obtener de manera "sigilosa" o se le debe preguntar al usuario, si es necesario proporcionarle feedback o no, etc. Considerando lo anterior, el presente trabajo propone que los analistas del negocio elijan el nivel de obtrusividad deseado para la funcionalidad del proceso de negocio siguiendo el framework conceptual introducido en [3]. Dicho framework proporciona una guía para diferenciar las interacciones implícitas de las interacciones explícitas, y cuándo diferentes tipos de interacciones implícitas son útiles.

[9] http://www-935.ibm.com/services/us/index.wss/ibvstudy/gbs/a1029693?cntxt=a1000050

El framework de interacción implícita -ver Fig. 4.1- divide el espacio de posibles interacciones a lo largo de dos ejes: demanda atencional e iniciativa:

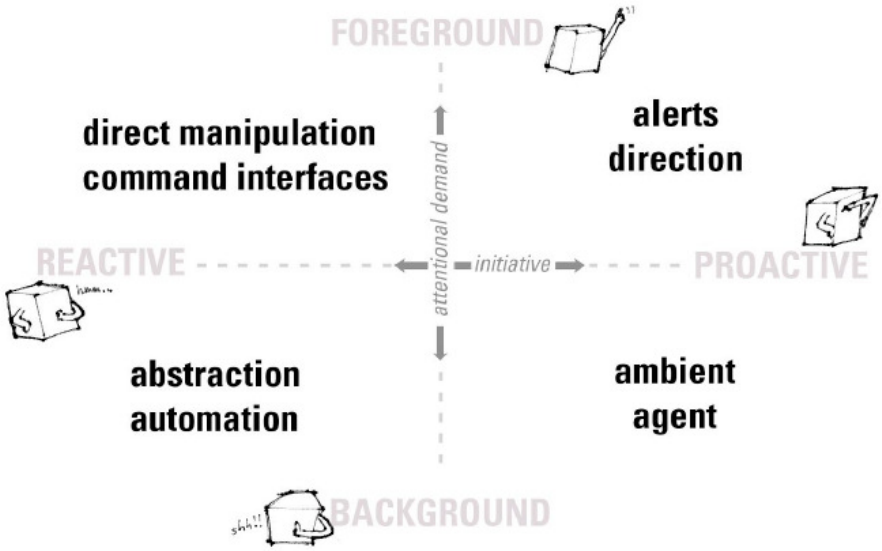


Fig. 4.1. Framework para la definición de interacciones implícitas. Basado en dos ejes, *iniciativa* (quién dispara la interacción) y *atención* (qué nivel de conciencia tiene el usuario respecto a la interacción).

Demanda atencional. Es el grado de carga cognitiva y perceptual impuesta por el sistema interactivo sobre el usuario. Las interacciones a nivel *Foreground* requieren un mayor grado de enfoque, concentración y conciencia, mientras que las interacciones *Background* son periféricas, tienen menos demanda cognitiva y pueden ocurrir en paralelo con otras interacciones.

Iniciativa. Es un indicador de cuánta presunción el sistema interactivo usa en la interacción. De esta manera, las interacciones que son iniciadas y conducidas por el usuario explícitamente son llamadas *interacciones reactivas*, mientras que las iniciadas por el sistema basadas sobre deseo o demanda inferida son *interacciones proactivas*.

Caracterizando las interacciones de este modo, se puede generalizar sobre las capacidades y características de todas las clases de interacciones en un modo independiente del dominio. Para aplicar el propósito planteado se ha seleccionado el caso de estudio Smart Library, el cual contempla el proceso de préstamo de libros en una biblioteca, para el que se definirá su funcionalidad a distintos niveles de obtrusividad siguiendo el framework para la definición de interacciones implícitas.

4.1. Definición del caso de estudio: Smart Library

El caso de estudio contemplado da soporte al proceso de préstamo de libros en una biblioteca, la cual tiene sus recursos etiquetados mediante tecnologías de identificación automática. De esta manera el usuario a través de su dispositivo móvil puede identificar el libro deseado y realizar sobre él las diferentes tareas consideradas en el proceso de negocio.

Para algunas tareas del proceso se definirán implementaciones alternativas. Estas implementaciones proporcionarán la misma funcionalidad pero a niveles de obtrusividad distintos. El caso de estudio será soportado por Presto implementado sobre la plataforma móvil Android.

4.1.1. Proceso de negocio

El proceso de negocio seguido en Smart Library considera a un usuario de la biblioteca que haciendo uso de su dispositivo móvil, y de las técnicas de Auto-ID incorporadas en él, identifica el libro deseado para realizar alguna de las siguientes tareas:

Prestar libro. Comprende los siguientes pasos:

1. Identificación del libro usando alguna de las técnicas Auto-ID soportadas para el caso de estudio.
2. Usando el identificador del libro se envía una consulta al catálogo de libros de la biblioteca, éste devuelve la información asociada al libro especificado.
3. Si el libro se encuentra disponible éste será prestado al usuario. Realizando las siguientes acciones:

Se modifica la información correspondiente al libro en el catálogo de libros de la biblioteca indicando su cambio de estado a "no disponible".

Se enviará un mensaje a Presto para que adicione la tarea "Retornar libro" a la lista de tareas pendientes del usuario.

Retornar libro. Es la tarea que completa el proceso iniciado por *Prestar el libro*, comprende los siguientes pasos:

1. El catálogo de libros de la biblioteca es accesado para actualizar el estado del libro a "disponible"
2. Se enviará un mensaje a Presto informándole que la tarea iniciada al Prestar el libro se ha completado y debe ser eliminada de la lista de pendientes.

Comentarios. Comprende los siguientes pasos:

1. Identificación del libro usando alguna de las técnicas Auto-ID soportadas para el caso de estudio.
2. Usando el identificador del libro se envía una consulta al catálogo de libros de la biblioteca, éste devuelve la información asociada al libro especificado.
3. El usuario podrá consultar los comentarios existentes o adicionar uno propio. Si se adiciona un nuevo comentario la siguiente acción se realizará:
    3.1. Se modifica la información correspondiente al libro en el catálogo de libros de la biblioteca adicionando un nuevo comentario.

Similares. Comprende los siguientes pasos:.

1. Identificación del libro usando alguna de las técnicas Auto-ID soportadas para el caso de estudio.
2. Usando el identificador del libro se envía una consulta al catálogo de libros de la biblioteca, éste devuelve la información asociada al libro especificado. Indicando al usuario los libros similares y su ubicación respectiva.

Bloquear libro. Permite bloquear por cierto tiempo un libro para seguir buscando otros. Comprende los siguientes pasos:

1. Identificación del libro usando alguna de las técnicas Auto-ID soportadas para el caso de estudio.
2. Usando el identificador del libro se envía una consulta al catálogo de libros de la biblioteca, éste devuelve la información asociada al libro especificado.
3. Si el libro se encuentra disponible éste será bloqueado. Realizando las siguientes acciones:
    3.1. Se modifica la información correspondiente al libro en el catálogo de libros de la biblioteca indicando su cambio de estado a “bloqueado”.
    3.2. Transcurrido el tiempo determinado de bloqueo, se modificará la información correspondiente al libro en el catálogo de libros de la biblioteca indicando su cambio de estado a “no bloqueado”.

La Figura 4.2 muestra el diagrama BPMN que representa el proceso de negocio contemplado en Smart Library.

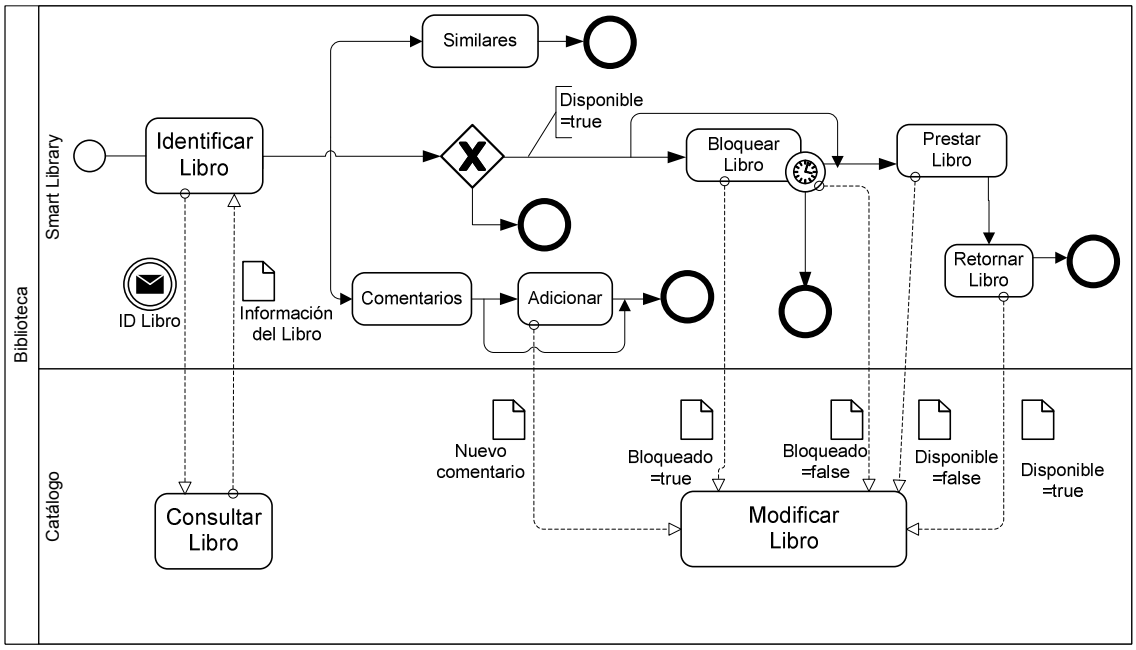


Fig. 4.2. Proceso de negocio, caso de estudio Smart Library

Los objetos físicos son identificados en este caso de estudio a través de diferentes medios. Los medios definidos son radio y papel (paper). La Fig. 4.3 representa una jerarquía de los diferentes medios empleados en el caso de estudio. El medio *papel* (paper) está especializado en dos medios dependiendo del uso de imágenes o letras para expresar el identificador. El medio *radio* ofrece identificación inalámbrica a diferentes distancias sin requerir línea directa de vista entre etiquetas y lectores.

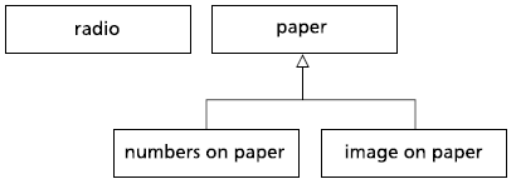


Fig. 4.3. Medios definidos para Smart Library

4.1.2. Tipos de Interacción

Con el objetivo de observar el proceso de negocio desde diferentes ángulos se han definido implementaciones alternativas para algunas de sus tareas. Estas implementaciones proporcionan la misma funcionalidad pero a niveles de obtrusividad distintos, situándolas para ello en la región del framework de interacciones implícitas que corresponda al tipo de interacción deseada -ver Fig. 4.4-.

De acuerdo con el framework presentado, las interacciones pueden llavarse a cabo en diferentes puntos del espacio definido por los ejes de iniciativa y atención. A continuación se detallan los

diferentes cuadrantes que determinan dichos ejes. Para ciertas tareas del proceso se han definido múltiples versiones en distintos niveles de obtrusividad.

4.1.2.1. Reactive/Foreground (RF)

Las interacciones tienen lugar en el plano atencional del usuario (Foreground) y las tareas del proceso se realizan por su mandato explícito (Reactive). El usuario tiene control total sobre las acciones a realizar y es informado sobre los resultados de éstas. Este tipo de interacción es apropiada cuando se necesita que el usuario esté consciente de cada acción que realiza para completar las tareas del proceso, requiriéndose así que el usuario tenga buen conocimiento del mismo. [4]

Serán implementadas en esta región las tareas Prestar libro, Retornar libro, Bloquear libro, Similares, y Comentarios. Las tareas presentarán interfaz gráfica interactiva para permitir el control del usuario sobre las acciones involucradas en el proceso, los resultados de sus acciones serán notificados proporcionandole así feedback.

4.1.2.2. Reactive/Background (RB)

La interacción ocurre en respuesta a acciones del usuario o a estímulos externos, pero el feedback es generalizado o se oculta del usuario (abstracción). Este tipo de interacción es apropiada para ahorrarle detalles sobre una tarea al usuario, también es apropiada para ejecutar tareas rutinarias que necesitan poca supervisión del usuario (automatización). [4]

Serán implementadas en esta región las tareas Prestar libro y Retornar libro. Las tareas presentarán interfaz gráfica para permitir que el usuario inicie la tarea o proporcione información básica para continuar el proceso. Los resultados de sus acciones no serán notificados al usuario.

4.1.2.3. Proactive/Foreground (PF)

Las interacciones tienen lugar en el plano atencional del usuario, pero son iniciadas por el sistema (Proactive). El sistema puede proporcionar información no solicitada explícitamente (alertas) o guiar la interacción instruyendo al usuario sobre lo que debe hacer (dirección). [4]

Serán implementadas en esta región las tareas Prestar libro, Retornar libro, Similares, y Comentarios. El sistema actuará proactivamente de acuerdo a la tarea seleccionada (en el caso de Prestar libro o Retornar libro), analizando si se cumplen las condiciones para realizarlas y notificando al usuario sobre su realización (feedback) de manera no obtrusiva. Se emplearán

alertas para comunicar al usuario que existen libros similares o comentarios negativos acerca del libro identificado en el proceso.

4.1.2.4. Proactive/Background (PB)

En este tipo de interacciones el sistema anticipa lo que debe hacer y ejecuta las acciones con poca o ninguna supervisión del usuario. Estas interacciones son adecuadas para tareas en las que el costo del error es bajo, también pueden emplearse para ejecutar tareas críticas que el usuario es incapaz de hacer (como por ejemplo alertar a la policía cuando un intruso penetra en su casa). [4]

Será implementada en esta región la tarea Prestar libro. El sistema actuará proactivamente respecto a la tarea seleccionada analizando si se cumplen las condiciones para realizarla, el resultado de su realización no será notificando al usuario.

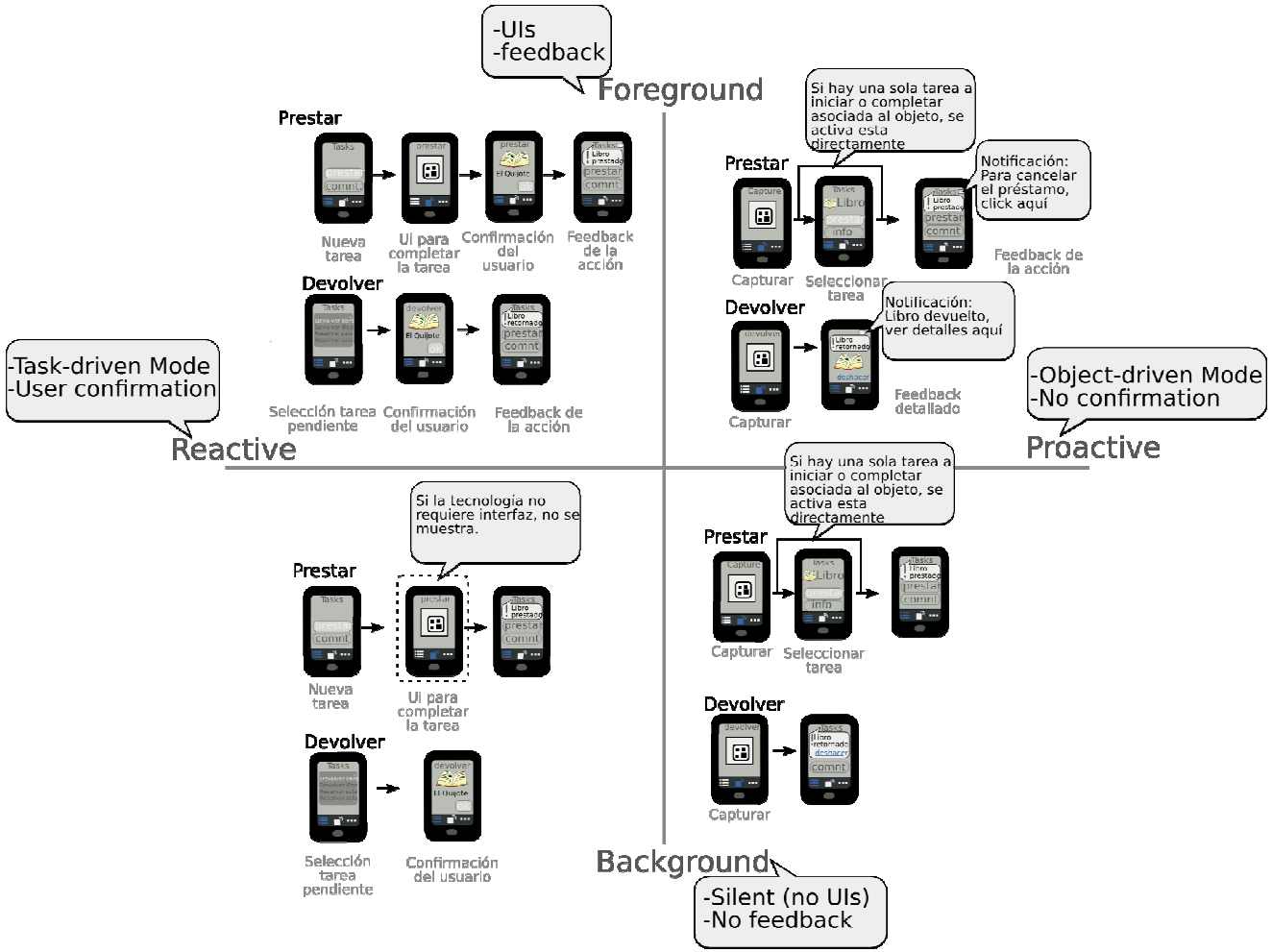


Fig. 4.4. Tareas de Smart Library a diferentes niveles de obtrusividad

4.1.3. Estructura de datos

La información manejada en el caso de estudio Smart Library incluye diferentes objetos del sistema. Se ha definido la estructura de información para los libros (clase Books), los comentarios (Comments) y los autores (Authors). El correspondiente diagrama de clases se muestra en la Fig. 4.5. Cuando un libro es prestado -lend()- o regresado -return()-, se actualiza el valor de su campo "available" a false o true respectivamente, de la misma forma cuando es bloqueado -block()-, se actualiza el valor de su campo "blocked" a true, volviendo a ser actualizado a false cuando transcurre el tiempo del bloqueo. También se considera información relativa a libros similares.

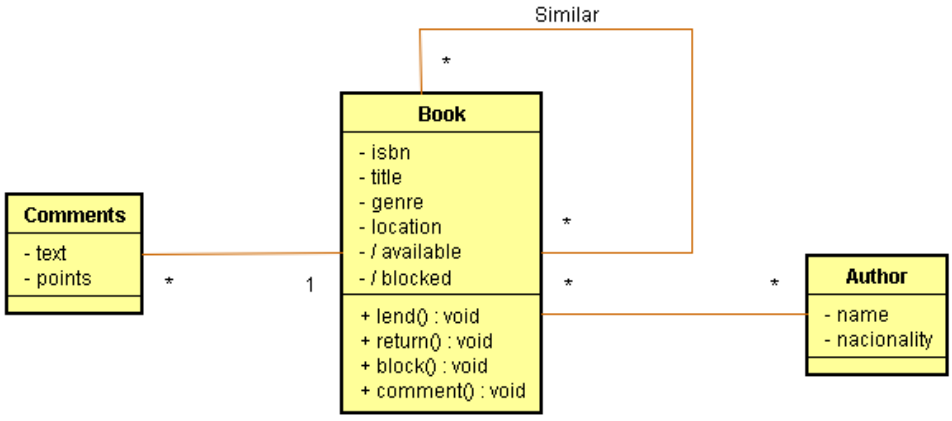


Fig. 4.5. Modelo de datos para Smart Library

Se han considerado sólo las clases y los atributos requeridos para la implementación de las tareas comprendidas en el caso de estudio. En un escenario real, información sobre el miembro de la biblioteca que realiza las diferentes operaciones sobre los libros deberá ser tenida en cuenta, igualmente podría definirse más información sobre los libros y otra clase de recursos.

4.2. Implementación del caso de estudio Smart Library sobre Presto

Esta sección comprende la descripción detallada de la implementación del caso de estudio Smart Library definido en la Sección 4.1. Para la implementación de Smart Library se llevó a cabo el desarrollo de tres Identification Components, un Data Provider, y varios Task Processors correspondientes a las tareas implementadas en las regiones del Framework de Interacciones Implícitas de referencia. A continuación se describen cada uno de los mencionados componentes.

4.2.1. Identification Components

El caso de estudio considera que la biblioteca mantiene sus libros identificados usando tecnologías Auto-ID, las tecnologías de identificación empleadas son: códigos QR, identificación por radio frecuencia RFID, y como medida de respaldo en caso que la tecnología falle los libros también se identifican adjuntándoles una etiqueta en papel con su código alfanumérico impreso.

4.2.1.1. QR Code Reader

La aplicación desarrollada hace uso de la cámara digital incorporada en el dispositivo móvil para realizar la captura del código QR que identifica el libro. Dicho código es decodificado usando la librería ZXing[10]. Se escogió esta librería dado que es open source y está disponible para diferentes dispositivos, soportando las ediciones estándar y móvil de Java, iPhone, y la plataforma Android.

La Figura 4.6 ilustra la implementación del componente QR Code Reader usando la notación gráfica definida para Android.

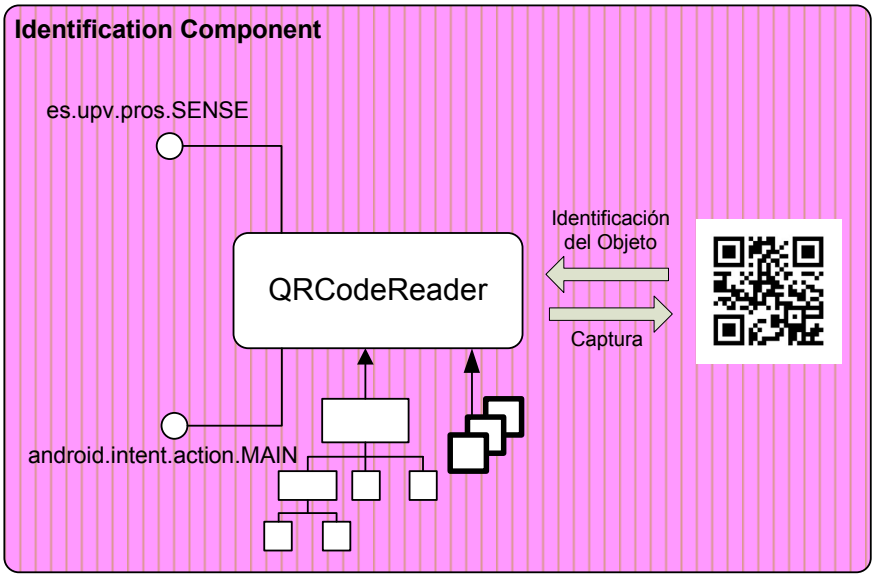


Fig. 4.6. QR Code Reader – Identification Component

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación, en el cual se resaltan los elementos ilustrados en la representación gráfica:

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
   <activity android:name=".QRCodeReader"
        android:label="@string/app_name">
        <intent-filter>
```

[10] http://code.google.com/p/zxing/

```
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        <intent-filter android:label="QR Code Reader">
            <action android:name="es.upv.pros.SENSE"></action>
            <category android:name="android.intent.category.DEFAULT">
            </category>
        </intent-filter>
    </activity>
</application>
```

4.2.1.2. RFID Reader

Debido a que el dispositivo móvil no dispone de capacidades RFID, la aplicación desarrollada ha simulado su uso aplicando técnicas de Wizard of Oz. Estas consisten en que un operario envía remotamente los eventos de identificación para que el usuario pueda experimentar el funcionamiento del RFID sin requerir de dicha tecnología. Para ello la aplicación ha definido un componente Service, el cual está encargado de capturar la identificación del libro cuando el evento de identificación es enviado por el operario. De esta forma la aplicación puede retornar la identificación del objeto físico al componente que lo está requiriendo.

La Figura 4.7 ilustra la implementación del componente RFID Reader usando la notación gráfica definida para Android.

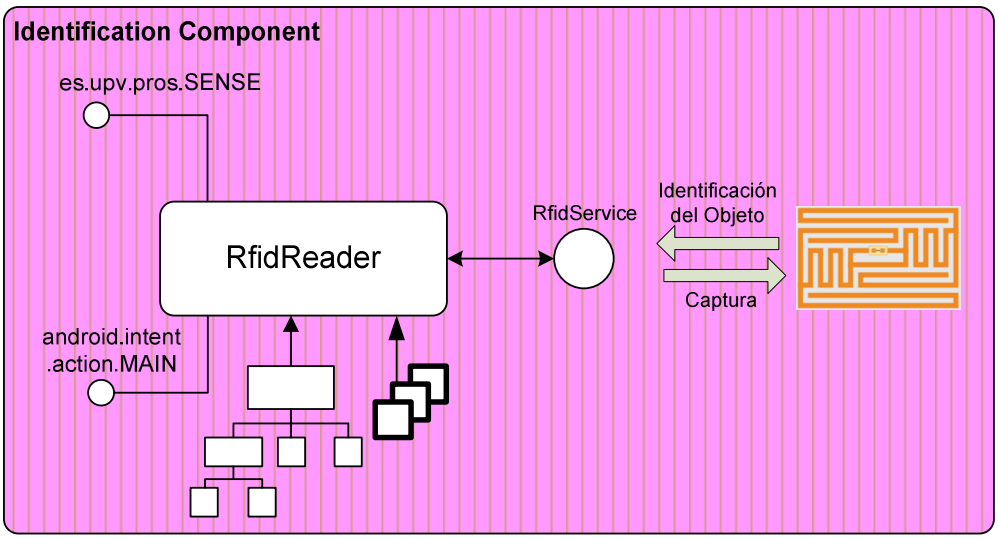


Fig. 4.7. RFID Reader – Identification Component

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación, en el cual se resaltan los elementos ilustrados en la representación gráfica:

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".RfidReader" android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
```

```
        </intent-filter>
        <intent-filter android:label="RFID Reader">
          <action android:name="es.upv.pros.SENSE"></action>
          <category android:name="android.intent.category.DEFAULT"></category>
        </intent-filter>
      </activity>
      <service android:name=".RfidService" android:label="Service">
      </service>
    </application>
```

4.2.1.3. User Mediated Interaction (UMI)

La aplicación mediante su interfaz gráfica solicita la identificación del libro al usuario, el cual la introduce haciendo uso del teclado del dispositivo móvil. La Figura 4.8 ilustra la implementación del componente UMI usando la notación gráfica definida para Android.

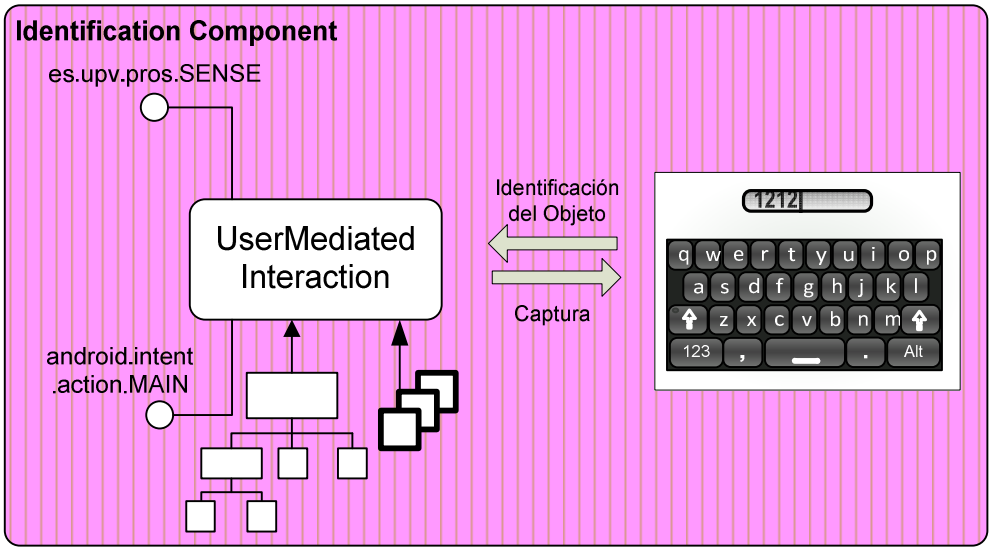


Fig. 4.8. User Mediated Interaction – Identification Component

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación, en el cual se resaltan los elementos ilustrados en la representación gráfica:

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
      <activity android:name=".UserMediatedInteraction"
        android:label="@string/app_name">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        <intent-filter>
          <action android:name="es.upv.pros.SENSE"></action>
          <category android:name="android.intent.category.DEFAULT"></category>
        </intent-filter>
      </activity>
    </application>
```

Para implementar la funcionalidad del caso de estudio se desarrolló una aplicación denominada: **SmartLibrary,** en la cual se definen los componentes Data Provider y Task Processors

necesarios para desarrollar las tareas involucradas en el proceso de negocio. La Sección 4.2.2 presenta la especificación del componente Data Provider. La Sección 4.2.3 presenta la especificación de los Task Processors definidos para realizar cada una de las tareas clasificándolos en la región respectiva del Framework de referencia.

4.2.2. Data Provider

Para realizar su implementación se definió un Content Provider al cual se le denominó: **SmartlibProvider.** El SmartlibProvider permite modificar o consultar información referente a un libro, para esto debe accesar al sistema de información de la biblioteca (Catálogo de libros) y mediante un servicio web recuperar o modificar la información correspondiente al libro dado.

El servicio web se desarrolló usando la edición para Android de Restlet[11]. Se escogió Restlet porque ofrece un conjunto de librerías java, de fácil uso y de código libre, basadas en REST. REST (Representational State Transfer.) define un estilo de arquitectura software comúnmente usado en el desarrollo de servicios web sin estado. En REST cada servicio es visto como un recurso (resource) identificado por una URL[5]. REST es fácil de entender, todo lo que se debe hacer es exponer un servicio web en la forma de una URL. Los usuarios pueden entonces consultar esta URL, a través de métodos http como get y post. Debido a esta facilidad, actualmente muchas compañías incluyendo Amazon y Yahoo, están exponiendo sus servicios web en la forma de recursos (resources) REST. En nuestro caso, la URL expuesta por el servicio desarrollado es: http://IP_SERVIDOR:8185/catalogue/books/bookid. El campo IP_SERVIDOR es la IP o nombre de dominio en dónde reside el servicio, y el campo bookid corresponde a la identificación del libro.

Los métodos implementados en el SmartlibProvider son accesados por los demás componentes de la aplicación, y por otras aplicaciones, usando la URI que expone para este propósito: content://es.upv.pros.provider.smartlib.Books. De esta manera un Task Processor puede consultar o modificar información relativa a un libro. El SmartlibProvider actúa de la siguiente manera:

1. El Task Processor o el componente que lo requiere, realiza una llamada al método adecuado (query o update) del SmartlibProvider.
2. De acuerdo al tipo de petición, el SmartlibProvider actuando como cliente del servicio web implementado hace peticiones a su URL para consultar o modificar el catálogo de libros de la biblioteca y de esta manera recuperar información de un libro particular o modificarla.

[11] http://www.restlet.org/

3. El SmartlibProvider re-trasmite la información solicitada al componente que hizo la llamada.

La Figura 4.9 ilustra la implementación del Data Provider (SmartlibProvider) usando la notación gráfica definida para Android y la representación gráfica del componente Restlet implementado para dar soporte al servicio web.

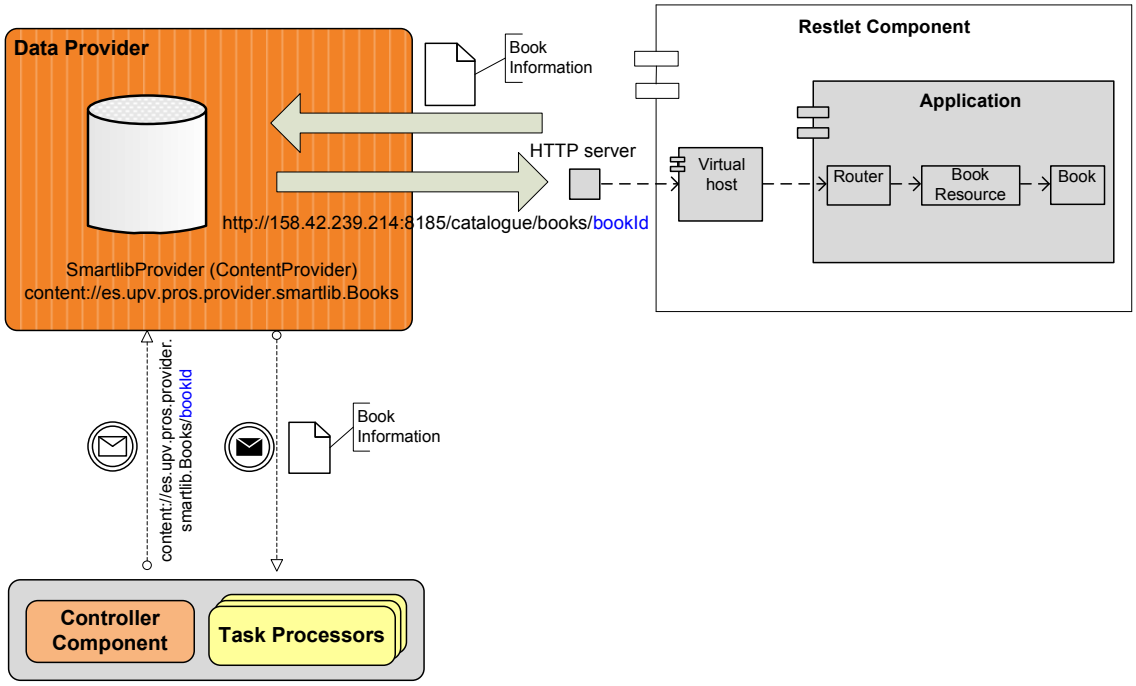


Fig. 4.9. Data Provider implementado para Smart Library

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación SmartLibrary, en el cual se resalta el componente que implementa la funcionalidad del Data Provider:

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
   <provider android:name="es.upv.pros.provider.smartlib.SmartlibProvider"
         android:authorities="es.upv.pros.provider.smartlib.Books" />
   . . .
</application>
```

4.2.3. Task Processors

Como se indicó en la Sección 4.1, las tareas consideradas en el proceso de negocio del caso de estudio Smart Library, son las siguientes: Prestar libro, Retornar libro, Comentarios, Similares, y Bloquear libro. Con el propósito de observar el proceso de negocio desde diferentes perspectivas cada tarea se definió a distintos niveles de obtrusividad, clasificándolas en la región del framework de interacciones implícitas correspondiente -ver Fig. 4.4-. Diseñando la interacción de cada tarea de acuerdo a las características del tipo de interacción deseada, especificadas en la Sección 4.1.2.

Para soportar lo anterior, se implementó un Task Processor para cada tarea en el nivel de obtrusividad deseado. La Sección 4.2.3.1 especifica el Task Processor que da soporte a la tarea "Prestar libro". La Sección 4.2.3.2 especifica el Task Processor que da soporte a la tarea "Retornar libro". La Sección 4.2.3.3 especifica el Task Processor que da soporte a la tarea "Comentarios". La Sección 4.2.3.4 especifica el Task Processor que da soporte a la tarea "Similares". La Sección 4.2.3.5 especifica el Task Processor que da soporte a la tarea "Bloquear libro". Detallando en cada sección el tipo de interacción requerida de acuerdo a la obtrusividad deseada.

4.2.3.1. Task Processors implementados para dar soporte a la tarea "Prestar Libro"

La tarea "Prestar Libro" posibilita el préstamo de libros a los usuarios de la biblioteca del caso de estudio considerado. El Task processor que implementa esta funcionalidad se comunica con los Identification Components definidos, para realizar la identificación del libro dado a través del dispositivo móvil del usuario. Usando el correspondiente Data Provider, accesa los servicios digitales de la biblioteca en orden a recuperar la información relacionada al libro identificado y registrar el respectivo préstamo. Proporcionando los mecanismos de interacción adecuados para completar la tarea.

"Prestar Libro" se implementó en Android utilizando principalmente los siguientes componentes:

- ✓ Activity. Presenta la interfaz gráfica de usuario, desplegando información referente al libro seleccionado y proporcionado la interactividad requerida para ejecutar la tarea. Este componente soporta la funcionalidad básica del negocio.
- ✓ Intent Filter. Maneja el paso de mensajes (comunicación) entre el Task Processor y los demás componentes de la plataforma. Los intent filter básicos para el adecuado funcionamiento de la tarea son:

Intent Filter		
Parámetros	**Valor**	**Descripción**
action android:name	"es.upv.pros.smartlib.BORROW"	Inicia la tarea "Prestar Libro" en el momento que recibe de los otros componentes de la plataforma un mensaje (Intent) con dicha action como parámetro.
category android:name	"android.intent.category.DEFAULT"	
Intent Filter		
android:label	"Borrow Book"	Corresponde al nombre de la

		tarea (Task Processor) que será presentado al usuario en la IU de Presto.
action android:name	"presto.pros.upv.es.task	Este conjunto de parámetros indica a Presto que la tarea se puede ejecutar en el modo task-driven -ver Tabla 3.3-. Por tanto, cuando se ejecuta la aplicación "Presto" en el dispositivo móvil del usuario, esta tarea es desplegada como una de las Tareas de inicio que se pueden realizar.
action android:name=	"es.upv.pros.smartlib.BorrowBookXX[12]"	
category android:name	"presto.pros.upv.es.launchable"	
category android:name=	"android.intent.category.DEFAULT"	

- ✓ Notification[13]. Es usado para informar el resultado de la ejecución de la tarea al usuario.

La tarea "Prestar Libro" da origen a una tarea pendiente (Pending Task), la cual es terminada cuando se ejecuta la tarea que implementa la funcionalidad requerida, en nuestro caso, cuando se ejecuta "Retornar Libro". Para crear una tarea pendiente, el Task Processor "Prestar Libro" envía un mensaje (Intent) a Presto -exactamente al Task Manager- con los parámetros introducidos en la Tabla 3.1, completándolos con la información correspondiente al Task Processor y a la tarea realizada -ver Tabla 4.1-.

Parámetro	Tipo de dato	Descripción	Valor
action	String	Le indica a Presto que debe crear una nueva tarea pendiente (*Pending task*).	"es.upv.pros.presto.CREATE"
Extras			
presto.action	String	Contiene la llamada al Task Processor que debe ejecutar la tarea pendiente y así completarla.	"es.upv.pros.smartlib.RETURN"
presto.name	String	Contiene el nombre de la	"Return " + {título del libro}

[12] BorrowBookXX, indica el nombre de la Activity que implementa el Task Processor en el nivel de obtrusividad correspondiente. Por ejemplo: BorrowBookRF, implementa el Task Processor en la región Reactive Foreground (RF).

[13] El componente Notification es usado por el Task Processor dependiendo del nivel de obtrusividad deseado. Por esta razón los Task Processors implementados en la región Proactive/Reactive Background (PB/RB) no requieren de su uso.

		tarea, éste será desplegado en la interfaz de usuario de Presto.	
presto.data	String	Contiene la información referente al objeto identificado y sobre el cual se ejecutará la acción (presto.action).	Información relativa al libro. La aplicación Smartlibrary organiza la información referente al libro (ISBN, título, autor, etc.) dentro de un objeto JSON[14].
presto.item_id	String	Contiene el código único de identificación del objeto.	Corresponde al código único que identifica al libro.
presto.process	String	Se refiere al proceso al que pertenece el Task Processor.	"es.upv.pros.smartlib.loanprocess"

Tabla 4.1. Parámetros para la creación de una nueva Pending task

La Figura 4.10 ilustra la implementación del Task Processor -Prestar Libro- usando la notación gráfica definida para Android. En ella se ilustran los diferentes Intent Filters requeridos para soportar la funcionalidad de la tarea, así como también otros elementos que describen la Activity que implementa el Task Processor.

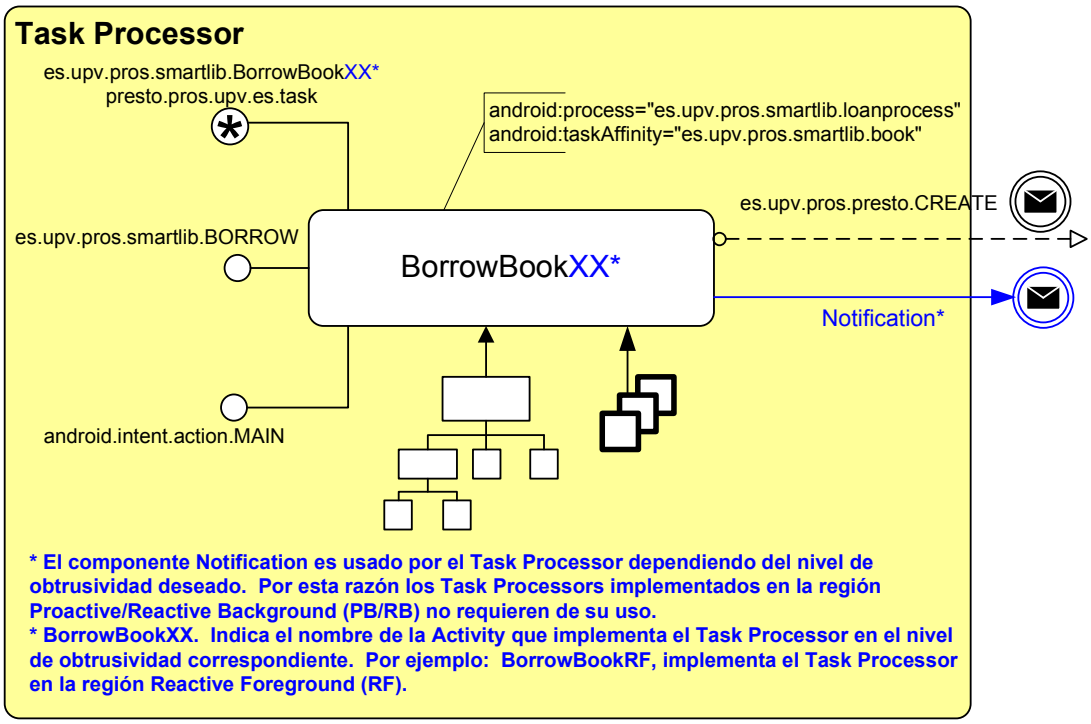


Fig. 4.10. Task Processor "Prestar Libro" implementado para Smart Library

[14] JSON es un formato para intercambio de datos, el cual estructura la información como una colección de parejas nombre/valor, proporcionando métodos para extraer y poner información correspondiente a cada campo de manera sencilla en el objeto. Igualmente suministra, entre otros, un método para convertir el objeto completo a String. Más información se obtiene consultando la página del proyecto: http://www.json.org.

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación Smartlibrary, en el cual se resaltan varios de los componentes que implementan la funcionalidad del Task Processor ilustrado en la Figura 4.10. La definición del Task Processor del ejemplo corresponde a la tarea "Prestar libro" implementada en la región Reactive Foreground (RF):

```
<activity android:name=".BorrowBookRF" android:label="@string/app_name"
     android:process="es.upv.pros.smartlib.loanprocess"
     android:taskAffinity="es.upv.pros.smartlib.book">
     <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
     </intent-filter>
     <intent-filter android:label="Borrow Book">
        <action android:name="es.upv.pros.smartlib.BORROW"></action>
        <category android:name="android.intent.category.DEFAULT"></category>
     </intent-filter>
     <intent-filter android:label="Borrow Book">
        <action android:name="presto.pros.upv.es.task"></action>
        <action android:name="es.upv.pros.smartlib.BorrowBookRF"></action>
        <category android:name="presto.pros.upv.es.launchable"></category>
        <category android:name="android.intent.category.DEFAULT"></category>
     </intent-filter>
  </activity>
```

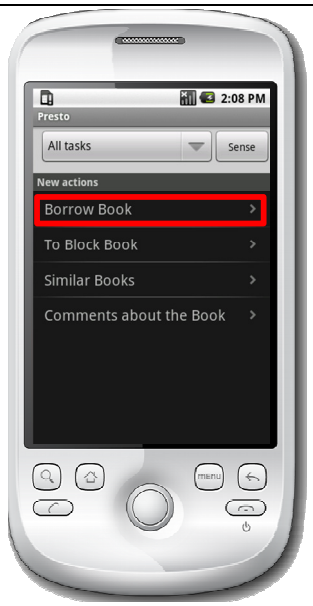
A continuación se detalla el tipo de interacción requerida para la tarea "Prestar Libro", de acuerdo a la obtrusividad deseada:

Prestar Libro en la región Reactive Foreground: Prestar Libro RF

La tarea involucra las siguientes acciones:

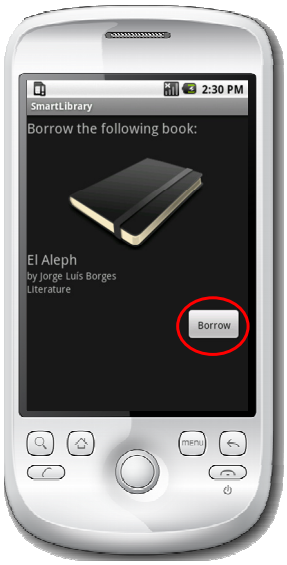
1. El usuario mediante el dispositivo móvil realiza la identificación del libro que desea prestar. Para ello hace uso de alguno de los Identification Components definidos en la Sección 4.2.1
2. La interfaz de usuario despliega información relevante al libro seleccionado. Si el libro no se encuentra disponible en el momento, el usuario es informado de esta situación y se le ofrece la posibilidad de seleccionar otro.
3. Si el libro está disponible, se solicita al usuario que confirme su intención de prestarlo.
4. Una vez realizado el préstamo, se envía una notificación al dispositivo móvil del usuario indicando datos relacionados con el préstamo: la hora, el título del libro, entre otros.

La Figura 4.11 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea "Prestar Libro" en la región Reactive Foreground:

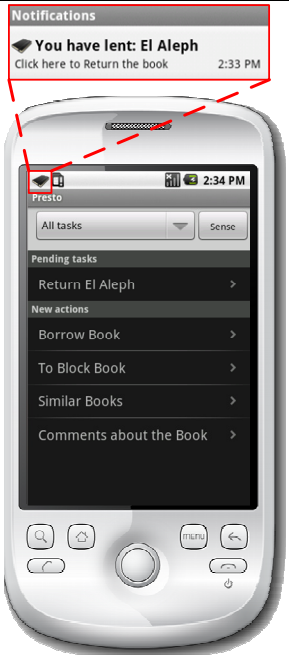


1. El usuario escoge la tarea Prestar Libro "Borrow Book".

2. Realiza la identificación del libro mediante un Identification Component.



3. El usuario confirma que desea prestar el libro.



4. Se proporciona feedback al usuario, a través de una notificación.

Fig. 4.11. Task Processor "Prestar Libro RF" implementado para Smart Library

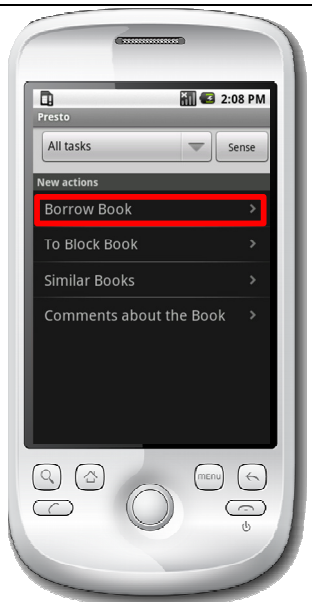
Prestar Libro en la región Reactive Background: Prestar Libro RB

La tarea involucra las siguientes acciones:

1. El usuario mediante el dispositivo móvil realiza la identificación del libro que desea prestar. Para ello hace uso de alguno de los Identification Components definidos en la Sección 4.2.1

2. La interfaz de usuario despliega información relevante al libro seleccionado. Si el libro no se encuentra disponible en el momento, el usuario es informado de esta situación y se le ofrece la posibilidad de seleccionar otro.
3. Si el libro está disponible, se solicita al usuario que confirme su intención de prestarlo.

La Figura 4.12 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea "Prestar Libro" en la región Reactive Background:



1. El usuario escoge la tarea Prestar Libro "Borrow Book".

2. Realiza la identificación del libro mediante un Identification Component.

3. El usuario confirma que desea prestar el libro.

Fig. 4.12. Task Processor "Prestar Libro RB" implementado para Smart Library

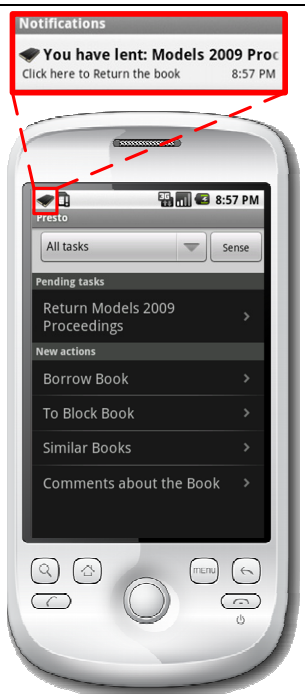
Prestar Libro en la región Proactive Foreground: Prestar Libro PF

La tarea involucra las siguientes acciones:

1. El usuario activa el lector RFID en su dispositivo móvil, cuando se detecta la proximidad de un libro que está disponible, inmediatamente se realiza su préstamo. Si el libro no se encuentra disponible en el momento, el usuario es informado de esta situación y se le ofrece la posibilidad de seleccionar otro.
2. Se envía una notificación al dispositivo móvil del usuario indicando datos relacionados con el préstamo, e informándole que puede acceder a una opción de "Retornar el libro" para anular dicho préstamo.

La Figura 4.13 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea "Prestar Libro" en la región Proactive Foreground:

1. El usuario activa el lector RFID en su dispositivo móvil.



2. Se envía una notificación al dispositivo móvil del usuario indicando datos relacionados con el préstamo, e informándole que puede acceder a una opción de "Retornar el libro" para anular dicho préstamo.

Fig. 4.13. Task Processor "Prestar Libro RB" implementado para Smart Library

Prestar Libro en la región Proactive Background: Prestar Libro PB

La tarea involucra las siguientes acciones:

1. El usuario activa el lector RFID en su dispositivo móvil, cuando se detecta la proximidad de un libro que está disponible, inmediatamente se realiza su préstamo. Si el libro no se encuentra disponible en el momento, el usuario es informado de esta situación y se le ofrece la posibilidad de seleccionar otro.

La Figura 4.14 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea "Prestar Libro" en la región Proactive Background:

1. El usuario activa el lector RFID en su dispositivo móvil. El libro se toma como prestado sin notificar al usuario de su préstamo.

Fig. 4.14. Task Processor "Prestar Libro PB" implementado para Smart Library

4.2.3.2. Task Processors implementados para dar soporte a la tarea "Retornar Libro"

La tarea "Retornar Libro" soporta la funcionalidad del retorno de los libros a la biblioteca. El Task processor que implementa esta funcionalidad se comunica con el Data Provider correspondiente para accesar los servicios digitales de la biblioteca en orden a recuperar la información relacionada al libro y registrar la respectiva devolución. Proporcionando a su vez, los mecanismos de interacción adecuados para completar la tarea.

"Retornar Libro" se implementó en Android utilizando principalmente los siguientes componentes:

- ✓ Activity. Presenta la interfaz gráfica de usuario, desplegando información referente al libro a retornar y proporcionado la interactividad requerida para completar la tarea. Este componente soporta la funcionalidad básica de la tarea.
- ✓ Intent Filter. Maneja el paso de mensajes (comunicación) entre el Task Processor y los demás componentes de la plataforma. El intent filter básico para el adecuado funcionamiento de la tarea es el siguiente:

Intent Filter		
Parámetros	**Valor**	**Descripción**
action android:name	"es.upv.pros.smartlib.RETURN"	Inicia la ejecución de la tarea "Retornar Libro" en el momento que recibe de los otros componentes de la plataforma un mensaje (Intent) con dicha action como parámetro.
category android:name	"android.intent.category.DEFAULT"	

- ✓ Notification[15]. Es usado para informar el resultado de la ejecución de la tarea al usuario.

La tarea "Retornar Libro" completa la tarea pendiente que originó "Prestar Libro". Lo anterior se consigue en dos pasos:

1. Cuando el usuario selecciona la tarea "Retornar Libro", Presto envía un mensaje (Intent) al Task Processor que implementa esta funcionalidad con los parámetros que se indican en la tabla 4.2:

Parámetro	**Tipo de dato**	**Valor/Descripción**
action	String	"es.upv.pros.smartlib.RETURN" Este parámetro inicia la ejecución de la tarea "Retornar Libro".
Extras		
es.upv.pros.presto.data	String	Corresponde a la información relativa al libro, proporcionada por el Task Processor "Prestar Libro" en el momento de ordenar la creación de la tarea pendiente.
es.upv.pros.presto.id	String	Corresponde al código único que

[15] El componente Notification es usado por el Task Processor dependiendo del nivel de obtrusividad deseado. Por esta razón los Task Processors implementados en la región Proactive/Reactive Background (PB/RB) no requieren de su uso.

		identifica al libro.

Tabla 4.2. Mensaje (Intent) que Presto envía al Task Processor "Retornar Libro"

2. Una vez realizada la tarea, el Task Processor "Retornar Libro" envía un mensaje (Intent) a Presto -exactamente al Task Manager- para que elimine la tarea de la lista de pendientes. El Intent debe contener los parámetros que se indican en la Tabla 4.3:

Parámetro	**Tipo de dato**	**Descripción**	**Valor**
action	String	Está instrucción le indica a Presto que debe eliminar la tarea de la lista de pendientes (*Pending task*).	"es.upv.pros.presto.COMPLETE"
Extras			
presto.id	String	Corresponde al código único que identifica al libro. Es el mismo que se suministra en el parámetro "es.upv.pros.presto.id" de la Tabla 4.2.	
presto.where	String	Especifica cuál tarea se debe borrar de la tabla Task, de la base de datos Tasks_list.db -ver Fig. 3.8-	"item_id = " + {presto.id}

Tabla 4.3. Parámetros para completar una Tarea Pendiente (Pending Task)

La Figura 4.15 ilustra la implementación del Task Processor -Retornar Libro- usando la notación gráfica definida para Android. En ella se ilustran los diferentes elementos requeridos para soportar la funcionalidad de la tarea.

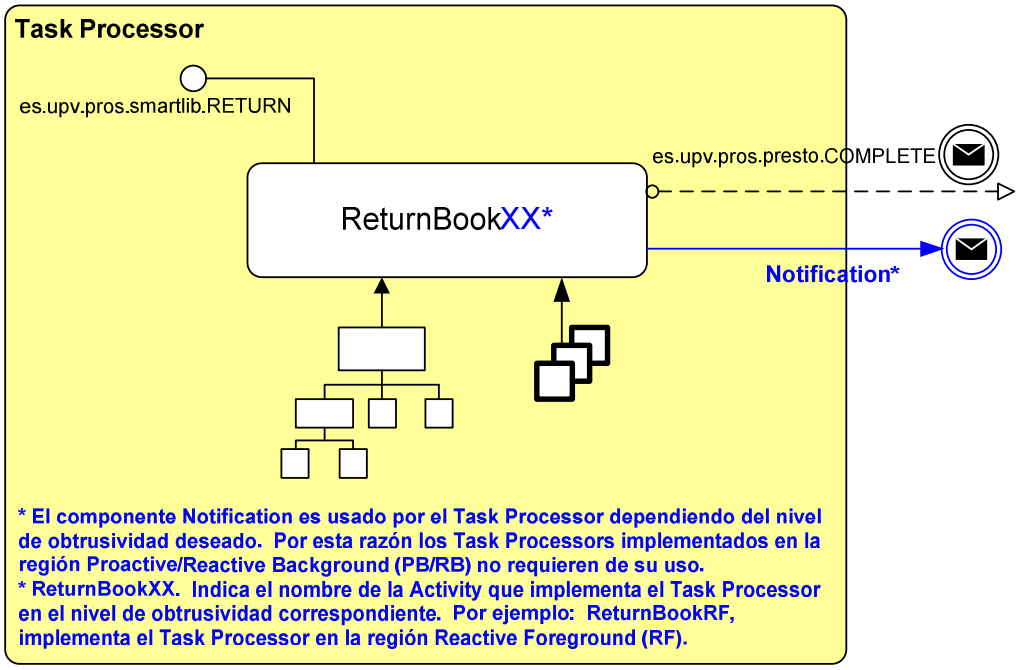


Fig. 4.15. Task Processor "Retornar Libro" implementado para Smart Library

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación Smartlibrary, en el cual se resaltan varios de los componentes que implementan la funcionalidad del Task Processor ilustrado en la Figura 4.15. La definición del Task Processor del ejemplo corresponde a la tarea "Retornar libro" implementada en la región Reactive Foreground (RF):

```
<activity android:name=".ReturnBookRF">
      <intent-filter android:label="Return Book">
         <category android:name="android.intent.category.DEFAULT"></category>
         <action android:name="es.upv.pros.smartlib.RETURN"></action>
      </intent-filter>
</activity>
```

A continuación se detalla el tipo de interacción requerida para la tarea "Retornar Libro", de acuerdo a la obtrusividad deseada:

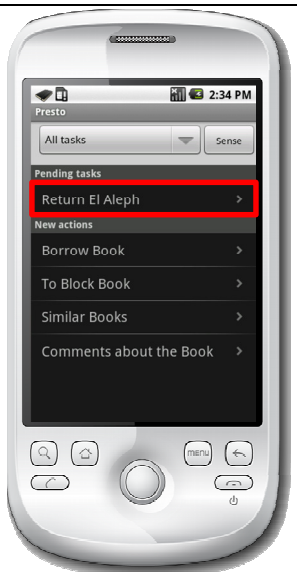
Retornar Libro en la región Reactive Foreground: Retornar Libro RF

La tarea involucra las siguientes acciones:

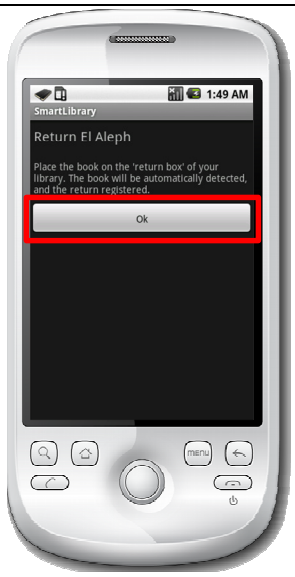
1. El usuario mediante el dispositivo móvil selecciona la tarea pendiente correspondiente al libro que desea Retornar.
2. La interfaz de usuario despliega el título del libro a retornar e información que le indica dónde debe ubicarlo. Adicionalmente, solicita al usuario que confirme su intención de retornar el libro.

3. Una vez realizado el proceso, se envía una notificación al dispositivo móvil del usuario indicando datos relacionados con la operación: la hora, el título del libro, entre otros.

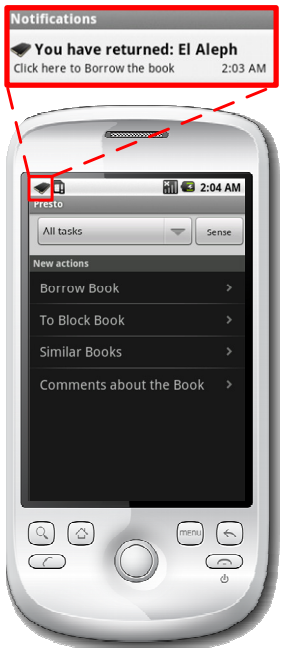
La Figura 4.16 ilustra en el dispositivo móvil las acciones requeridas para desarrollar la tarea "Retornar Libro" en la región Reactive Foreground:



1. El usuario selecciona la tarea pendiente "Return El Aleph".



2. El usuario confirma que desea retornar el libro.



Se proporciona feedback al usuario, a través de una notificación.

Fig. 4.16. Task Processor "Retornar Libro RF" implementado para Smart Library

Retornar Libro en la región Reactive Background: Retornar Libro RB

La tarea involucra las siguientes acciones:

1. El usuario mediante el dispositivo móvil selecciona la tarea pendiente correspondiente al libro que desea Retornar.
2. Se ejecuta la devolución del libro sin ofrecer feedback al usuario sobre el resultado de la tarea.

La Figura 4.17 ilustra en el dispositivo móvil las acciones requeridas para desarrollar la tarea "Retornar Libro" en la región Reactive Background:

El usuario selecciona la tarea pendiente "Return El Aleph". La devolución del libro se ejecuta sin ofrecer feedback al usuario.

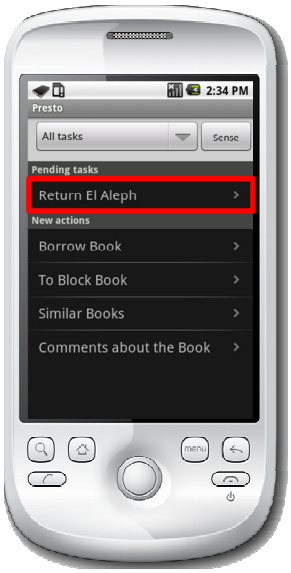
Fig. 4.17. Task Processor "Retornar Libro RB" implementado para Smart Library

Retornar Libro en la región Proactive Foreground: Retornar Libro PF

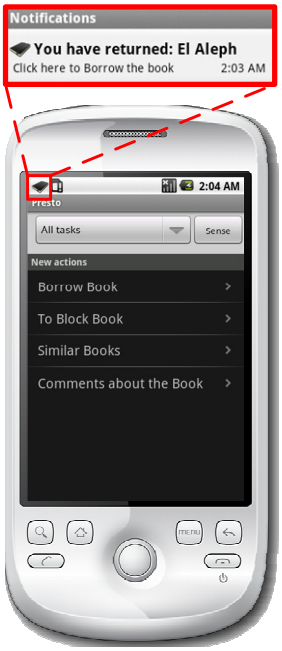
La tarea involucra las siguientes acciones:

1. El usuario mediante el dispositivo móvil selecciona la tarea pendiente correspondiente al libro que desea Retornar.
2. El proceso de devolución del libro se realiza sin solicitar confirmación al usuario. Una vez realizado el proceso, se envía una notificación al dispositivo móvil del usuario indicando datos relacionados con la operación: la hora, el título del libro, entre otros.

La Figura 4.18 ilustra en el dispositivo móvil las acciones requeridas para desarrollar la tarea "Retornar Libro" en la región Proactive Foreground:



1. El usuario selecciona la tarea pendiente "Return El Aleph".



2. Se proporciona feedback al usuario, sobre el resultado del proceso a través de una notificación.

Fig. 4.18. Task Processor "Retornar Libro PF" implementado para Smart Library

4.2.3.3. Task Processors implementados para dar soporte a la tarea "Comentarios"

La tarea "Comentarios" posibilita al usuario para consultar comentarios que otros usuarios han realizado acerca de un libro dado, permitiéndole también agregar sus propios comentarios. El Task processor que implementa esta funcionalidad se comunica con los Identification Components definidos, para realizar la identificación del libro a través del dispositivo móvil del usuario. Usando el correspondiente Data Provider, accesa los servicios digitales de la biblioteca en orden a recuperar la información relacionada al libro identificado y registrar el nuevo comentario si es el caso. Proporcionando los mecanismos de interacción adecuados para completar la tarea.

"Comentarios" se implementó en Android utilizando principalmente los siguientes componentes:

- ✓ Activity. Presenta la interfaz gráfica de usuario, desplegando información referente al libro seleccionado y proporcionado la interactividad requerida para ejecutar la tarea. Este componente soporta la funcionalidad básica de la tarea.
- ✓ Service. El componente Service se utiliza para alertar al usuario cuando el libro que éste ha identificado presenta comentarios negativos.

- ✓ Intent Filter. Maneja el paso de mensajes (comunicación) entre el Task Processor y los demás componentes de la plataforma. Los intent filter básicos para el adecuado funcionamiento de la tarea son:

Intent Filter		
Parámetros	**Valor**	**Descripción**
android:label	"Comments about the Book"	Corresponde al nombre de la tarea (Task Processor) que será presentado al usuario en la IU de Presto.
action android:name	"presto.pros.upv.es.task"	Este conjunto de parámetros indica a Presto que la tarea se puede ejecutar en el modo task-driven -ver Tabla 3.3-. Por tanto, cuando se ejecuta la aplicación "Presto" en el dispositivo móvil del usuario, esta tarea es desplegada como una de las Tareas de inicio que se pueden realizar.
action android:name	"es.upv.pros.smartlib.CommentsRF[16]"	
category android:name	"presto.pros.upv.es.launchable"	
category android:name	"android.intent.category.DEFAULT"	

- ✓ Notification[17]. Es usado por el componente Service para alertar al usuario cuando el libro que éste ha identificado presenta comentarios negativos.

La Figura 4.19 ilustra la implementación del Task Processor -Comentarios- usando la notación gráfica definida para Android. En la figura se observa la descripción de los dos componentes Activity que soportan la funcionalidad básica de la tarea, la cual es Consultar y Realizar comentarios sobre un libro particular. También se aprecian los componentes Service y Notification, cuya función es alertar al usuario cuando un libro presenta comentarios negativos, permitiéndole a través de la notificación acceder a la tarea "Comentarios". Finalmente, se ilustran los Intent Filters requeridos para permitir la comunicación entre los elementos que implementan el Task Processor y los otros componentes de la plataforma.

[16] es.upv.pros.smartlib.CommentsRF, corresponde al nombre de la Activity que implementa el Task Processor en la región Reactive Foreground (RF).

[17] El componente Notification es usado por el Task Processor dependiendo del nivel de obtrusividad deseado. Por esta razón los Task Processors implementados en la región Proactive/Reactive Background (PB/RB) no requieren de su uso.

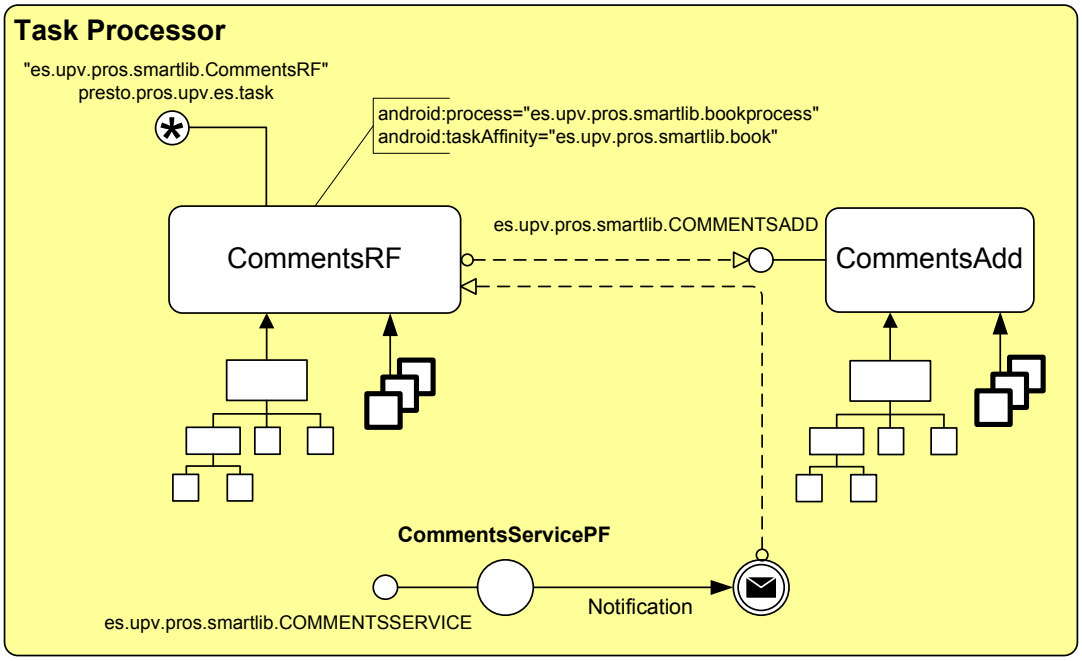


Fig. 4.19. Task Processor "Comentarios" implementado para Smart Library

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación Smartlibrary, en el cual se resaltan varios de los componentes que implementan la funcionalidad del Task Processor ilustrado en la Figura 4.19.

```
<activity android:name=".CommentsRF" android:label="@string/comments_name"
      android:process="es.upv.pros.smartlib.bookprocess"
      android:taskAffinity="es.upv.pros.smartlib.book">
      <intent-filter android:label="Comments about the Book">
         <action android:name="presto.pros.upv.es.task"></action>
         <action android:name="es.upv.pros.smartlib.CommentsRF"></action>
         <category android:name="presto.pros.upv.es.launchable"></category>
         <category android:name="android.intent.category.DEFAULT"></category>
      </intent-filter>
</activity>


<activity android:name=".CommentsAdd"
      android:label="@string/comments_addnew">
      <intent-filter>
         <category android:name="android.intent.category.DEFAULT"></category>
         <action android:name="es.upv.pros.smartlib.COMMENTSADD"></action>
      </intent-filter>
</activity>


<service android:name=".CommentsServiceBR">
      <intent-filter>
         <action
               android:name="es.upv.pros.smartlib.COMMENTSSERVICE">
         </action>
      </intent-filter>
</service>
```

A continuación se detalla el tipo de interacción requerida para la tarea "Comentarios", de acuerdo a la obtrusividad deseada:

Comentarios, en la región Reactive Foreground: Comentarios RF

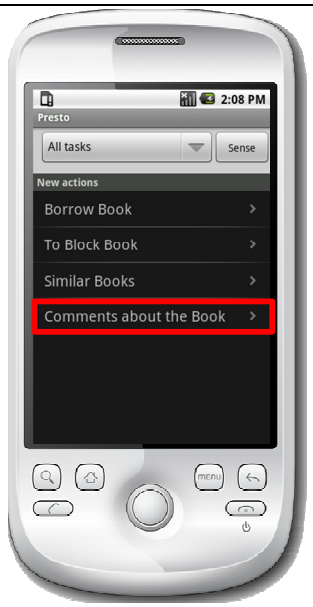
La tarea involucra las siguientes acciones:

1. El usuario mediante el dispositivo móvil realiza la identificación del libro sobre el cual desea consultar o agregar comentarios. Para ello hace uso de alguno de los Identification Components definidos en la Sección 4.2.1
2. La interfaz de usuario despliega el título del libro, la calificación que otros usuarios han otorgado al libro y la lista de comentarios registrados. Si se desea agregar un comentario respecto al libro, se debe hacer lo siguiente:

   Usando la tecla "Menú" de su dispositivo móvil el usuario accede a la opción "Agregar comentario"

   El usuario completa la información solicitada y realiza el envío de su comentario.

La Figura 4.20 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea "Comentarios" en la región Reactive Foreground:

	
1. El usuario escoge la tarea Comentarios "Comments about the Book".	2. Realiza la identificación del libro mediante un Identification Component.

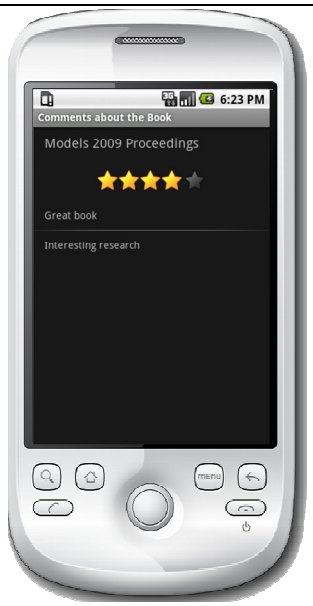
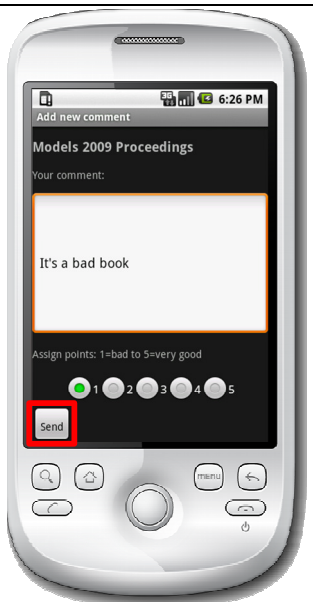
	
3. La interfaz de usuario despliega el título del libro, la calificación que otros usuarios han otorgado al libro y la lista de comentarios registrados.	4. Usando la tecla "Menú" de su dispositivo móvil el usuario accede a la opción "Agregar comentario". Completa la información solicitada y confirma el envío de su comentario.

Fig. 4.20. Task Processor "Comentarios RF" implementado para Smart Library

Comentarios, en la región Proactive Foreground: Comentarios PF

Dado que la tarea "Comentarios PF" no presenta Interfaz de Usuario, su implementación se realizó utilizando el Service "CommentsServicePF" descrito anteriormente en la Figura 4.19.

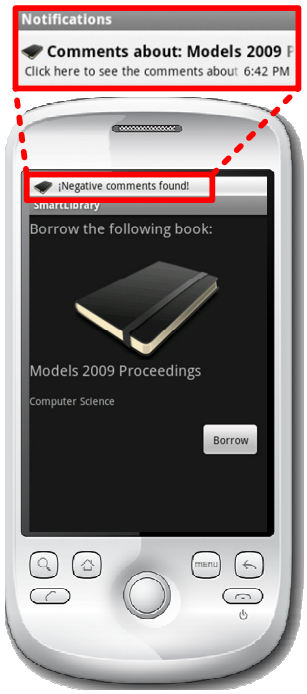
La tarea involucra las siguientes acciones:

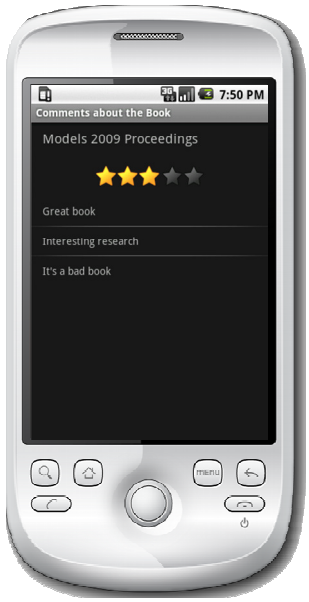
1. Cuando un libro identificado tiene comentarios negativos, se envía una notificación al dispositivo móvil del usuario informándole de tal situación.
2. El usuario puede acceder a la tarea "Comentarios RF" a través de la notificación enviada.

La Figura 4.21 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea "Comentarios" en la región Proactive Foreground:

1. El usuario realiza la identificación del libro mediante un Identification Component.

2. Si el libro identificado tiene comentarios negativos, se envía una notificación al dispositivo móvil del usuario informándole de tal situación.





3. El usuario puede acceder a la tarea "Comentarios RF" a través de la notificación enviada.

Fig. 4.21. Task Processor "Comentarios PF" implementado para Smart Library

4.2.3.4. Task Processors implementados para dar soporte a la tarea "Similares"

La tarea "Similares" proporciona información al usuario sobre libros disponibles similares al que en ese momento está consultando. El Task processor que implementa esta funcionalidad se comunica con los Identification Components definidos, para realizar la identificación del libro a través del dispositivo móvil del usuario. Empleando el correspondiente Data Provider, accesa los servicios digitales de la biblioteca en orden a recuperar la información relacionada al libro identificado y sus libros similares. Proporcionando los mecanismos de interacción adecuados para completar la tarea.

"Similares" se implementó en Android utilizando principalmente los siguientes componentes:

- ✓ Activity. Presenta la interfaz gráfica de usuario, desplegando información referente al libro seleccionado y proporcionado la interactividad requerida para ejecutar la tarea. Este componente soporta la funcionalidad básica de la tarea.
- ✓ Service. El componente Service se utiliza para alertar al usuario cuando existen libros disponibles similares al seleccionado.
- ✓ Intent Filter. Maneja el paso de mensajes (comunicación) entre el Task Processor y los demás componentes de la plataforma. Los intent filter básicos para el adecuado funcionamiento de la tarea son:

Intent Filter		
Parámetros	**Valor**	**Descripción**
android:label	"Similar Books"	Corresponde al nombre de la tarea (Task Processor) que será presentado al usuario en la IU de Presto.
action android:name	"presto.pros.upv.es.task"	Este conjunto de parámetros indica a Presto que la tarea se puede ejecutar en el modo task-driven -ver Tabla 3.3-. Por tanto, cuando se ejecuta la aplicación "Presto" en el dispositivo móvil del usuario, esta tarea es desplegada como una de las Tareas de inicio que se pueden realizar.
action android:name	"es.upv.pros.smartlib.SimilarsRF"[18]	
category android:name	"presto.pros.upv.es.launchable"	
category android:name	"android.intent.category.DEFAULT"	

[18] es.upv.pros.smartlib.SimilarsRF, corresponde al nombre de la Activity que implementa el Task Processor en la región Reactive Foreground (RF).

- ✓ Notification[19]. Es usado por el componente Service para alertar al usuario cuando existen libros disponibles similares al seleccionado.

La Figura 4.22 ilustra la implementación del Task Processor -Similares- usando la notación gráfica definida para Android. En la figura se describen los dos componentes Activity que soportan la Interfaz de Usuario, uno de ellos despliega la lista de los libros similares encontrados, y el otro despliega la ubicación y datos adicionales sobre un libro específico seleccionado. También se aprecian los componentes Service y Notification, cuya función es alertar al usuario cuando existen libros disponibles similares al seleccionado, permitiéndole a través de la notificación acceder a la tarea "Similares". Finalmente, se ilustran los Intent Filters requeridos para permitir la comunicación entre los elementos que implementan el Task Processor y los otros componentes de la plataforma.

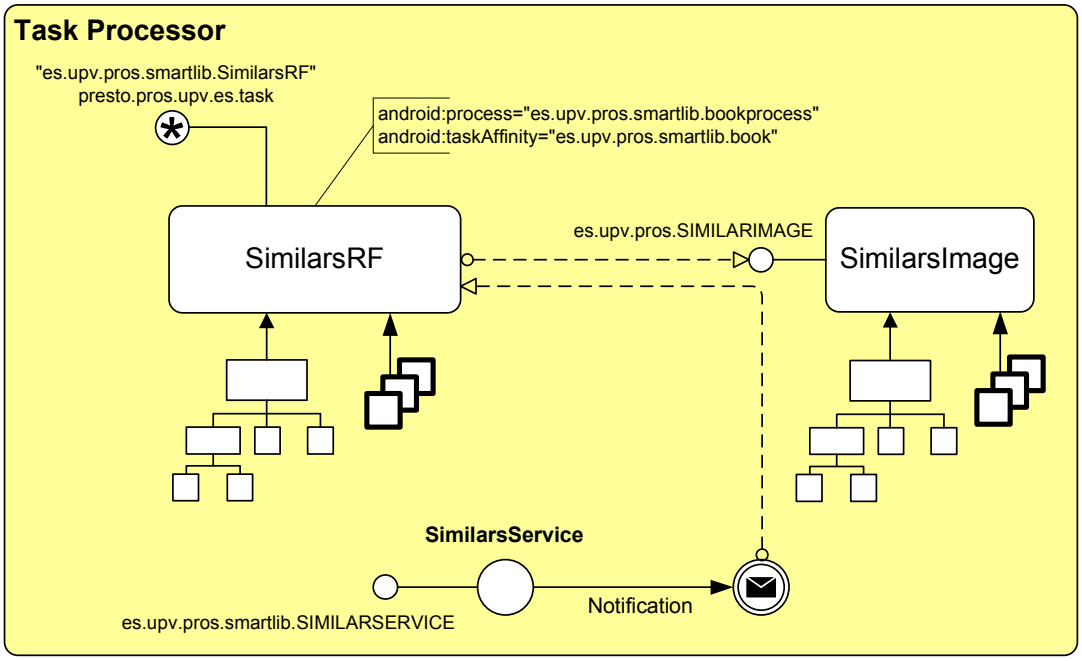


Fig. 4.22. Task Processor "Similares" implementado para Smart Library

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación Smartlibrary, en el cual se resaltan varios de los componentes que implementan la funcionalidad del Task Processor ilustrado en la Figura 4.22.

```
<activity android:name=".SimilarsRF" android:label="@string/similars_name"
      android:process="es.upv.pros.smartlib.bookprocess"
      android:taskAffinity="es.upv.pros.smartlib.book">
```

[19] El componente Notification es usado por el Task Processor dependiendo del nivel de obtrusividad deseado. Por esta razón los Task Processors implementados en la región Proactive/Reactive Background (PB/RB) no requieren de su uso.

```
        <intent-filter android:label="Similar Books">
            <action android:name="presto.pros.upv.es.task"></action>
            <action android:name="es.upv.pros.smartlib.SimilarsRF"></action>
            <category android:name="presto.pros.upv.es.launchable"></category>
            <category android:name="android.intent.category.DEFAULT"></category>
        </intent-filter>
    </activity>

    <activity android:name=".SimilarsImage">
        <intent-filter android:label="Book information">
            <category android:name="android.intent.category.DEFAULT"></category>
            <action android:name="es.upv.pros.SIMILARIMAGE"></action>
        </intent-filter>
    </activity>

    <service android:name=".SimilarsService">
        <intent-filter>
            <action android:name="es.upv.pros.smartlib.SIMILARSERVICE"></action>
        </intent-filter>
    </service>
```

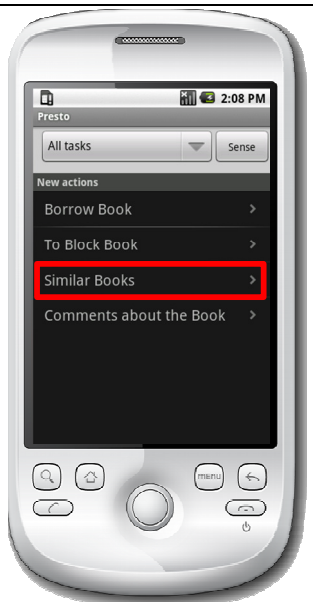
A continuación se detalla el tipo de interacción requerida para la tarea “Similares”, de acuerdo a la obtrusividad deseada:

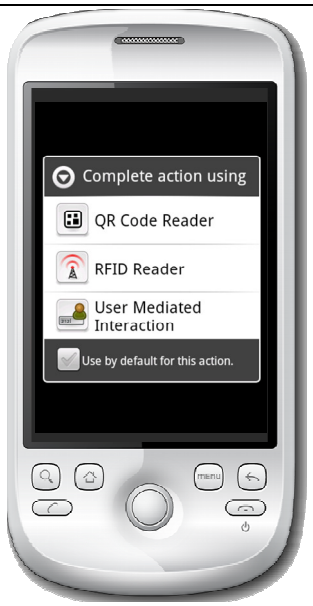
Similares, en la región Reactive Foreground: Similares RF

La tarea involucra las siguientes acciones:

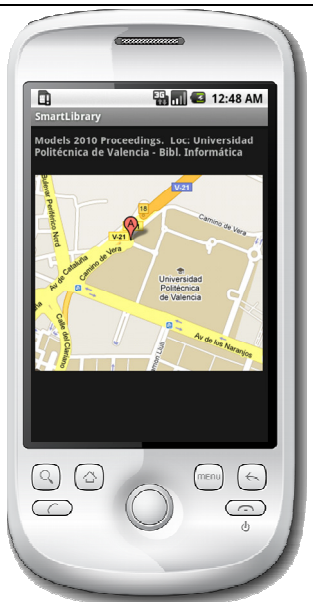
1. El usuario mediante el dispositivo móvil realiza la identificación del libro sobre el cual desea consultar información de libros similares. Para ello hace uso de alguno de los Identification Components definidos en la Sección 4.2.1
2. La interfaz de usuario despliega una lista de los libros disponibles similares al identificado. Así mismo, permite escoger un libro de la lista y brinda información más detallada sobre éste: título del libro, ubicación, imagen, entre otros.

La Figura 4.23 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea “Similares” en la región Reactive Foreground:

<table>
<tr>
<td> 

1. El usuario escoge la tarea Similares “Similar Books”.</td>
<td> 

2. Realiza la identificación del libro mediante un Identification Component.</td>
</tr>
<tr>
<td> 

3. La interfaz de usuario despliega la lista de los libros disponibles similares al identificado.</td>
<td> 

4. Al seleccionar uno de los libros de la lista se ofrece información más detallada sobre éste: título, ubicación, imagen, entre otros.</td>
</tr>
</table>

Fig. 4.23. Task Processor “Similares RF” implementado para Smart Library

Similares, en la región Proactive Foreground: Similares PF

Dado que la tarea “Similares PF” no presenta Interfaz de Usuario, su implementación se realizó utilizando el Service “SimilarsService” descrito anteriormente en la Figura 4.22.

La tarea involucra las siguientes acciones:

1. Cuando existen libros disponibles similares al libro identificado, se envía una notificación al dispositivo móvil del usuario informándole de tal situación.
2. El usuario puede acceder a la tarea “Similares RF” a través de la notificación enviada.

La Figura 4.24 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea “Similares” en la región Proactive Foreground:

	
1. El usuario realiza la identificación del libro mediante un Identification Component.	2. Si existen libros disponibles similares al libro identificado, se envía una notificación al dispositivo móvil del usuario informándole de tal situación.

3. El usuario puede acceder a la tarea "Similares RF" a través de la notificación enviada.

Fig. 4.24. Task Processor "Similares PF" implementado para Smart Library

4.2.3.5. Task Processor implementado para dar soporte a la tarea "Bloquear Libro"

La tarea "Bloquear Libro" ofrece al usuario la posibilidad de bloquear un libro durante diez minutos para que éste pueda continuar buscando otros libros, con esto se evita que el usuario tenga que llevar físicamente los libros consigo mientras busca otros que también le interesan. El Task processor que implementa esta funcionalidad se comunica con los Identification Components definidos, para realizar la identificación del libro a través del dispositivo móvil del usuario. Empleando el correspondiente Data Provider, accesa los servicios digitales de la biblioteca en orden a recuperar la información relacionada al libro identificado y cambiar su estado a "bloqueado". Proporcionando los mecanismos de interacción adecuados para completar la tarea.

"Bloquear Libro" se implementó en Android utilizando principalmente los siguientes componentes:

- ✓ Activity. Aunque la tarea no presenta interfaz gráfica de usuario, este componente es requerido para que la tarea se pueda ejecutar en Presto en el modo task-driven, introducido en la Sección 3.2.1.2.

- ✓ Service. El componente Service se utiliza para medir el tiempo que el libro permanece bloqueado, y alertar al usuario cuando este tiempo se haya terminado. Este componente soporta la funcionalidad básica de la tarea.
- ✓ Intent Filter. Maneja el paso de mensajes (comunicación) entre el Task Processor y los demás componentes de la plataforma. Los intent filter básicos para el adecuado funcionamiento de la tarea son:

Intent Filter		
Parámetros	**Valor**	**Descripción**
android:label	"To Block Book"	Corresponde al nombre de la tarea (Task Processor) que será presentado al usuario en la IU de Presto.
action android:name	"presto.pros.upv.es.task"	Este conjunto de parámetros indica a Presto que la tarea se puede ejecutar en el modo task-driven -ver Tabla 3.3-. Por tanto, cuando se ejecuta la aplicación "Presto" en el dispositivo móvil del usuario, esta tarea es desplegada como una de las Tareas de inicio que se pueden realizar.
action android:name	"es.upv.pros.smartlib.BlockRF"[20]	
category android:name	"presto.pros.upv.es.launchable"	
category android:name	"android.intent.category.DEFAULT"	

- ✓ Notification[21]. Es usado por el componente Service para alertar al usuario cuando el tiempo de bloqueo del libro ha terminado.

La Figura 4.25 ilustra la implementación del Task Processor -Bloquear Libro- usando la notación gráfica definida para Android. En la figura se describe el componente Activity que soporta la funcionalidad de la tarea en el modo task-driven. También se aprecian los componentes Service y Notification, cuya función respectivamente es medir el tiempo que el libro permanece bloqueado y alertar al usuario cuando este tiempo ha terminado. Finalmente, se ilustran los Intent Filters requeridos para permitir la comunicación entre los elementos que implementan el Task Processor y los otros componentes de la plataforma.

[20] "es.upv.pros.smartlib.BlockRF", corresponde al nombre de la Activity que implementa el Task Processor en la región Reactive Foreground (RF).
[21] El componente Notification es usado por el Task Processor dependiendo del nivel de obtrusividad deseado. Por esta razón los Task Processors implementados en la región Proactive/Reactive Background (PB/RB) no requieren de su uso.

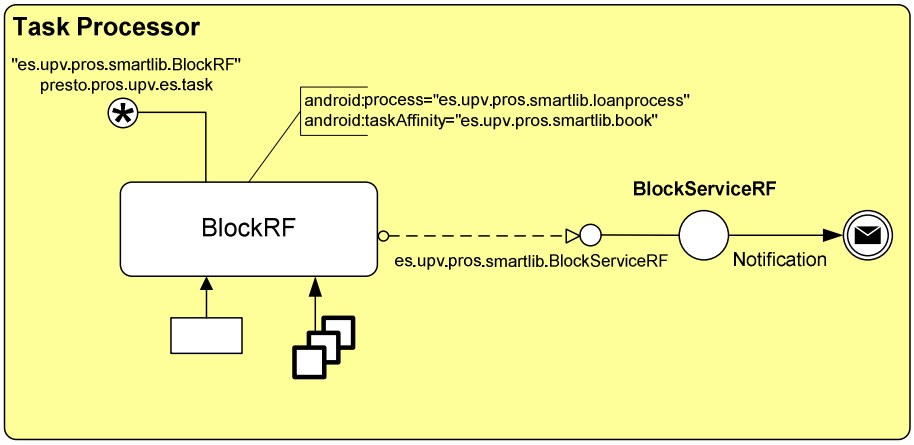


Fig. 4.25. Task Processor "Bloquear Libro" implementado para Smart Library

El siguiente fragmento de código corresponde a la definición del AndroidManifest.xml de la aplicación Smartlibrary, en el cual se resaltan varios de los componentes que implementan la funcionalidad del Task Processor ilustrado en la Figura 4.25.

```
<activity android:name=".BlockRF" android:label="@string/blockrf_name"
   android:process="es.upv.pros.smartlib.loanprocess"
   android:taskAffinity="es.upv.pros.smartlib.book">
   <intent-filter android:label="To Block Book">
      <action android:name="presto.pros.upv.es.task"></action>
      <action android:name="es.upv.pros.smartlib.BlockRF"></action>
      <category android:name="presto.pros.upv.es.launchable"></category>
      <category android:name="android.intent.category.DEFAULT"></category>
   </intent-filter>
</activity>

<service android:name=".BlockServiceRF">
   <intent-filter android:label="@string/blockrf_name">
      <action android:name="es.upv.pros.smartlib.BlockServiceRF"></action>
   </intent-filter>
</service>
```

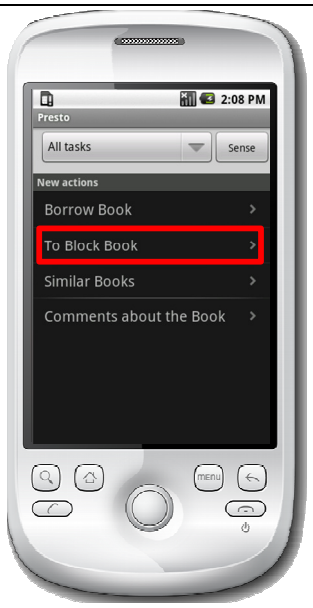
Bloquear Libro, en la región Reactive Foreground: Bloquear Libro RF

Dado que la tarea "Bloquear Libro RF" no presenta Interfaz de Usuario, su implementación se realizó principalmente utilizando el Service "BlockServiceRF" descrito anteriormente en la Figura 4.25.

La tarea involucra las siguientes acciones:

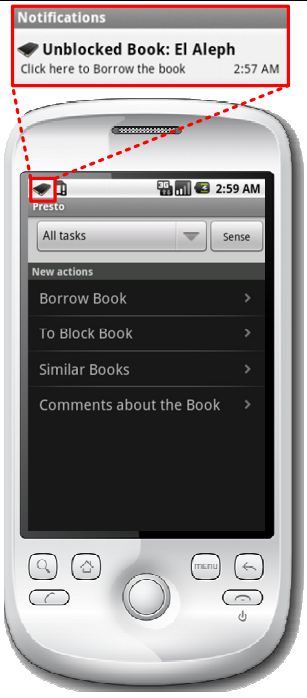
1. El usuario mediante el dispositivo móvil realiza la identificación del libro que desea bloquear. Para ello hace uso de alguno de los Identification Components definidos en la Sección 4.2.1
2. Cuando el tiempo de bloqueo ha terminado, se envía una notificación al dispositivo móvil del usuario informándole de tal situación. Dicha notificación ofrece un acceso directo a la tarea "Prestar Libro", en caso que al usuario le interese realizar el préstamo.

La Figura 4.26 ilustra en el dispositivo móvil las acciones necesarias para desarrollar la tarea "Bloquear Libro" en la región Reactive Foreground:



1. El usuario escoge la tarea Bloquear Libro "To Block Book".

2. Realiza la identificación del libro mediante un Identification Component.



3. Cuando el tiempo de bloqueo concluye, se envía una notificación al dispositivo móvil del usuario informándole de la situación. La notificación ofrece un acceso directo a la tarea "Prestar Libro", en caso que al usuario le interese realizar el préstamo..

Fig. 4.26. Task Processor "Bloquear Libro RF" implementado para Smart Library

5. Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones a las que se llegó después del desarrollo del trabajo de tesis, las contribuciones realizadas a la línea de investigación en la que se contextualiza, y se da una visión de extensiones que mejorarán el desarrollo presentado. A continuación se desarrolla el contenido de cada sección.

5.1 Conclusiones

Los principios arquitecturales sobre los cuales se fundamenta el sistema operativo móvil Android, ofrecen una plataforma adecuada para extender a Presto. Permitiendo implementar sus conceptos arquitectónicos genéricos y facilitando su personalización a través del desarrollo de nuevas aplicaciones por parte de terceros.

El uso de una notación gráfica para representar los diferentes componentes de una arquitectura basada en Android. Proporciona un medio entendible para simbolizar los elementos involucrados en la implementación de la arquitectura de Presto, y los concernientes a la implementación de la aplicación que dará soporte a los procesos de negocio considerados.

La utilización de dispositivos móviles con capacidades para identificar y detectar objetos del mundo real, sin duda beneficia el desarrollo de los procesos de negocio que involucran objetos físicos dentro de sus tareas. Proporcionando mayor fluidez y confiabialidad en las tareas realizadas, al evitar la intervención de las personas como portadoras de información, y permitiendo la movilidad que algunos procesos de negocio requieren.

Cuando se trabaja en sistemas de información, que dan soporte a procesos de negocio que involucran elementos del mundo real con información digital asociada, es indispensable tener claridad en el diseño de los diferentes tipos de interacción necesarios para llevar a cabo las tareas comprendidas en dicho proceso, éstas se pueden definir a través de la utilización del framework de interacciones implícitas introducido en [3]. Consiguiendo de esta manera la agilidad que requiere el proceso de negocio

El presente trabajo ha desarrollado una solución software que apoya la ejecución de las tareas de procesos de negocio en el contexto de la Internet de las Cosas, en un ambiente de movilidad. Atendiendo a los requisitos de obtrusividad considerados para cada tarea del proceso.

5.2 Contribuciones

El trabajo desarrollado en esta tesina constituye un ejercicio de aplicación de ideas y tecnologías avanzadas. El desarrollo obtenido ha permitido realizar en la práctica la evaluación de trabajos de investigación con presencia en foros importantes del área, como son los siguientes:

• Giner, P.; Cetina, C.; Fons, J.; Pelechano, V. (2009), Presto: A pluggable platform for supporting user participation in Smart Workflows, in 'Proceedings of the Sixth Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)'. Presented at the work-in-progress session.
• Giner, P.; Cetina, C.; Fons, J.; Pelechano, V. (2010), Developing Mobile Business Processes for the Internet of Things, in 'IEEE Pervasive Computing'. Accepted for publication subject to minor revisions.

5.2 Trabajo Futuro

En el trabajo de tesis desarrollado, fue de importante aplicación el concepto de obtrusividad presentado por las tareas requeridas para completar un proceso de negocio. Dicho desarrollo se realizó de manera manual, implementando el componente arquitectural que da soporte a cada tarea. Un futuro trabajo podría abordar la automatización de la arquitectura, para que apartir de la descripción del nivel de obtrusividad requerido en una tarea, se genere el componente que le dará soporte.

Trabajos que se están llevando cabo en la línea de investigación bajo la cual se desarrolló esta tesis, pretenden variar el nivel de obtrusividad según avanza el proceso. Para ello se integrarán trabajos de adaptación en run-time como este:
Cetina, C.; Giner, P.; Fons, J. & Pelechano, V. (2009), 'Autonomic Computing through Reuse of Variability Models at Runtime: The Case of Smart Homes', Computer 42(10), 37-43.
http://www.computer.org/portal/web/csdl/doi/10.1109/MC.2009.309

6. Referencias

[1] Giner, P., Cetina, C., Fons, J., Pelechano, V.: A Pluggable Platform for the Development of Smart Workflows.

[2] Wieland, M., Kaczmarczyk, P., Nicklas, D.: Context integration for smart workflows. In PerCom, p. 239 - 242, 2008.

[3] Ju, W., Leifer, L.: The Design of Implicit Interactions: Making Interactive Systems Less Obnoxious. In Design Issues, p. 72-84, 24(3) Summer 2008.

[4] Ju, W., Lee, B., Klemmer, S.: Range: Exploring Implicit Interaction through Electronic Whiteboard Design. 2008.

[5] Richardson, L., Ruby, S.: RESTful web services – Web Services for the Real World. O'REILLY, p. 343 – 354, 2007.

[6] Meier, R.: Professional Android Application Development. WROX, 2009.

[7] DiMarzio, J.: Android A Programmer's Guide. Mc Graw Gill, 2008.

[8] Gibbs, W.: Considerate Computing. Scientific American, p. 55 – 61, 2004.

[9] Einstein, G. O., McDaniel, M. A., Williford, C. L., Pagan, J. L., Dismukes, R. K.: Forgetting of intentions in demanding situations is rapid, Journal of Experimental Psychology: Applied, 9 (3), p. 147-162, 2003.

[10] Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee J., Yang, J.: Predicting human interruptibility with sensors: A Wizard of Oz feasibility study, in: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'03), New York: ACM Press, p. 257-264, 2003.

[11] Fogarty, J., Hudson, S. E., Atkeson, C. G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J. C., Yang, J.: Predicting human interruptibility with sensors, ACM Transactions on Computer-Human Interaction, 12 (1), p. 119-146, 2005.

[12] Fogarty, J., Hudson, S. E., Lai, J.: Examining the robustness of sensor-based statistical models of human interruptibility, in: Human Factors in Computing Systems: Proceedings of CHI'04, New York: ACM Press, p. 207-214, 2004.

[13] Czerwinski, M., Horvitz, E., Wilhite, S.: A diary study of task switching and interruptions, in: Human Factors in Computing Systems: Proceedings of CHI'04, New York: ACM Press, p. 175-182, 2004.

[14] Cutrell, E. B., Czerwinski, M., Horvitz, E.: Effects of instant messaging interruptions on computing tasks, in: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2000 Extended Abstracts), New York: ACM Press, p. 99-100, 2000.

[15] Hudson, S. E., Smith, I.: Techniques for addressing fundamental privacy and disruption tradeoffs in awareness support systems, in: Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work (CSCW'96), New York: ACM Press, p. 248-257, 1996.

[16] Vertegaal, R., Dickie, C., Sohn, C., Flickner, M.: Designing attentive cell phone using wearable eyecontact sensors, in: CHI '02 Extended Abstracts on Human Factors in Computing Systems, New York: ACM, p. 646-647, 2002.

[17] Bessière, K., Newhagen, J. E., Robinson, J. P., Shneiderman, B.: A model for computer frustration: The role of instrumental and dispositional factors on incident, session, and post-session frustration and mood, Computers in Human Behavior, 22 (6), p. 941-961. 2006.

[18] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S., Winer, D.: Simple object access protocol (SOAP) 1.1. Technical Report 08 May 2000, W3C (2000).

[19] Hackmann, G., Haitjema, M., Gill, C., Roman, G.-C.: “Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices”, Dept. of Computer Science and Engineering, Washington University in St. Louis. 2006.

[20] Bolcer, G.A.: “Magi: an architecture for mobile and disconnected workflow”, IEEE Internet Computing, Vol.4, Iss. 3, p. 46-54, May/Jun 2000.

[21] Pajunen, L., Chande, S.: Developing Workflow Engine for Mobile Devices, 11th IEEE International Enterprise Distributed Object Computing Conference, p. 279-286, 2007.

[22] Gershenfeld, N.: When Things Start to Think (New York: Henry Holt, 1999), p. 102.

[23] Dumas, M., & ter Hofstede, A. H. M. (2001). Uml activity diagrams as a workflow specification language. In UML’01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, (p. 76–90). London, UK: Springer-Verlag.

[24] Mayer, R. J., Painter, M. K., & DeWitte, P. (1992). Idef family of methods for concurrent engineering and business re-engineering applications. College Station, TX: Knowledge Based Systems, Inc.

[25] Hofreiter, B., Huemer, C., & Klas, W. (2002). ebxml: Status, research issues, and obstacles. In Proc. Of 12th Int. Workshop on Research Issues on Data Engineering (RIDE02), (p. 7–16).

[26] OMG (2006). Business Process Modeling Notation (BPMN) Specification. OMG Final Adopted Specification. Dtc/06-02-01.

[27] Documento de especificación: Web Services Business Process Execution Language Version 2.0. Disponible en http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[28] Giner, P.: Business Process Modeling for the Internet of Things. Master's Thesis, p. 9-12, 2008.