Universidad de las Ciencias Informáticas
Facultad 2

Título: BluePub Sender: Sistema de Envío de Publicidad vía Bluetooth v2.0.

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

Autor: Jossue Díaz Pérez

Tutores: Ing. Leslye Bravo García
Ing. Joan Martínez Herrera

La Habana, enero de 2013.
"Año 55 de la Revolución"

"Si no existe la organización, las ideas, después del primer momento de impulso, van perdiendo eficacia."

Ernesto Guevara de la Serna.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Telemática de la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autor:

_____

Jossue Díaz Pérez

Tutores:

_____ _____

Ing. Leslye Bravo García Ing. Joan Martínez Herrera

Dedicatoria

A mis queridos padres Ángela y Juan.

A mis abuelos, hermanos y demás familiares….

Especialmente a mi novia Lisset.

Agradecimientos

Quisiera poder devolver a todas las personas que colaboraron con la realización de este trabajo de diploma por su ayuda incondicional, agradezco principalmente a mi madre por darme todo y ayudar a formar la persona que soy hoy.

A mi padre que aunque ya no está en este mundo representa mi ejemplo y principal guía en mis estudios

Agradezco a mi hermano Juan por ser el ejemplo vivo de mi padre.

Agradezco a mi hermanito Julio por ver en él, el niño que un día fui y querer ser mejor cada día.

Agradezco a mis abuelos por luchar tanto en la vida, no menguar nunca y continuar trabajando aún.

Agradezco a mi novia Lisset, que personifica la inspiración para lograr salir adelante en la vida. Te amo.

Agradezco a Reynaldo por ofrecerme su apoyo incondicional para lo que necesitara.

Agradezco a mis tutores Leslye y Joan por preocuparse, encaminarme y guiarme en toda mi investigación.

Agradezco a todos mis amigos que me apoyaron y me ofrecieron ayuda.

Agradezco a los profesores que me formaron y enseñaron en esta universidad.

RESUMEN

Desde el surgimiento del comercio la publicidad ha sido la vía más eficiente para elevar el consumo de productos en venta. Sin embargo, en muchas ocasiones los medios para promover productos no son efectivos, debido a que no llegan a la mayor cantidad de público. Por lo que se hace necesario encontrar nuevas vías para la difusión de publicidad, incursionando de esta forma en la tecnología celular. El envío de contenido publicitario para teléfonos móviles haciendo uso de la tecnología Bluetooth disminuye potencialmente las barreras de comunicación entre el cliente y la empresa. En la Universidad de las Ciencias Informáticas (UCI) se desarrolló un sistema denominado BluePub Sender v1.0 que se encarga del envío de archivos publicitarios hacia los dispositivos móviles con tecnología Bluetooth. Sin embargo no se realiza una correcta gestión de los mismos en el terminal móvil debido a que los archivos publicitarios son almacenados por defecto en un directorio predefinido, lo que trae como consecuencia que la visualización, o borrado de una publicidad debe realizarse después de un proceso de localización manual. El presente trabajo pretende desarrollar un sistema que gestione en el teléfono móvil la publicidad recibida desde el sistema BluePub Sender v1.0. Para ello se describen los procesos de envío de publicidad que realiza el sistema BluePub Sender existente. Se obtuvo como resultado un sistema con las funcionalidades que facilitan la gestión del archivo publicitario, y de esta forma dar solución a una serie de problemas que existen en la ejecución del proceso publicitario.

PALABRAS CLAVE

Publicidad, Bluetooth, Tecnología, Android BluePub Sender v1.0

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La publicidad es una forma de comunicación de propósito comercial, que intenta incrementar el consumo de un producto o servicio utilizando distintos medios de difusión. Desde el surgimiento del comercio ha sido una necesidad para las empresas dar a conocer a la mayor cantidad de clientes, los productos que ofertan, pero esta tarea en la mayoría de los casos es bastante compleja debido a que depende de los medios de publicidad, de los recursos con que cuente la empresa y de cuan efectivo sea el anuncio publicitario. (1)

Con el desarrollo de la humanidad han evolucionado también las formas y técnicas de difundir publicidad como son la expresión oral, carteles y distintas técnicas para atraer la atención del público. Posteriormente; el surgimiento de la imprenta permitió la difusión más extensa de los mensajes publicitarios con la impresión de periódicos, almanaques y afiches.

En la actualidad la manera de ofrecer publicidad ha progresado en gran medida con respecto a los métodos utilizados anteriormente, desde los medios de difusión masivos hasta la telefonía celular. Los celulares representan, mejor que ningún otro medio, un punto de convergencia tecnológica digno de consideración. Esto se debe a que no solo transmiten conversaciones, sino que incursionan en el envío de mensajes, imágenes, videos, aplicaciones y otros tipos de datos. Asimismo son capaces de establecer conexión a Internet gracias a su capacidad de comunicación inalámbrica y sirven como medio de publicidad.

Para gestionar todas las funcionalidades que ofrece un teléfono celular se hace necesario que estos posean un sistema operativo (SO) en específico. A medida que los teléfonos móviles crecen en popularidad, los SO con los que funcionan adquieren mayor importancia.

Android es un SO de código abierto para teléfonos móviles que brinda numerosas ventajas algunas de las cuales son: es multitarea, posee un amplio soporte a multimedia y tecnologías de conectividad inalámbricas como WIFI y Bluetooth (tecnología de ondas de radio de corto alcance). (2)

La tecnología Bluetooth, surge por la necesidad de mantener conectados los teléfonos móviles para intercambiar información de manera inalámbrica. La comunicación puede establecerse a través de dispositivos con Bluetooth, creando una Red de Área Personal (PAN) entre un dispositivo y otro. La publicidad por proximidad haciendo uso de la tecnología Bluetooth aumenta la comunicación entre el cliente y la empresa, gracias a esta el cliente se mantiene informado de nuevos productos, rebajas de

precios, entre otros servicios de forma directa, además es una publicidad no intrusiva puesto que el cliente tiene la opción de decidir entre recibirla o no.

En Cuba, la publicidad haciendo uso de las nuevas tecnologías y medios de información es un campo prácticamente inexplorado, los centros comerciales y entidades de venta carecen de un sistema que les permita enviar la publicidad de sus productos. Los métodos usados son primarios, entre ellos se encuentran: los afiches en el interior de los salones comerciales, los seminarios y cursos que se les brindan a los trabajadores encargados de la venta de los productos y de relacionarse directamente con el cliente.

En la Universidad de las Ciencias Informáticas (UCI) se desarrolló un sistema denominado BluePub Sender v1.0 que se encarga del envío de archivos publicitarios hacia los dispositivos móviles con tecnología Bluetooth. Sin embargo no se realiza una correcta gestión del archivo publicitario en el terminal móvil debido a que no existe un mecanismo para que el usuario pueda visualizar o borrar la publicidad de manera directa, puesto que las imágenes, videos y textos recibidos se guardan por defecto en el directorio predefinido del móvil, para realizar estas tareas es necesario localizar manualmente los archivos publicitarios. Por lo que debe existir una aplicación cliente para el sistema de envío de publicidad BluePub Sender v1.0 que gestione el proceso de recepción y gestión de la publicidad en el móvil. (3)

Por lo expuesto anteriormente surge el siguiente **problema a resolver**:

¿Cómo gestionar la publicidad recibida en dispositivos móviles con SO Android provenientes del sistema BluePub Sender v1.0?

A partir del problema a resolver se puede definir como **objeto de estudio**: el proceso de recepción de archivos a dispositivos móviles vía Bluetooth, teniendo como **campo de acción**: gestionar los archivos publicitarios proveniente del sistema BluePub Sender a dispositivos móviles con SO Android.

El **objetivo general** es: crear un mecanismo capaz de gestionar la publicidad enviada desde el sistema BluePub Sender v1.0 a dispositivos móviles con SO Android.

Para dar cumplimiento al objetivo general se definieron las siguientes **tareas de la investigación:**

- Análisis y comparación de los sistemas actuales que realizan el envío de publicidad vía Bluetooth.
- Elaboración de un mecanismo de interconexión con el sistema BluePub Sender para el envío de publicidad.
- Definición de la arquitectura de aplicaciones para SO Android.

- Análisis del funcionamiento de la tecnología Bluetooth.
- Descripción sobre el funcionamiento del sistema BluePub Sender v1.0
- Selección de la metodología, tecnología, herramientas y lenguaje de programación a utilizar para desarrollar el sistema.
- Definición de los frameworks a utilizar.
- Selección de librerías para el trabajo con Bluetooth.
- Selección de tecnologías para el trabajo con Android.

Para desarrollar el sistema surgen las siguientes **preguntas científicas:**

- ¿Cómo funciona el proceso de envío de archivos publicitarios del sistema BluePub Sender v1.0 a un dispositivo inalámbrico a través de la tecnología Bluetooth?
- ¿Cómo establecer un mecanismo de conexión directo entre sistema BluePub Sender v1.0 y el sistema Android cliente?
- ¿Es posible realizar una mejor organización de los archivos publicitarios recibidos desde el sistema BluePub Sender v1.0?

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se hace una investigación del estado del arte de los sistemas informáticos que envían publicidad utilizando la tecnología Bluetooth. Se muestra un estudio de las metodologías, lenguajes y herramientas que serán utilizadas para el desarrollo del sistema.

1.2 Estado del Arte

Existen a nivel mundial diversas aplicaciones de envío de publicidad que utilizan la tecnología Bluetooth. A continuación se muestran ejemplos de estos sistemas informáticos.

1.2.1 SEM-BT

Sistema de Envío Masivo vía Bluetooth: es un sistema de transmisión masiva de información y publicidad, el cual ha sido desarrollado por IngSoft I+D+I. El sistema está basado en la tecnología Bluetooth que permite enviar información multimedia (Publicidad, Catálogos, Juegos, etc.) a terminales móviles tales como teléfonos móviles y Personal Digital Assistant (PDAs) encontradas en su radio de acción. (4)

Principales ventajas: (4)

- Este sistema permite hasta 7 conexiones simultáneas.
- Cuenta con un registro de datos donde almacena información sobre los envíos aceptados, rechazados e ignorados.
- Soporta todo tipo de archivos, imágenes, videos, audio, juegos, documentos y programas.
- Tiene un radio de acción de cien metros en espacios abiertos.

El sistema no cuenta con una aplicación que se instale en el móvil para gestionar la publicidad recibida, posee una licencia privativa y no se puede establecer varios servidores de envío en diferentes departamentos de una misma entidad de venta.

1.2.2 Zonablu PC

Zonablu PC: es un software desarrollado para la realización de campañas de marketing de proximidad o acciones de comunicación mediante el envío gratuito de mensajes a dispositivos móviles equipados con Bluetooth. El software permite la configuración de diferentes parámetros para la ejecución de una campaña de marketing Bluetooth o la emisión de uno o múltiples mensajes. (5)

Principales ventajas: (5)

- 7 conexiones simultáneas con diferentes dispositivos móviles.

- Tiempo entre envíos de un archivo a un mismo dispositivo móvil.
- Tiempo entre reintentos de un mensaje no enviado correctamente.
- Tiempo entre envíos si un mensaje fue rechazado.
- Carga de ilimitados archivos de mensaje en diferentes formatos, texto, imágenes, imágenes animadas.
- Contador de dispositivos detectados y envíos.
- Monitorización en tiempo real de la actividad en la zona de cobertura.
- Configuración y almacenaje de múltiples campañas para su posterior emisión.

El software solo funciona con un emisor compatible con USB Dongle Bluetooth además no cuenta con una interfaz o aplicación que se instale en el móvil para gestionar la publicidad recibida, no se puede establecer varios servidores de envío en diferentes departamentos de una misma entidad de venta.

1.2.3 BlueMessenger

BlueMessenger: es un sistema diseñado por Ditecom2 para el envío de publicidad a móviles que tengan activado Bluetooth. Permite crear, modificar y ver las estadísticas de cualquier campaña. (6)

Principales ventajas: (6)

- Envío de cualquier tipo de archivo publicitario: imágenes, videos, música, juegos, programas java.
- Radio de acción de hasta cien metros en espacios abiertos alrededor del servidor de publicidad.
- Los servidores de publicidad BlueMessenger reconocen gran parte de los móviles del mercado, adaptando automáticamente el tamaño de las imágenes para que se ajusten al tamaño de la pantalla de cada móvil.

El software posee una licencia privativa, se ejecuta desde una misma computadora y no contiene una aplicación que se instale en el teléfono para gestionar la publicidad.

1.2.4 AreaBluetooth Light

AreaBluetooth Light: es un software de marketing de proximidad especialmente diseñado para realizar acciones de comunicación mediante el envío gratuito de contenidos multimedia a dispositivos móviles equipados con tecnología Bluetooth. (7)

Principales ventajas: (7)

- 7 conexiones simultáneas con diferentes dispositivos móviles.
- Radio de acción de hasta cien metros en espacios abiertos alrededor del servidor de publicidad.

- Envío de cualquier tipo de archivo.

El software no puede administrar el tráfico de publicidad, posee una licencia privativa, no cuenta con una aplicación que se instale en el teléfono para gestionar la publicidad, no es compatible con todos los adaptadores Bluetooth y no se pueden establecer varios servidores de envío en diferentes departamentos de una misma entidad de venta.

1.2.5 Bluespace

En Cuba existe un sistema llamado Bluespace que es un software de marketing de proximidad mediante antena Bluetooth, que se basa en el envío de contenidos a los teléfonos móviles que se encuentran en el área de influencia del dispositivo emisor. El objetivo fundamental es llegar a las personas, con informaciones valiosas, en el lugar y momento adecuado a través de sus dispositivos móviles. Muy recomendado para eventos, ferias, discotecas y recepciones de las instalaciones. Este sistema es capaz de enviar tipos de contenidos como textos, imágenes, multimedia y tonos para móviles. (8)

Algunas de las ventajas que tiene este sistema son: (8)

- Interfaz simple y cómoda de usar.
- Permite detección y envío simultáneo de contenido a todos los dispositivos móviles ubicados en el área de cobertura Bluetooth.
- Realiza control de la cantidad de dispositivos encontrados, las peticiones aceptadas y rechazadas por los clientes y el número de repeticiones, identificando si el terminal ya ha recibido el contenido.
- Una vez aceptado el contenido no es vuelto a enviar.
- Es muy liviano y consume muy pocos recursos.

El software posee una licencia privativa y no brinda la posibilidad de poder establecer varios servidores de envío en diferentes departamentos de una misma entidad de venta.

1.2.6 BluePub Sender v1.0

En la UCI existe un software denominado BluePub Sender v1.0 que se encarga del envío de publicidad hacia los dispositivos móviles con tecnología Bluetooth. (9)

Las ventajas que presenta este sistema son: (10)

- Envío de cualquier tipo de archivo.
- Permite detección y envío simultáneo de contenido a todos los dispositivos móviles ubicados en el área de cobertura Bluetooth.

- Una vez aceptado el contenido no es vuelto a enviar hasta un tiempo determinado.
- Cuenta con un registro de datos donde almacena información sobre los envíos aceptados, rechazados e ignorados.
- Se pueden establecer varios servidores de envío de publicidad distribuidos por departamentos para una misma entidad de venta.
- Se pueden controlar los servidores de envío permitiendo iniciar, detener y agregar nuevas publicidades a servidores por separado.

Sin embargo no se realiza una correcta gestión de la publicidad en el terminal móvil puesto que las imágenes y videos recibidos, se almacenan por defecto en las carpetas de imágenes y videos predefinidos en el terminal, por lo que el proceso de visualización en otro momento que no sea el de recepción se dificulta. Además no existe un mecanismo para que el usuario pueda borrar o buscar la publicidad recibida de forma directa.

1.2.7 Valoración de sistemas existentes para el envío de publicidad

Luego de analizar los sistemas existentes, se llega a la conclusión de que no cumplen de manera general todas las características requeridas ya que no cuentan con una aplicación que reciba y gestione los archivos publicitarios en el teléfono móvil, estando algunos bajo licencias privativas y otros son sistemas que se encuentran ubicados en una misma computadora, y no pueden establecer varios servidores de envío en diferentes departamentos de una misma entidad de venta. El sistema BluePub Sender posee la mayoría de las especificaciones necesarias, por tanto es necesario desarrollar una aplicación que se ejecute en el teléfono para la gestión del archivo publicitario preveniente del sistema BluePub Sender.

1.3 Tecnologías a utilizar

A medida que avanza el desarrollo de la humanidad, también avanza la tecnología en el mundo como resultado de las necesidades de las personas. Para implementar el sistema informático es necesario contar con las tecnologías que permiten llegar a la solución del problema y favorezcan a aumentar la calidad, confiabilidad y seguridad de la aplicación.

1.3.1 Bluetooth

La tecnología Bluetooth es una tecnología de comunicación por ondas de radio de corto alcance que es simple, segura, y fácil de usar. Posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda

ISM de los 2,4 Jhs. Los dispositivos Bluetooth también pueden clasificarse según su ancho de banda, llegando a alcanzar en la versión 4.0 24mb/s y en la versión 2.0 3mb/s aunque en la práctica se define un canal de comunicación de máximo 720 Kb/s (1 Mb/s de capacidad bruta) con rango óptimo de 10 m. El hardware que compone el dispositivo Bluetooth está compuesto por dos partes: (11)

- Un dispositivo de radio: encargado de modular y transmitir la señal.
- Un controlador digital: compuesto por una CPU, por un procesador de señales digitales (DSP - Digital SignalProcessor) llamado Link Controller (o controlador de Enlace) y de las interfaces con el dispositivo anfitrión.

El LC o Link Controller está encargado de hacer el procesamiento de la banda base y del manejo de los protocolos automatic repeat request (ARQ) y forward error correction (FEC) de capa física. Además, se encarga de las funciones de transferencia (tanto asíncrona como síncrona), codificación de Audio y cifrado de datos.

La unidad de procesamiento central o CPU del dispositivo, se encarga de atender las instrucciones relacionadas con Bluetooth del dispositivo anfitrión, para así simplificar su operación. Para ello, sobre el CPU corre un software denominado Link Manager Protocol (LMP) que tiene la función de comunicarse con otros dispositivos por medio de dicho protocolo.

La información que se intercambia entre dos unidades Bluetooth se realiza mediante un conjunto de ranuras que forman un paquete de datos. Cada paquete comienza con un código de acceso de 72 bits, que se deriva de la identidad maestra, seguido de un paquete de datos de cabecera de 54bits. Éste contiene importante información de control, como tres bits de acceso de dirección, tipo de paquete, bits de control de flujo, bits para la retransmisión automática de la pregunta, y chequeo de errores de campos de cabecera. La dirección del dispositivo es en forma hexadecimal. Finalmente, el paquete que contiene la información, que puede seguir al de la cabecera, tiene una longitud de 0 a 2745 bits.

Si un equipo se encuentra dentro del radio de cobertura de otro, éstos pueden establecer conexión entre ellos. Cada dispositivo tiene una dirección única de 48 bits, basada en el estándar IEEE (Institute of Electrical and Electronics Engineers) 802.11 para WLAN (Wireless Local Area Network). En principio sólo son necesarias un par de unidades con las mismas características de hardware para establecer un enlace. Dos o más unidades Bluetooth que comparten un mismo canal forman una piconet. (12)

Para una mejor comprensión de lo antes explicado se muestra a continuación la siguiente imagen:

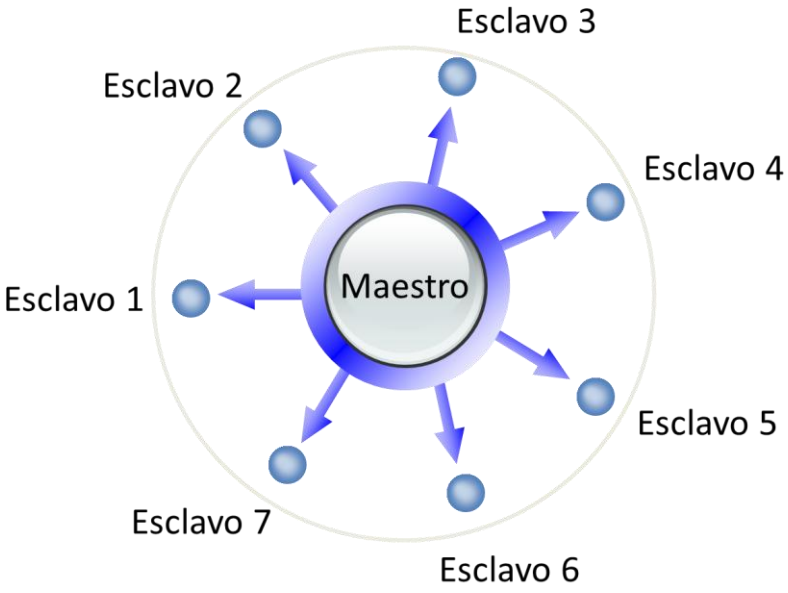


Figura 1.1Diagrama de red Piconet

1.3.2 Lenguaje de programación

Un lenguaje de programación está diseñado para facilitar el entendimiento del programador con el equipo de cómputo. Está formado por un conjunto de reglas lógicas, sintácticas y símbolos que definen su estructura y el significado de sus expresiones. Para desarrollar un programa informático es imprescindible la necesidad de un lenguaje de programación. (13)

1.3.2.1 Java

Para el desarrollo de la presente investigación se selecciona Java como lenguaje de desarrollo debido a que:

- El lenguaje tiene un modelo de objetos muy simple que elimina herramientas de bajo nivel, que suelen inducir a errores de programación. Un objeto es la forma de simular el comportamiento y el estado de las entidades lo cual propicia que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad.
- Distribuido: proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets, establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.
- Robusto: diseñado para crear software altamente fiable. Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.

- Seguro: dada la naturaleza distribuida de Java, la seguridad se impuso como una necesidad de vital importancia. Se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real además no ofrece acceso directo al hardware de la arquitectura ni al espacio de direcciones.
- Posee múltiples beneficios y propiedades que los hacen idóneo para el desarrollo de aplicaciones Android. (14)

1.3.3 Lenguaje de modelado.Unified Modeling Language (UML)

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (15) (16)

Sus principales ventajas son:

- Se puede usar para modelar distintos tipos de sistemas: software, hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar distintos sistemas.
- Es una consolidación de muchas de las notaciones y conceptos más usados.

1.3.4 Frameworks a utilizar

Un framework o marco de trabajo define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

1.3.4.1 Spring framework

Spring Framework es un marco de trabajo código abierto de desarrollo de aplicaciones para la plataforma Java. Por su diseño ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes.

Algunas características de Spring: (17)

- Ligero: la mayor parte de Spring Framework se puede distribuir en un único archivo .jar que pesa poco más de 2,5MB.
- Orientado a aspectos: ofrece soporte para programación Orientada a Aspectos (AOP) que permite el desarrollo cohesionado por la separación de la aplicación, la lógica de negocio de los servicios del sistema (tales como la auditoría y la gestión de transacciones).

- Configurable: permite configurar y componer complejas solicitudes de los componentes más simples. Los objetos de aplicación son compuestos de forma declarativa, por lo general en un archivo Extensible Markup Language (XML).
- Flexible: diseñado como una serie de módulos que pueden trabajar independientemente uno de otro. Además intenta mantener un mínimo acoplamiento entre la aplicación y el propio Framework de forma que podría ser desvinculada de él sin demasiada dificultad.

1.3.5JDBC (Java Database Connectivity)

JDBC es una API de Java para ejecutar sentencias SQL (Structured Query Language). Consta de un conjunto de clases e interfaces escrito en lenguaje de programación Java. Permite enviar sentencias SQL virtualmente a cualquier base de datos relacional. Se puede escribir un solo programa usando la API JDBC y será capaz de enviar sentencias SQL a la base de datos apropiada. JDBC extiende lo que puede hacerse con Java. Posibilita establecer una conexión con una base de datos; enviar sentencias SQL y procesar los resultados. (18)

Dentro de los objetivos de la filosofía JDBC se encuentran:

- Proveer una interfaz homogénea al resto de APIs de Java.
- Ser un API simple, y desde ahí, ir creciendo.
- Mantener los casos comunes de acceso a Base de Datos lo más sencillo posible.
  - Conservar la sencillez en los casos más comunes (SELECT, INSERT, DELETE y UPDATE).
  - Hacer realizables los casos menos comunes: Invocación de procedimientos almacenados, entre otros.
- JDBC prefiere incluir gran cantidad de métodos, en lugar de hacer métodos complejos con gran cantidad de parámetros.

Teniendo en cuenta que el sistema se desarrollará en lenguaje Java y que la tecnología JDBC es el mecanismo por excelencia de acceso a datos dentro de este lenguaje se decide utilizar esta API para el acceso a los datos.

1.3.6Socket

Un socket designa un concepto abstracto por el cual dos aplicaciones pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada. Es

un método para la comunicación entre un programa cliente y un programa servidor en una red. Finalmente queda definido por un par de direcciones Internet Protocol (IP) local y remota, un protocolo de transporte y un par de números de puerto local y remoto. Para efectuar la comunicación entre dos programas es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro.
- Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

Un sistema de comunicación por socket necesita de dos entidades bien diferenciadas: el Servidor y el Cliente. (19)

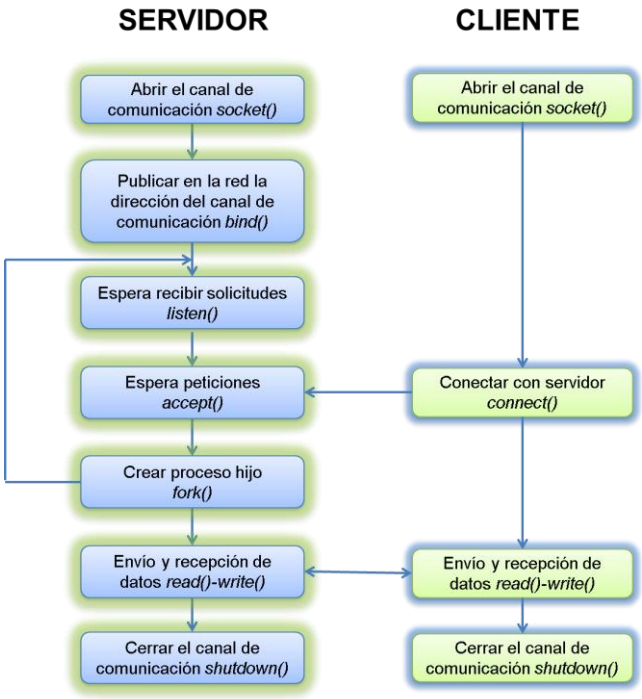


Figura 1.2 Funcionamiento de una conexión socket.

Servidor: se ejecuta sobre una computadora específica y tiene un socket que responde en un puerto específico. El servidor únicamente espera, escuchando a través del socket a que un cliente haga una petición.

Cliente: conoce el nombre de host de la máquina en la cual el servidor se encuentra ejecutando y el número de puerto en el cual el servidor está conectado. Para realizar una petición de conexión, el cliente intenta encontrar al servidor en la máquina servidora en el puerto especificado.

1.3.7API para Bluetooth. BlueCove 2.1.0

BlueCove es una implementación JSR-82 (Java Specification Request) de J2SE (Java Standard Edition). Originalmente desarrollado por Intel Research. BlueCove se ejecuta en cualquier JVM (Java Virtual Machine) a partir de la versión 1.1 en los sistemas operativos Windows Mobile, Windows XP y Windows Vista, Mac OS. Desde la versión 2.1 BlueCove distribuye bajo la Apache Software License, versión 2.0. (20)

1.3.8 SO Android v2.3

Android es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes y tablets. A medida que fue creciendo su historial de versiones también mejoró el nivel de comportamiento y las prestaciones que brida el sistema operativo, se seleccionó la versión 2.3 por prevalecer sobre las demás versiones con cerca de un 50 % de uso, estadística realizada en febrero del 2013 (21). La siguiente tabla presenta los por ciento de uso de las versiones de Android:

Versión	Nombre	API	Distribución
1.6	Donut	4	0.2 %
2.1	Eclair	7	2.2 %
2.2	Froyo	8	8.1 %
2.3-2.3.2	Gingerbread	9	0.2 %
2.3.3-2.3.7		10	45.4 %
3.1	Honeycomb	12	0.3 %
3.2		13	1.0 %
4.0.3-4.0.4	Ice Cream Sandwich	15	29 %
4.1	Jelly Bean	16	12.2 %

Tabla 1.1Distribución de Versiones de Android

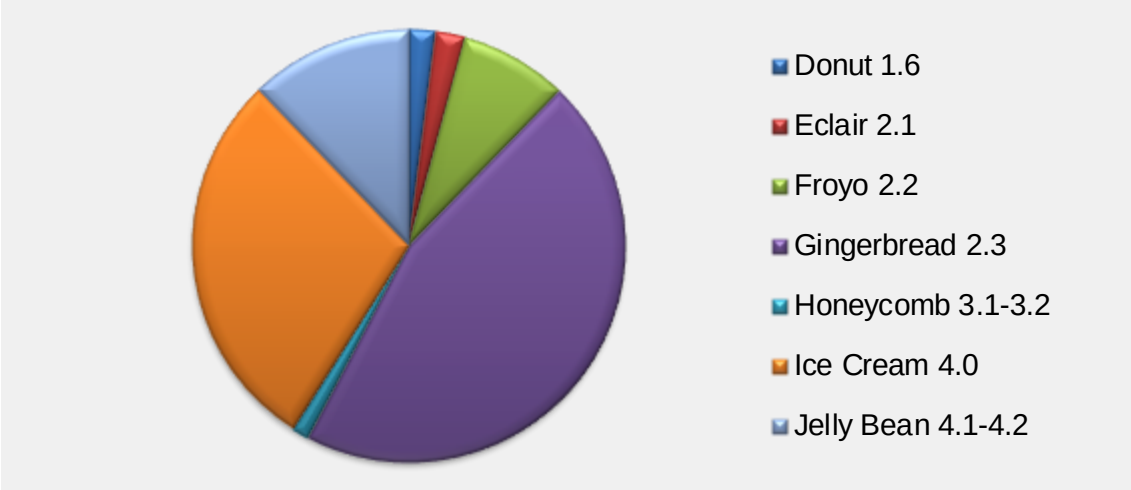


Figura 1.3 Distribución de Versiones de Android

Entre sus principales mejoras se encuentran:

- Soporte para dispositivos móviles.
- Actualización del diseño de la interfaz de usuario.
- Soporte para pantallas extra grandes y resoluciones WXGA y mayores.
- Soporte nativo para telefonía VoIP SIP.
- Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC.
- Nuevos efectos de audio como reverberación, ecualización, visualización de los auriculares y refuerzo de graves.
- Soporte para Near Field Communication.
- Funcionalidades de cortar, copiar y pegar disponibles a lo largo del sistema.
- Teclado multi-táctil rediseñado.
- Soporte mejorado para desarrollo de código nativo.
- Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos.
- Recolección de elementos concurrentes para un mayor rendimiento.
- Soporte nativo para más sensores (como giroscopios y barómetros).
- Un administrador de descargas para descargar archivos grandes.
- Administración de la energía mejorada y control de aplicaciones mediante el administrador de tareas.
- Soporte nativo para múltiples cámaras.
- Cambio de sistema de archivos de YAFFS a ext4.

(22)

1.4Metodología de desarrollo

Las metodologías de desarrollo surgen por la necesidad de organizar el proceso de desarrollo de software. Definen una serie de parámetros que guían al equipo de desarrollo ofreciendo técnicas, herramientas y soporte documental en dependencia de las necesidades requeridas. Cada metodología de desarrollo de software tiene su propio enfoque para la evolución del proceso de software cuyo objetivo final es lograr un producto satisfactorio. (23)

1.4.1 FDD Feature Driven Development

FDD es una metodología ágil, iterativa y adaptativa. A diferencia de otras metodologías ágiles no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción y se considera adecuado para proyectos mayores y de misión crítica. No requiere un modelo específico de proceso y se complementa con otras metodologías. Enfatiza cuestiones de calidad y define claramente entregas tangibles y formas de evaluación del progreso.

La metodología se divide en cinco fases las cuales se muestran en la siguiente imagen

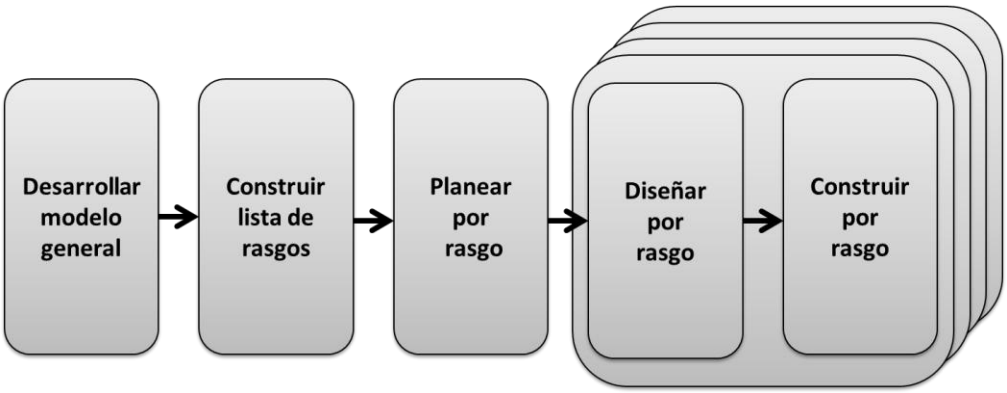


Figura.1.4 Fases de la metodología FDD.

Teniendo en cuenta las características del software que se desea desarrollar se decide usar esta metodología por adaptarse mejor a proyectos cortos y equipos pequeños además, se cuenta con experiencia en el trabajo con la misma. Genera una cantidad de documentación aceptable lo que brinda la posibilidad de contar con más tiempo para la implementación y no se basa en formalismos en la documentación, sino en controles propios y una comunicación fluida con el cliente. (24)

1.5 Herramientas

Para potenciar el desarrollo del software se hace necesario seleccionar las herramientas de trabajo que contribuyen a aumentar la calidad, confiabilidad y seguridad la aplicación.

1.5.1 Entorno Integrado de Desarrollo (IDE). Eclipse 3.5.0

El entorno de desarrollo Eclipse 3.5.0 combina una potente y completa plataforma de programación con elementos de desarrollo y compilación variados, donde la compilación es en tiempo real.

En cuanto a la utilización de Eclipse para la creación de aplicaciones clientes se puede decir que provee al programador con frameworks para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software y aplicaciones web. Incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. Permite la integración con la herramienta de desarrollo de software Android SDK.

Se decide utilizar Eclipse 3.5.0 como plataforma de desarrollo del software. (25)

1.5.2 Servidor Web. Apache Tomcat 6.0.26

Apache Tomcat es una implementación de software de código con soporte de servlets y Java Server Pages (JSPs) que puede funcionar como servidor web por sí mismo. Desarrollado en un entorno abierto y participativo gestiona solicitudes y respuestas Hyper Text Transfer Protocol (HTTP); además es servidor de aplicaciones o contenedor de Servlets/JSP.

Se selecciona el servidor web Apache Tomcat por la necesidad de usar aplicaciones distribuidas en el sistema debido a que permite el trabajo con Servlets, es fácil de configurar, fiable y seguro. Funciona en todos los sistemas operativos que dispongan de la máquina virtual de Java y por sus características favorece el desarrollo de la aplicación. (26)

1.5.3 Sistema Gestor de Base de Datos. PostgreSQL 9.1.0

PostgreSQL es un sistema de gestión de bases de datos relacional basado en código abierto orientado a objetos. Posee una alta concurrencia ya que permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Incluye características como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional lo cual lo define como sistema objeto relacional. Algunas de las características principales del gestor de bases de datos son:

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes, cadenas de bits, entre otros. También permite la creación de tipos propios.

- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

(27)

1.5.4 Gestor de Base de Datos. SQLite

SQLite es una base de datos de código abierto, instalada por defecto en dispositivos con SO Android. Utilizar SQLite no requiere configuración, no tiene un servidor de base de datos ejecutándose en un proceso separado y es relativamente simple su empleo. Por otro lado SQLite es fácil de implementar en una aplicación Android además se puede migrar o compartir datos con otras bases de datos populares a través de drivers o manejadores de bases de datos. (28)

1.5.5 Herramienta de modelado. Visual Paradigm 8.0

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software; ayuda a una rápida construcción de aplicaciones de calidad, y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (29)

Algunas de las características son: (29)

- Entorno de creación de diagramas para UML 2.1.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

Se selecciona esta herramienta como herramienta de modelado teniendo en cuenta todas sus ventajas y además se cuenta con experiencia en el trabajo con la misma.

1.5.6Android SDK (Kit[1] de Desarrollo de Software) 2.0

Incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. La plataforma integral de desarrollo soportada oficialmente es Eclipse junto con el complemento Android Development Tools plugin (ADT), aunque también puede utilizarse un editor de texto para escribir ficheros Java y Xml y permite utilizar comandos en un terminal para crear y depurar aplicaciones. (30)

CONCLUSIONES PARCIALES

En el presente capítulo se realizó una introducción sobre los contenidos que se abordan en el trabajo, se hizo un estudio de las aplicaciones de envío de publicidad existentes y se planteó la necesidad de desarrollar un sistema que gestione los archivos publicitarios provenientes del sistema de envío de publicidad BluePub Sender para teléfonos móviles que posean SO Android v2.3,se decide seleccionar esta versión debido a que contiene las características necesarias para implementar el sistema cliente y predomina el uso de esta en el mercado. Además se definieron las herramientas, tecnologías y metodologías a utilizar más favorables y que se ajusten mejor al desarrollo del software requerido.

[1] Conjunto de herramientas que sirven para un mismo uso

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se realiza la descripción de la solución propuesta, se presenta un modelo de dominio para un mejor entendimiento de la solución planteada. Este modelo de dominio contiene la información necesaria: las principales entidades del dominio y sus relaciones. Se documenta la propuesta de solución, construyéndose una lista de rasgos que representará las características o funcionalidades que los componentes deben cumplir lo que nos permite hacer una concepción general del sistema, Una vez creada la lista de rasgos se planifica cada rasgo para la posterior fase iterativa de diseño, construcción y prueba.

2.2 Situación Problemática

La publicidad es un área muy importante para el desarrollo del comercio de muchas empresas y centros comerciales. Los medios de difusión de publicidad son componentes esenciales en esta tarea. La publicidad por proximidad para teléfonos móviles utilizando la tecnología Bluetooth resulta una de las formas más certeras de promover un producto o servicio. (31)

El sistema de envío de publicidad BluePub Sender v1.0 ofrece el servicio de envío de publicidad utilizando como canal de transmisión para el envío de contenidos la tecnología Bluetooth sin embargo carece de un sistema cliente que se ejecute en el terminal móvil y permita la visualización directa de los archivos publicitarios tales como: imágenes, videos o textos los cuales son almacenados por defecto en el directorio predefinido del teléfono móvil por lo que no se pueden buscar o borrar directamente. Esto dificulta que la publicidad sea visualizada utilizando la vía más rápida, este inconveniente disminuye la eficacia del anuncio publicitario ya que no es suficiente con tenerla almacenada en el móvil si no es vista en el momento y lugar apropiado.

Por todo lo antes expuesto se hace necesario desarrollar un sistema cliente que permita la gestión de los archivos publicitarios para teléfonos celulares con tecnología Bluetooth y que posean SO Android v2.3.

2.3 Objeto de automatización

Se desea automatizar la gestión y organización de los archivos publicitarios almacenados en el móvil y el proceso de envío de un archivo publicitario a otro dispositivo móvil utilizando la tecnología Bluetooth.

2.4 Propuesta del sistema

Se propone desarrollar un sistema cliente que permita interactuar con el sistema servidor de envío de publicidad. Se deben hacer modificaciones en el sistema servidor existente BluePub Sender v1.0 con el objetivo de establecer un mecanismo de interconexión entre ambos sistemas. Además debe permitir diferenciar los terminales que contienen el sistema cliente BluePub Sender para Android de los terminales que no lo poseen.

El sistema cliente debe permitir organizar los archivos publicitarios almacenados en el móvil, anteriormente enviados desde el sistema servidor BluePub Sender, para ser visualizados o borrados.

Estas operaciones se realizarán de manera sencilla, teniendo en cuenta que el sistema cliente se desarrollará para teléfonos móviles táctiles, las interfaces del sistema estarán enfocadas al trabajo con teléfonos celulares siguiendo los patrones de diseño más usados en este ámbito. (32)

El sistema informará al usuario con una notificación de nuevas publicidades recibidas y del estado de la conexión con el servidor.

2.4.1 Descripción de la propuesta del sistema

El teléfono móvil que posee el sistema cliente de BluePub Sender intenta establecer comunicación con el servidor de envío de publicidad, cuando se establece la conexión el sistema cliente procede a realizar las peticiones de anuncios publicitarios. Por parte del servidor este comprueba los anuncios publicitarios que no han sido enviados al teléfono móvil, selecciona la publicidad y la envía, una vez que ha enviado todos los anuncios publicitarios, espera un tiempo determinado y reinicia el ciclo de envío nuevamente. El teléfono móvil recibe la publicidad, es almacenada en el terminal y el usuario es notificado del recibo, posteriormente puede visualizar o eliminar el archivo publicitario. El usuario del sistema cliente puede además buscar las publicidades almacenadas en el móvil mediante diferentes criterios de búsqueda: nombre del departamento emisor, fecha de recibo, nombre de la publicidad, preferencia y tipo de archivo publicitario.

Figura 2.1Propuesta del Sistema.

2.5 Modelo de Dominio

Para desarrollar el modelo de dominio se seleccionan y abstraen las funciones y objetos específicos, se identifican las características comunes y se establecen las relaciones fundamentales. Se propone realizar un modelo de dominio debido a que los procesos de negocio no son claramente visibles y para ayudar a comprender los conceptos que se manejan en el dominio del sistema se muestra la siguiente descripción del mismo.

2.5.1 Descripción del Modelo de Dominio

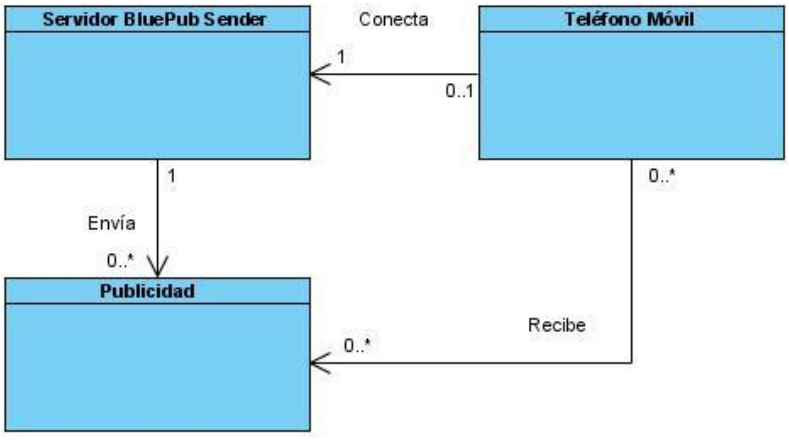


Figura 2.2 Diagrama de Modelo de Dominio.

El sistema servidor de envío de publicidad BluePub Sender v1.0 selecciona la publicidad y la envía al teléfono móvil vinculado mediante Bluetooth.

2.5.2 Descripción de los objetos del dominio

Descripción de los objetos que conforman el modelo de domino:

- Servidor BluePub Sender: Sistema que se encarga de enviar la publicidad a los dispositivos móviles.
- Publicidad: Representa el archivo publicitario (imagen, música, video, texto).
- Teléfono Móvil: Teléfono móvil que posee Bluetooth y posee la capacidad de conectarse con el sistema BluePub Sender.

2.6Relación de las funcionalidades

Para establecer los procedimientos con que cuenta el sistema se plantean los siguientes requisitos funcionales, los cuales determinan su funcionamiento y definen una serie de condiciones y capacidades de interés para el usuario.

2.6.1 Construcción de la lista de funcionalidades

Área temática	Actividades	Funcionalidades
Sistema Cliente	Recibir Publicidad	1 Enviar petición de conexión
		2 Conectar servidor de envío
		3 Enviar petición de recibo
		4 Guardar archivo publicitario
		5 Enviar notificación al SO
	Gestionar Publicidad	6 Buscar archivo publicitario
		7 Mostrar archivo publicitario
		8 Actualizar preferencia de la publicidad
		9 Mostrar propiedades de la publicidad
		10 Eliminar publicidad
	Controlar Estado	11 Verificar estado de conexión
		12 Eliminar publicidades antiguas
Sistema Servidor	Enviar Publicidad	1 Crear conexión
		2 Aceptar solicitud de conexión
		4 Seleccionar publicidad
		5 Enviar publicidad
		6 Registrar dispositivo
		7 Registrar envío de publicidad

Tabla 2.1 Lista de Funcionalidades.

2.6.2Planeación por funcionalidad

En las siguientes tablas se encuentran planificados los rasgos conforme a su prioridad y dependencia. A la hora de planear cada rasgo se tuvo en cuenta que dentro de una iteración no pueden existir rasgos que dependan de otros que no han sido implementados, con la excepción de que puede depender de rasgos no implementados, pero que todos estos se encuentren en la misma iteración.

Recibir Publicidad				Rasgos : 5
Funcionalidad	Modelo general	Diseño	Implementación	Prueba
	Plan	Plan	Plan	Plan
Enviar petición de conexión	19/03/2013	20/03/2013	24/03/2013	26/05/2013
Conectar servidor de envío	19/03/2013	20/03/2013	21/03/2013	26/05/2013
Enviar petición de recibo	19/03/2013	21/03/2013	24/03/2013	26/05/2013
Guardar archivo publicitario	19/03/2013	20/03/2013	23/03/2013	26/05/2013
Enviar notificación al SO	19/03/2013	21/03/2013	21/03/2013	21/03/2013

Tabla 2.2 Planeación de la funcionalidad: Recibir Publicidad.

Gestionar Publicidad				Rasgos : 5
Funcionalidad	Modelo general	Diseño	Implementación	Prueba
	Plan	Plan	Plan	Plan
Buscar archivo publicitario	20/03/2013	25/03/2013	28/03/2013	30/03/2013
Mostrar archivo publicitario	21/03/2013	26/03/2013	28/03/2013	02/04/2013
Actualizar preferencia de la publicidad	21/03/2013	26/03/2013	28/03/2013	02/04/2013
Mostrar propiedades de la publicidad	21/03/2013	26/03/2013	28/03/2013	02/04/2013

Eliminar publicidad	21/03/2013	26/03/2013	28/03/2013	03/04/2013

Tabla 2.3 Planeación de la funcionalidad: Gestionar Publicidad.

Controlar Sistema				Rasgos : 2
Funcionalidad	Modelo general	Diseño	Implementación	Prueba
	Plan	Plan	Plan	Plan
Verificar estado de conexión	20/03/2013	25/03/2013	28/03/2013	03/06/2013
Eliminar publicidades antiguas	21/03/2013	26/03/2013	28/03/2013	03/04/2013

Tabla 2.4 Planeación de la funcionalidad: Controlar Sistema.

Enviar Publicidad				Rasgos : 7
Funcionalidad	Modelo general	Diseño	Implementación	Prueba
	Plan	Plan	Plan	Plan
Crear conexión	19/03/2013	20/03/2013	24/03/2013	26/05/2013
Aceptar solicitud de conexión	19/03/2013	20/03/2013	21/03/2013	26/05/2013
Aceptar solicitud de conexión	20/03/2013	20/03/2013	21/03/2013	04/06/2013
Seleccionar publicidad	20/03/2013	25/03/2013	28/03/2013	04/06/2013
Enviar publicidad	21/03/2013	26/03/2013	28/03/2013	04/06/2013
Verificar tipo de dispositivo	06/04/2013	10/04/2013	24/04/2013	04/06/2013
Registrar envío de publicidad	21/03/2013	26/03/2013	28/03/2013	04/06/2013

Tabla 2.5 Planeación de la funcionalidad: Enviar Publicidad.

2.6.3Requisitos no funcionales del sistema cliente BluePub Sender v2.0

Los requerimientos no funcionales son cualidades con las que debe contar el producto a fin de hacerlo atractivo, usable, rápido y/o confiable; esto podría marcar la diferencia entre un producto bien aceptado y otro no tan aceptado. Los requisitos no funcionales presentes son:

RNF 1. Apariencia o interfaz externa:

La aplicación propuesta poseerá una interfaz sencilla, amigable y fácil de utilizar, dirigida directamente al usuario del sistema y a dispositivos con pantalla táctil.

RNF 2. Software:

El móvil con el sistema BluePub Sender v2.0 cliente requiere del SO Android v2.3 o superior.

RNF 3 Hardware:

Requisitos recomendados:

- CPU ARM 1 GHz
- RAM 256 MB.
- ROM 1 GB.
- Bluetooth v2.0.

RNF 4 Usabilidad:

El sistema cliente a desarrollar deberá ser fácil de usar por lo que no es necesario que el usuario posea conocimientos avanzados en tecnología móvil.

RNF5 Rendimiento:

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos en consecuencia con el Hardware recomendado para el sistema.

CONCLUSIONES PARCIALES

En el presente capítulo se expuso la descripción de la propuesta del sistema, se mostró el modelo de dominio, se definieron la lista de funcionalidades y los requisitos no funcionales que debe cumplir el sistema cliente y el sistema servidor, y se planeó en función de la complejidad y dependencia de cada funcionalidad el tiempo para dar cumplimiento a la misma.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

Durante este capítulo se describen los procesos que se llevan a cabo durante la fase diseñar en base a las funcionalidades, por lo que se exponen la arquitectura del sistema y los patrones de diseño. Además se muestran los diagramas de paquetes, de clases del diseño, de secuencia y el diagrama de entidad relación de la base de datos.

3.2 Arquitectura

La arquitectura de software es el esqueleto o base de una aplicación, que define estilos y patrones arquitectónicos seleccionados de forma adecuada para satisfacer la mayor funcionalidad y desempeño del sistema. (23)

3.2.1 Arquitectura Cliente-Servidor

La arquitectura Cliente-Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Normalmente el servidor es una máquina que actúa de depósito de datos. Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red; en este caso la vía de comunicación entre ambas partes será el canal de comunicación inalámbrica Bluetooth. (33)

Para comprender con mayor claridad lo explicado anteriormente se muestra la siguiente imagen:

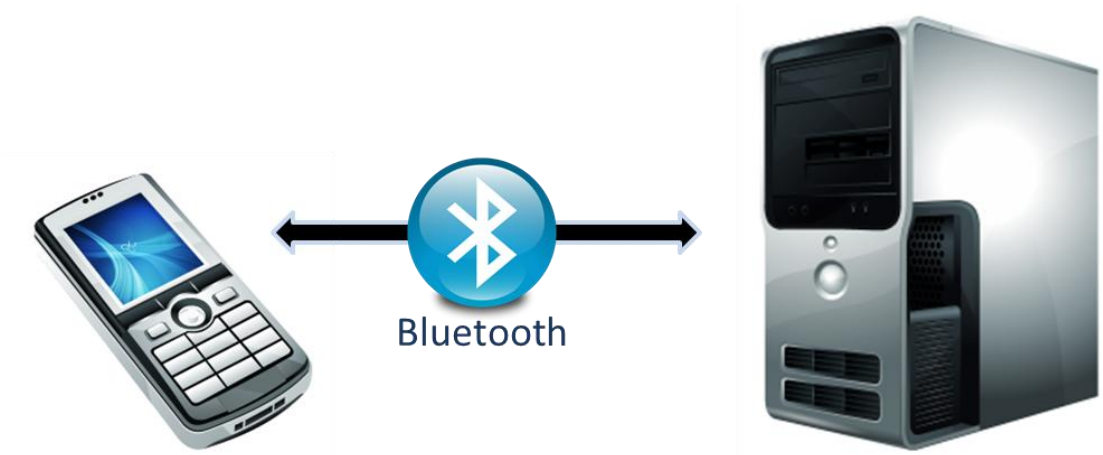


Figura 3.1 Descripción de la arquitectura Cliente-Servidor.

Elementos Principales de la Arquitectura Cliente Servidor

Cliente: es un sistema que permite al usuario formular los requerimientos y pasarlos al servidor. Éste normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario, además de acceder a los servicios distribuidos en cualquier parte de la red. Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos: (34)

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir y mostrar resultados del servidor.
- Formatear resultados.

Servidor: es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Las principales funciones que lleva a cabo el proceso servidor se enumeran a continuación: (34)

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

3.2.2 Arquitectura MVP (Modelo-Vista-Presentador)

Este patrón arquitectónico es un derivado del patrón Modelo Vista Controlador (MVC). Por un lado, la vista, que se encarga de mostrar la información al usuario y de interactuar con él para hacer ciertas operaciones, por otro lado, tenemos el modelo que, ignorante de cómo la información es mostrada al usuario, realiza toda la lógica de las aplicaciones usando las entidades del dominio, y por último tenemos al presentador que es el que “presenta” a ambos actores sin que haya ningún tipo de dependencia entre ellos. (35)

A continuación se muestra una imagen del patrón Modelo-Vista-Presentador:

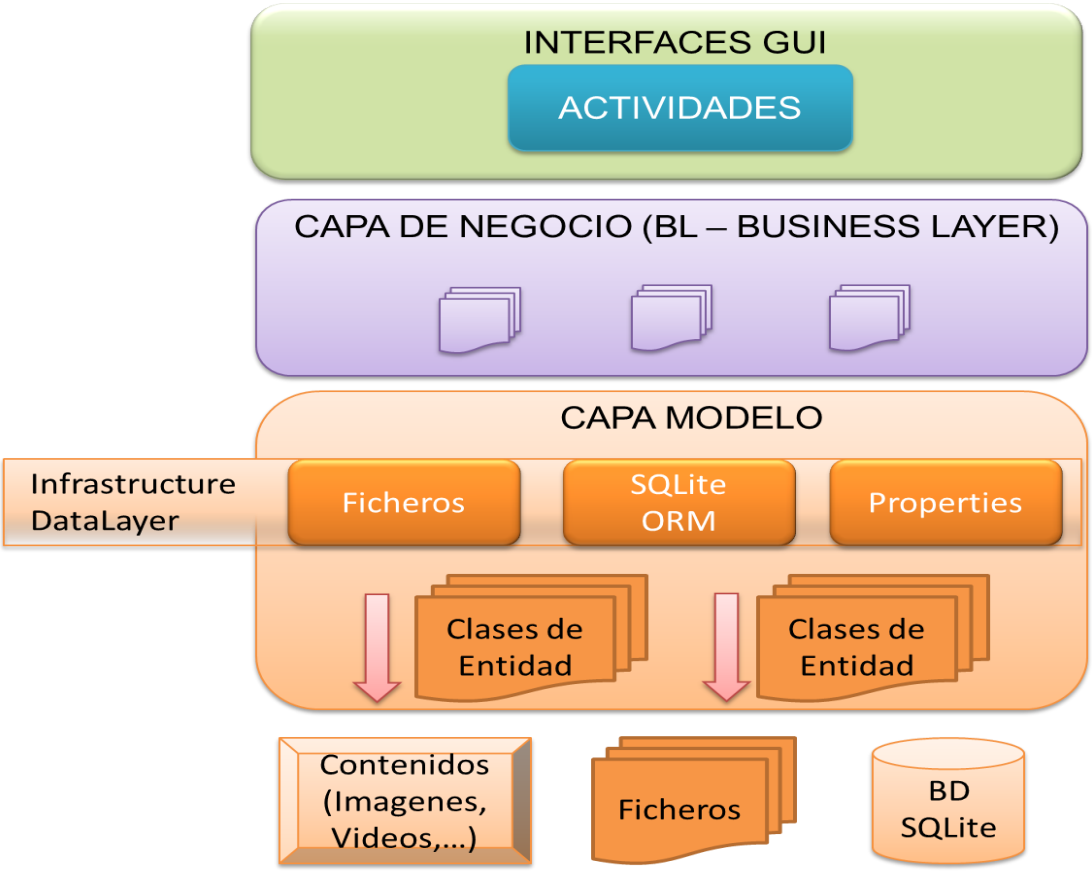


Figura 3.2 Descripción de la arquitectura Modelo Vista Presentador.

En la imagen se puede observar la siguiente separación de capas: (35)

- Capa vista: incluye las interfaces de usuario (actividades, recursos, layouts, etc...), o aquellos artefactos de Android sin GUI (Interfaz Gráfica de Usuario) como servicios o brocadas receivers.
- Capa presentador-(Capa de negocio): incluye nuestras clases de negocio, hacen de puente entre la vista y el modelo, incluyendo clases con métodos que alberguen la lógica de nuestra aplicación. Siempre los artefactos de la capa vista invocarán a las clases albergadas en esta capa, nunca accederán al modelo directamente.
- Capa modelo: incluye las clases
  - ✓ Clases de Entidad: incluyen la definición lógica del modelo de datos con el que vamos a trabajar. Estas clases serán manejadas por la capa de negocio.
  - ✓ Capa Infraestructura: abstrae la complejidad tecnológica de la gestión del almacenamiento físico. Como puede apreciarse en la imagen existen

diversas formas en Android de manejar información, desde content providers, ficheros, bases de datos, hasta webservices.

- En la base de la pila de capas se encuentran las fuentes físicas de información: ficheros en SD / RAM interna, bases de datos, contenidos gestionados por otras aplicaciones, servicios web.

3.3 Patrones de Diseño

Un patrón es un conjunto de información que proporciona respuesta a un conjunto de problemas similares, es decir, es una solución a un problema en un una situación recurrente a la que es posible aplicar el patrón. Los patrones de diseño son aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.

3.3.1 Patrones Grasp (General Responsability Assignment Software Patterns)

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades con el objetivo de seguir buenas prácticas de aplicación recomendable en el diseño de software. (36)

3.3.1.1 Experto

Este patrón define que la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y da la oportunidad de reutilizar los componentes en futuras aplicaciones. (36 p. 6)

La siguiente figura muestra la utilización del patrón Experto en el desarrollo del sistema:

```
public class Cliente implements Parcelable {

    public Cliente(Context contex) {
    public ArrayList<String> getDepartamentos() {
    public ArrayList<Publicidad> getPublicidadNombre() {
    public ArrayList<Publicidad> getPublicidadPreferencia() {
    public ArrayList<Publicidad> getPublicidadFecha(String fecha1, String fecha2) {
    public ArrayList<Publicidad> getPublicidadDepartamento(String departamento) {
    private boolean existePublicidad(String direccion) {
    public Publicidad getUltimaPublicidad(int n) {
    public int describeContents() {
    public void writeToParcel(Parcel arg0, int arg1) {

}
```

Figura 3.3 Ejemplo de utilización del patrón experto.

3.3.1.2 Creador

El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que:

- Tiene la información necesaria para realizar la creación del objeto, o
- Usa directamente las instancias creadas del objeto, o
- Almacena o maneja varias instancias de la clase
- Contiene o agrega la clase.

(36 p. 9)

La siguiente figura muestra la utilización del patrón Creador en el desarrollo del sistema:

```
public class I_Principal extends Activity {

    Cliente cliente;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_i__principal);

        cliente = new Cliente(this);
    }
```

Figura 3.4 Ejemplo de utilización del patrón creador.

3.2.1.3 Bajo Acoplamiento

Plantea mantener las clases lo menos ligadas entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las mismas. (36 p. 13)

3.3.1.4 Alta Cohesión

Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Mejora la claridad y facilidad con que se entiende el diseño, se simplifica el mantenimiento y las mejoras de funcionalidad. (36 p. 16)

3.3.1.5 Controlador

Plantea asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades. El controlador no

realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. (36 p. 18)

La siguiente figura muestra la utilización del patrón Controlador en el desarrollo del sistema:

```
public class I_Buscar extends Activity {
    Cliente cliente;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.buscar);
        cliente = new Cliente(this);
        final Button fecha = (Button) findViewById(R.id.buttonfecha);
        fecha.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
        });
        final Button departamento = (Button) findViewById(R.id.buttondepartamento);
        departamento.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
        });
        final Button preferencia = (Button) findViewById(R.id.buttonpreferencia);
        preferencia.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
        });
        final Button tipo = (Button) findViewById(R.id.buttontipo);
        tipo.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
        });
    }
```

Figura 3.5Ejemplo de utilización del patrón controlador.

3.3.2 Patrones Estructurales

Los patrones de diseño estructurales solucionan problemas de composición de clases y objetos, se ocupan de cómo se combinan las clases y los objetos para formar estructuras más grandes. (37)

3.3.2.1 Adaptador

Este patrón convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permite que cooperen clases que de otra manera no podrían por tener interfaces incompatibles. Dentro de la plataforma móvil, puede emplearse para generar los elementos de componentes visuales como las listas expandible, que requieren de un adaptador para crear los grupos padres y los hijos, que han de ser mostrados al usuario. (37)

La siguiente figura muestra la utilización del patrón Adaptador en el desarrollo del sistema:

```
public class AdaptadorMostrarLista extends ArrayAdapter<Publicidad> {
    Activity context;
    int pos;
    private ArrayList<Publicidad> data;
    public AdaptadorMostrarLista(Activity context, ArrayList<Publicidad> data) {

    public Publicidad getItem(int position) {
    public View getView(int position, View convertView, ViewGroup parent) {
```

Figura 3. 6 Ejemplo de utilización del patrón controlador.

3.3.2.2 Decorador.

Este patrón añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad. Para el entorno de este trabajo, se utiliza de conjunto con el patrón Adaptador, mientras aquel crea los elementos, este le asigna los atributos correspondientes; por ejemplo cada vez que una lista expandible se expande o se contrae, el decorador indica qué mostrar en cada índice y subíndice. (37)

La siguiente figura muestra la utilización del patrón Adaptador en el desarrollo del sistema:

```
public class AdaptadorMostrarLista extends ArrayAdapter<Publicidad> {
    Activity context;
    int pos;
    private ArrayList<Publicidad> data;
    public AdaptadorMostrarLista(Activity context, ArrayList<Publicidad> data) {

    public Publicidad getItem(int position) {
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        pos = position;
        LayoutInflater inflater = context.getLayoutInflater();
        View item = inflater.inflate(R.layout.list_items, null);
        TextView lblvisto = (TextView) item.findViewById(R.id.visto);
        String visto = String.valueOf(data.get(position).getPreferencia());
        if (visto.equals("1"))
            lblvisto.setText("visto " + visto + " ves.");
        else
            lblvisto.setText("visto " + visto + " veces.");
        TextView lblTitulo = (TextView) item.findViewById(R.id.LblTitulo);
        String titulo = data.get(position).getNombre();
        lblTitulo.setText(titulo);
        TextView lblSubtitulo = (TextView) item.findViewById(R.id.LblSubTitulo);
        String subtitulo = data.get(position).getDescripcion();
        lblSubtitulo.setText(subtitulo);
        ImageView image;
        int tipo = data.get(position).getTipo();
        switch (tipo) {
        case 1:
            image = (ImageView) item.findViewById(R.id.imageView1);
            image.setImageResource(R.drawable.image);
```

Figura 3.7 Ejemplo de utilización del patrón decorador.

3.3.3 Patrones de Comportamiento.

Los patrones de comportamiento tienen que ver con algoritmos y con la asignación de responsabilidades a objetos. Los patrones de comportamiento describen no sólo patrones de clases y objetos, sino también patrones de comunicación entre ellos.

3.3.3.1 Observador

Define una dependencia de forma que cuando un objeto cambie su estado se notifique y se actualicen automáticamente los objetos que dependen de él.

La siguiente figura muestra la utilización del patrón Observador en el desarrollo del sistema:

```
private final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case MESSAGE_STATE_CHANGE:
            switch (msg.arg1) {
            case Receive.STATE_CONNECTED:
                mTitle.setText(R.string.title_connected_to);
                mTitle.append(mConnectedDeviceName);
                conectado = true;
                break;
            case Receive.STATE_CONNECTING:
                mTitle.setText(R.string.title_connecting);
                break;
            case Receive.STATE_LISTEN:
                conectado = false;
                break;
```

Figura 3. 8 Ejemplo de utilización del patrón observador.

3.3.4 Patrones de diseño de interfaces

Para desarrollar aplicaciones destinadas al SO Android es necesario conocer no sólo los patrones que dan solución a problemas frecuentes relacionados con la programación sino que también hay que tener en cuenta la forma de interacción entre la interfaz del software y el usuario. Entre los patrones de diseño de interfaces podemos encontrar la forma de mostrar los datos, cómo ordenarlos, filtros típicos, la entrada de datos y la selección en los distintos spinners o sliders y cómo cambiar entre vistas (listados, mapas, pestañas). (38)

3.3.4.1 Carrusel

Un carrusel se puede utilizar cuando se quiere proporcionar un método atractivo para la selección de objetos, el uso de objetos pictóricos. También es útil cuando se tiene limitado espacio de la pantalla disponible para muchos elementos de navegación. Cuando en pantalla es muy limitado, se considera el uso de elementos de navegación en el menú de opciones. Navegación Carrusel viene en diferentes formas y tamaños, dependiendo del espacio de la pantalla que está disponible y la experiencia de usuario que desee. (39)

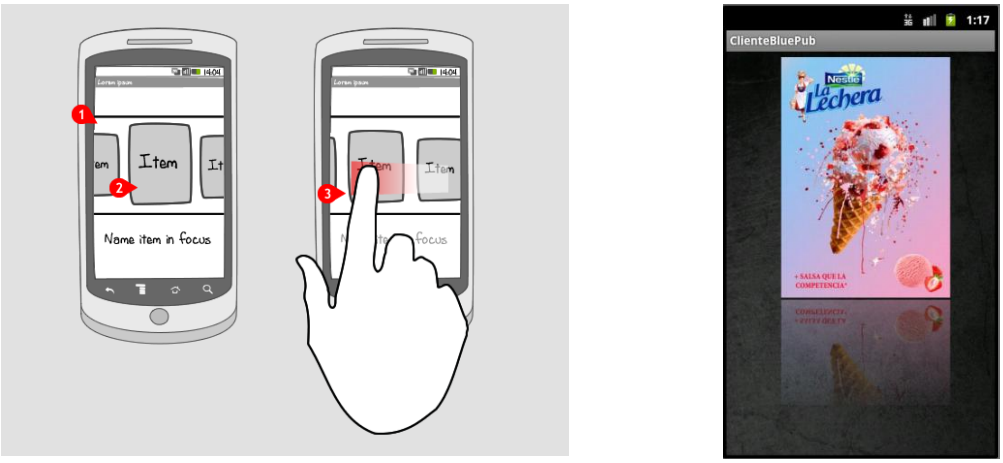


Figura 3.9 Ejemplo de utilización del patrón carrusel.

3.3.4.2 Menú Contextual

El menú contextual es como el menú contextual del botón derecho en los sistemas Windows. Contiene las funciones que el usuario puede encontrar en otros lugares. Cuando se quiere proporcionar al usuario un acceso directo a los comandos utilizados con frecuencia que se pueden realizar en un artículo, usted puede poner los comandos en el menú contextual. Se ejecuta cuando el usuario realiza una presión prolongada en un punto específico en una vista. (40)

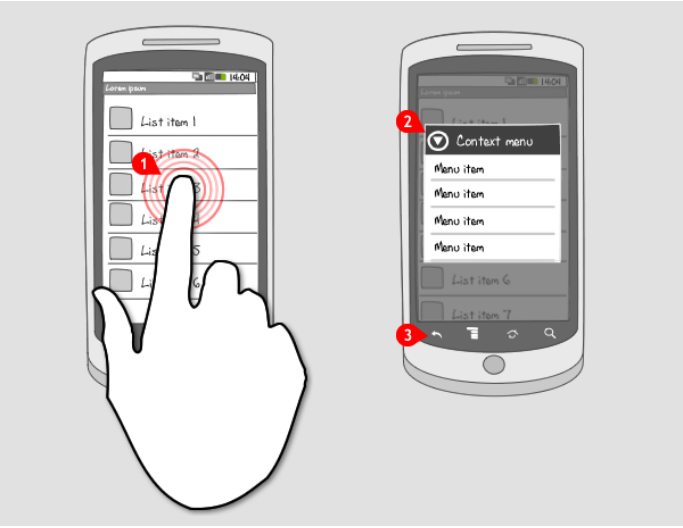


Figura 3.10 Ejemplo de utilización del patrón menú contextual.

3.3.4.3 Lista Expandible

Listas desplegables son útiles cuando el contenido tiene que ser visible en la misma pantalla. Los contenidos se organizan en grupos (a menudo por categoría) en una lista de dos niveles. Grupos individualmente pueden expandirse para mostrar sus hijos. (41)

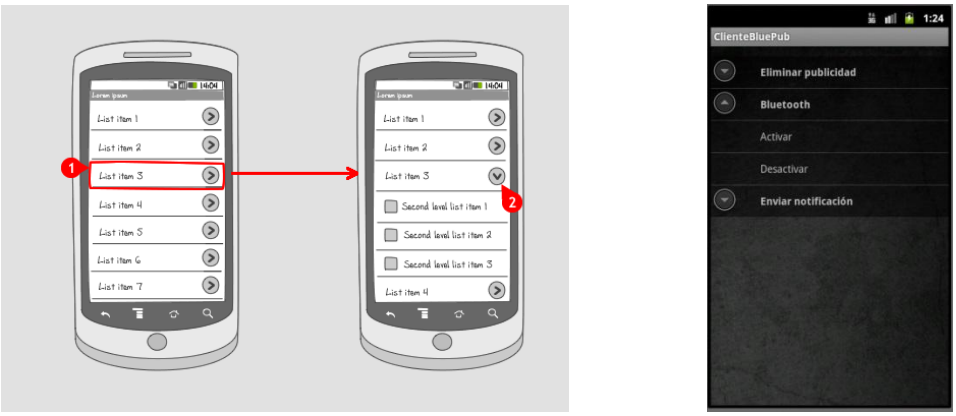


Figura 3.11 Ejemplo de utilización del patrón lista expandible.

3.3.4.4 Lista de Navegación

Una lista de navegación es muy útil, por lo tanto, se aplica muy a menudo. Una vista de la lista es una manera simple y directa para mostrar los elementos de navegación, especialmente cuando el número de elementos a mostrar no es demasiado extenso. Se ponen en un solo nivel, la lista desplazable verticalmente. Sub-encabezados que dividen los elementos en grupos se pueden utilizar para hacer la navegación más fácil. (42)

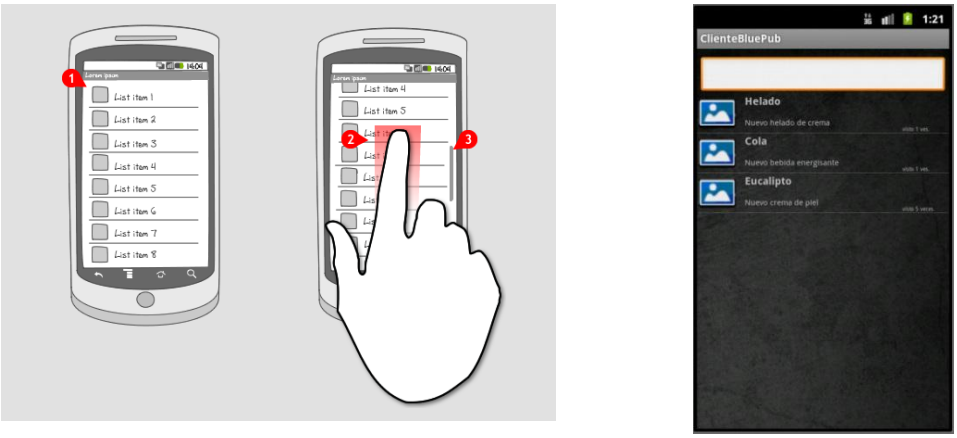


Figura 3.12 Ejemplo de utilización del patrón lista de navegación.

3.4 Diseño

El diseño del sistema es la estrategia de alto nivel para resolver problemas y construir una solución. Éste incluye decisiones acerca de la organización del sistema en subsistemas.

3.4.1 Diagrama de Paquetes

Los paquetes de diseño son usados fundamentalmente como herramienta organizacional del modelo para agrupar elementos, ya sea una colección de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes que estén relacionados de alguna forma.

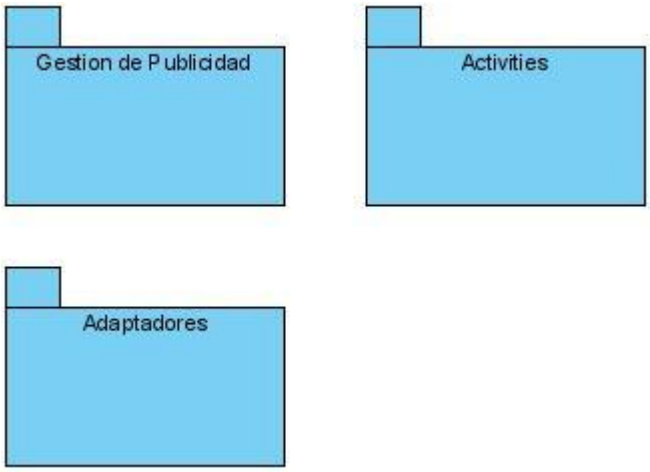


Figura 3.13 Diagrama de paquetes.

3.4.2 Diagrama de Clases del Diseño

Los diagramas de clase del diseño ayudan a tener una mejor comprensión de la implementación del sistema a través de la descripción gráfica de las especificaciones de las clases de software y de las interfaces en una aplicación. A continuación se muestra el Diagrama de Clases del Diseño de la funcionalidad Recibir Publicidad.

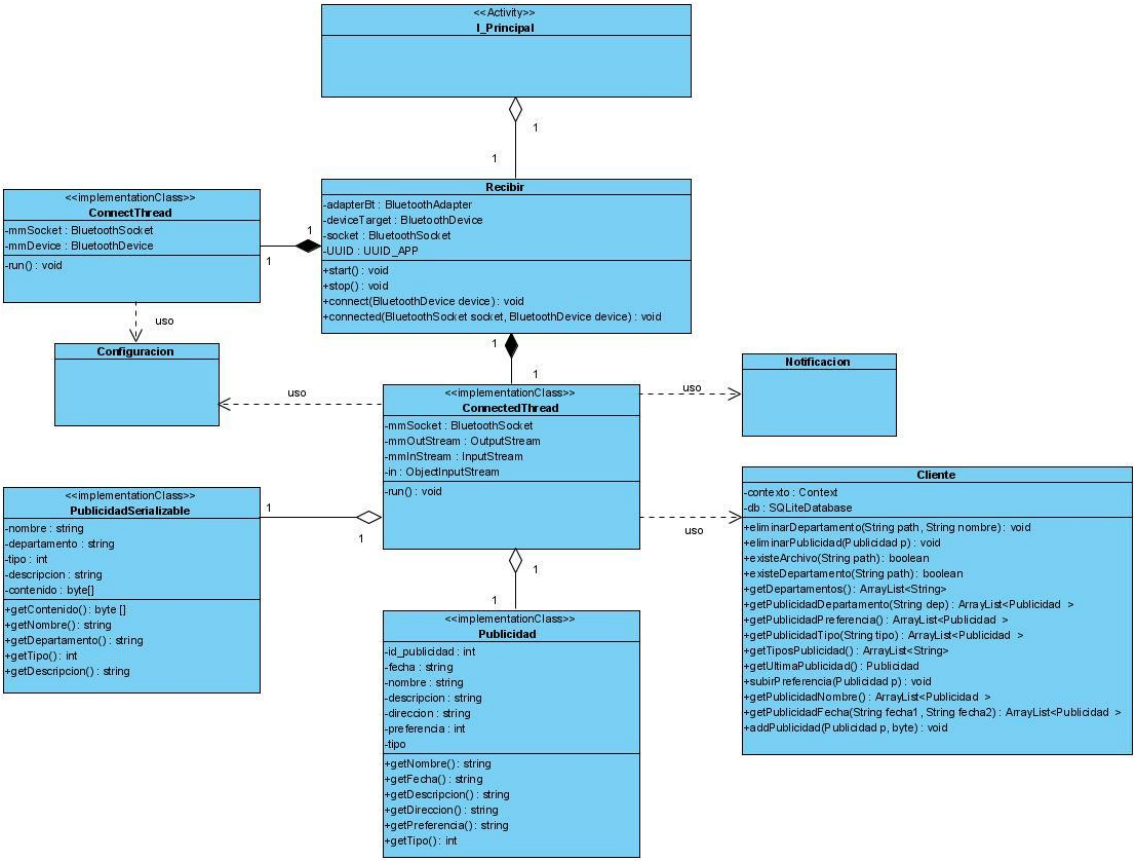


Figura 3.14 Diagrama de clases de la funcionalidad Recibir Publicidad.

3.4.2.1 Descripción de las clases del diseño

En la presente sección se describen las principales clases presentes en el diagrama de clases del diseño de la funcionalidad Recibir Publicidad. Para cada una de ellas se muestra su nombre, una breve descripción de su función y se expone además el objetivo y forma de funcionamiento de todas las operaciones contenidas por la clase, exceptuando aquellas comúnmente conocidas como “get” y “set”.

Clases contenidas en el diagrama de clases del diseño de la funcionalidad Recibir Publicidad.

Nombre: I_Principal	
Tipo de clase: Actividad (Activity) La actividad “I_Principal” se implementa para el inicio de los servicios que brinda el sistema.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**comprobarDevice()**
Descripción:	Comprueba si el dispositivo posee adaptador Bluetooth y el estado del mismo, para en caso que esté desactivado solicitar la activación.
Nombre:	**iniciarRecibo()**
Descripción:	Instancia la clase recibir y comienza el recibo.

Tabla 3.1 Descripción de la clase I_Principal.

Nombre: Recibir	
Tipo de clase: Controladora Está concebida para el control y manejo de la petición de conexión y recibo de la publicidad.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**start()**
Descripción:	Comprueba si existe alguna tarea de petición de conexión o recibo de publicidad, en caso positivo cancela dichas tareas. Modifica el estado de la conexión a

	"escuchar".
Nombre:	**stop()**
Descripción:	Comprueba si existe alguna tarea de petición de conexión o recibo de publicidad, en caso positivo cancela dichas tareas. Modifica el estado de la conexión a "ninguno".

Tabla 3.2 Descripción de la clase Recibir.

Nombre: ConnectThread	
Tipo de clase: Extiende Thread Está concebida para la petición de conexión con el servidor de publicidad.	
Atributo	**Tipo**
Device	BluetoothDevice
Para cada responsabilidad:	
Nombre:	**run()**
Descripción:	Inicia la tarea de petición de conexión con el servidor de publicidad.

Tabla 3.3 Descripción de la clase ConnectThread.

Nombre: ConnectedThread	
Tipo de clase: Extiende Thread Está concebida para la petición de publicidad al servidor de publicidad.	
Atributo	**Tipo**
Socket	BluetoothSocket
Para cada responsabilidad:	
Nombre:	**run()**
Descripción:	Inicia la tarea de petición de publicidad al servidor de publicidad.

Tabla 3.4 Descripción de la clase ConnectedThread.

Nombre: Cliente	
Tipo de clase: Controladora Está concebida para gestionar todas las operaciones referentes a la publicidad.	
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	**eliminarPublicidad(Publicidad p)**
Descripción:	Elimina de la base de datos y de la memoria interna del teléfono la publicidad.
Nombre:	**getDepartamentos()**
Descripción:	Genera un listado con los departamentos de envío de publicidad.
Nombre:	**getDepartamentos()**
Descripción:	Genera un listado con los departamentos de envío de publicidad.

Tabla 3.5 Descripción de la clase ConnectedThread.

3.4.3 Diagramas de interacción. Diagrama de secuencia

Los diagramas de interacción permiten ilustrar paso a paso cómo es que se llevan a cabo los procesos internos del sistema. Se realizan para comprender la lógica establecida para la implementación y pueden ser de dos tipos: secuencia y colaboración. Los diagramas de secuencia representan una descripción del modo en el que cada operación lleva a cabo sus responsabilidades y modifica el estado del sistema. Esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. A continuación se muestra el diagrama de secuencia de un escenario de la funcionalidad Recibir Publicidad.

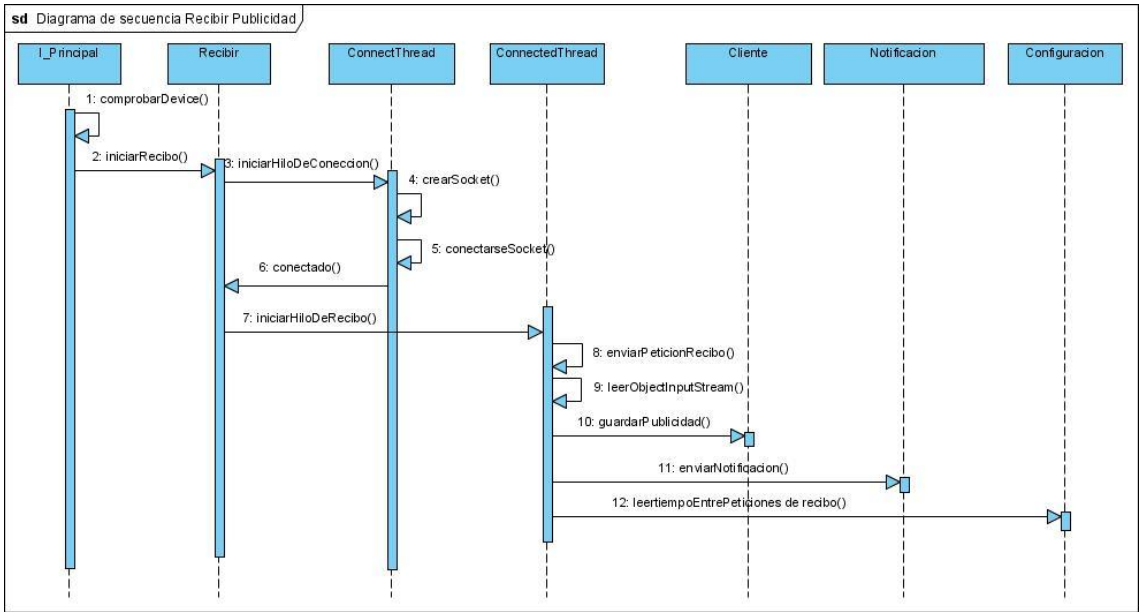


Figura 3.15 Diagrama de secuencia de la funcionalidad Recibir Publicidad.

3.4.3.1Descripción del Diagrama de Secuencia de la funcionalidad Recibir Publicidad.

La secuencia de la funcionalidad Recibir Publicidad comienza cuando se ejecuta el sistema cliente BluePub Sender en el teléfono móvil con SO Android. Se comprueba que el terminal cuenta con un adaptador Bluetooth, en caso positivo se procede a comprobar el estado del dispositivo, cuando el dispositivo se encuentre activado se inicia el hilo de conexión con el servidor. Una vez establecida la conexión con el servidor se inicia el hilo de recibo de publicidad. Se realiza la petición de recibo al servidor de publicidad, una vez obtenido el contenido publicitario se guarda la publicidad, se lanza una notificación para informar al usuario del nuevo anuncio recibido y se espera un tiempo determinado para hacer la próxima petición al servidor.

3.5 Diseño de la base de datos.

El diseño de la base de datos es una de las tareas más importantes en la construcción de un sistema que utilice una base de datos. Se construye con el fin de poder acceder a la información de una manera eficiente y con la menor redundancia posible.

3.5.1 Modelo de Datos

Describe la representación lógica y física de los datos persistentes usados por el sistema.

Modelo de datos del sistema cliente:

Figura 3.16 Modelo de Datos del sistema cliente.

Para una mejor comprensión del modelo de datos y el objetivo del mismo se describe a continuación:

Publicidad: contiene los datos relacionados con la publicidad la cual es almacenada dentro de la memoria del teléfono como un archivo del tipo de la publicidad.

Modelo de datos del sistema servidor:

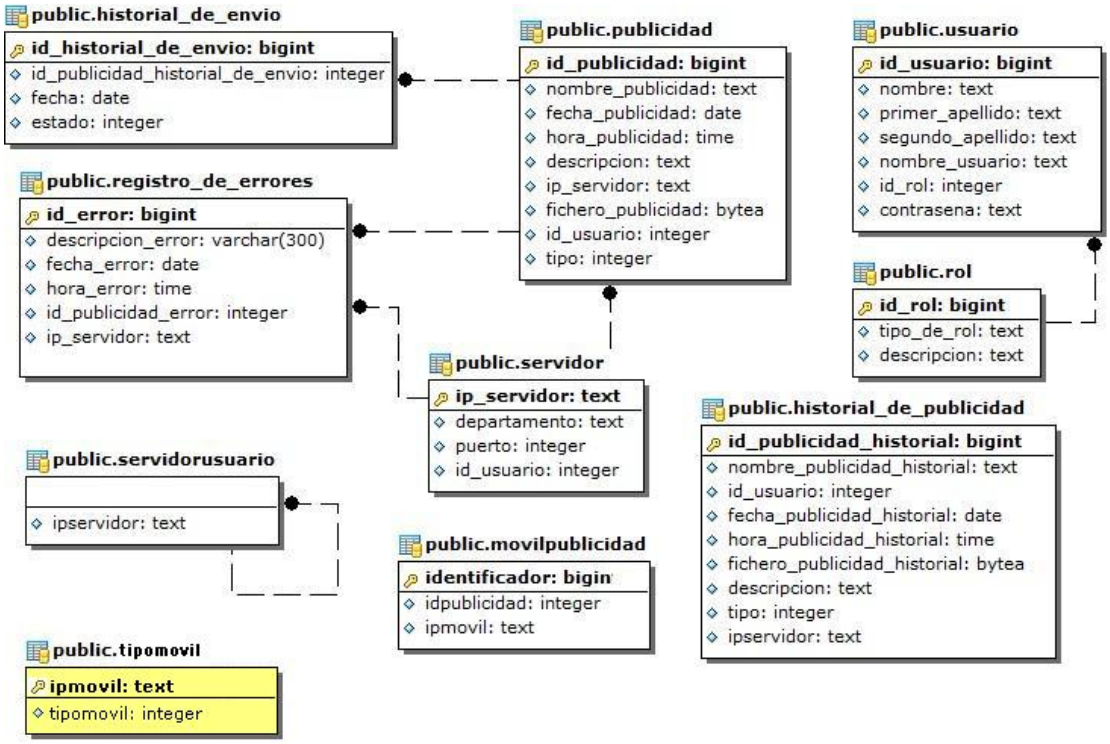


Figura 3.17 Modelo de Datos del sistema servidor.

Para controlar si el terminal que se le procederá a enviar la publicidad es un móvil común o un móvil que posee el sistema cliente BluePub Sender se hizo necesario agregar al modelo de datos del sistema servidor de envío de publicidad la tabla tipomovil (color amarillo). (3 p. 46)

CONCLUSIONES PARCIALES

En el desarrollo del capítulo se mostró el estilo y patrones arquitectónicos presentes en el sistema y los patrones de diseño a usar durante el desarrollo del mismo. Además se describió el modelo de datos, y se elaboraron los diagramas de paquetes, de clases del diseño y de secuencia de los casos de uso más significativos, lo que ha permitido obtener una visión general del funcionamiento del sistema.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

Mediante este capítulo se describen las características de la fase de Implementación en base a las funcionalidades, donde se describe cómo se implementan en términos de componentes los elementos del modelo de diseño. Se modelan los diagramas de componentes y de despliegue, dando una visión de cómo quedará desarrollado el sistema. Además se especifica el tipo de prueba realizada a todas las funcionalidades necesarias.

4.2 Implementación

Durante la implementación del sistema se define la estructuración y organización del código, se implementan los elementos del diseño, se integran los resultados producidos en un sistema ejecutable y se distribuyen físicamente asignando componentes ejecutables a nodos.

4.2.1 Diagrama de despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador o un dispositivo. A continuación se muestra el diagrama de despliegue del sistema:

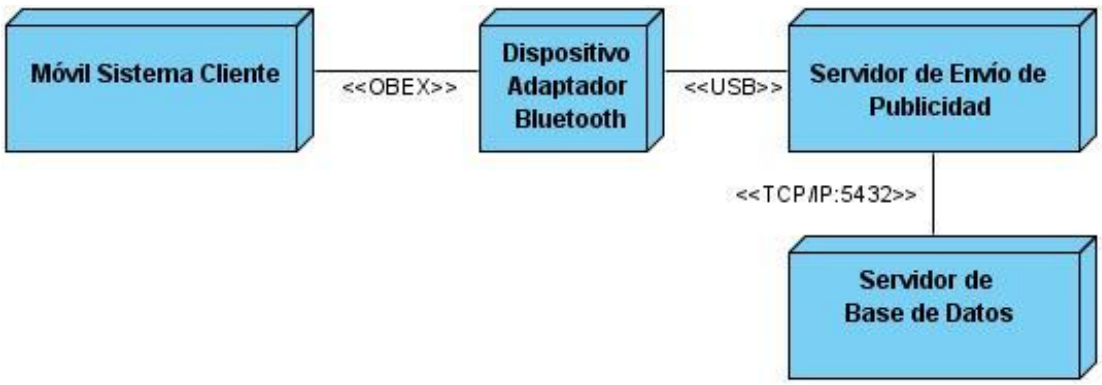


Figura 4.1 Diagrama de despliegue.

- Móvil Sistema Cliente: teléfono móvil que posee la tecnología Bluetooth tiene SO Android v2.3 o superior y cuenta con la aplicación cliente BluePub Sender.

- Servidor de Envío de Publicidad: cuenta con el dispositivo Bluetooth que se encarga de realizar el envío de los ficheros de publicidad hacia los dispositivos móviles que se encuentran dentro del área de cobertura.
- Servidor de Base de Datos: contiene la base de datos donde se encuentran registradas las publicidades, los servidores, los usuarios y los historiales de las publicidades enviadas por el sistema.

4.2.2 Diagrama de componentes

Los diagramas de componentes describen los elementos lógicos del sistema, además modelan los aspectos físicos, es decir, un conjunto de elementos físicos y sus relaciones. Los componentes son la representación de todos los tipos de elementos de software que intervienen en la creación de software. Los componentes exponen los subsistemas de implementación y sus dependencias de importación y los subsistemas de implementación organizados en capas

A continuación se expone el diagrama de subsistemas de implementación del sistema BluePub Sender, el diagrama de componentes de la Publicidad y algunos de los componentes presentes en las librerías del sistema.

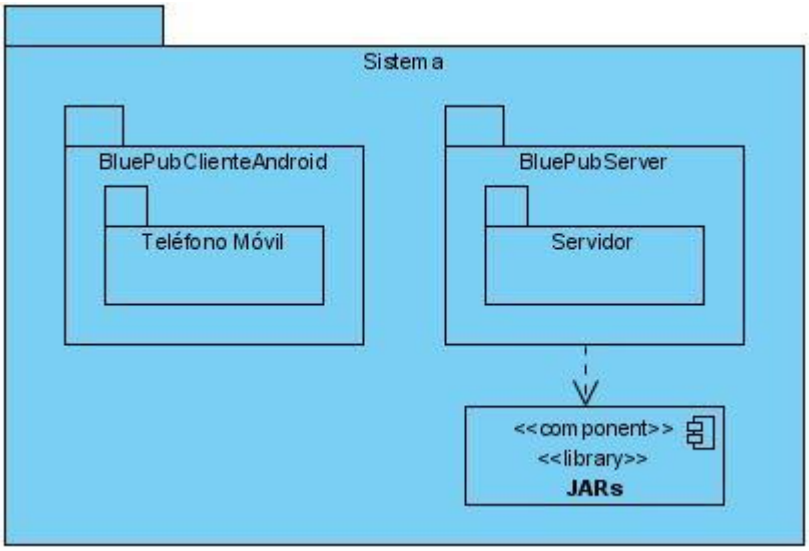


Figura 4.2 Diagrama de Subsistemas de Implementación.

Descripción de los elementos:

Sistema: contiene los artefactos del sistema.

- BluePubClienteAndroid: contiene los artefactos relacionados con las funcionalidades del cliente para teléfonos móviles.
- BluePubServer: contiene los artefactos relacionados con las funcionalidades del servidor.

- JARs: contiene las librerías usadas, tanto las de Spring como las creadas durante el desarrollo del sistema.

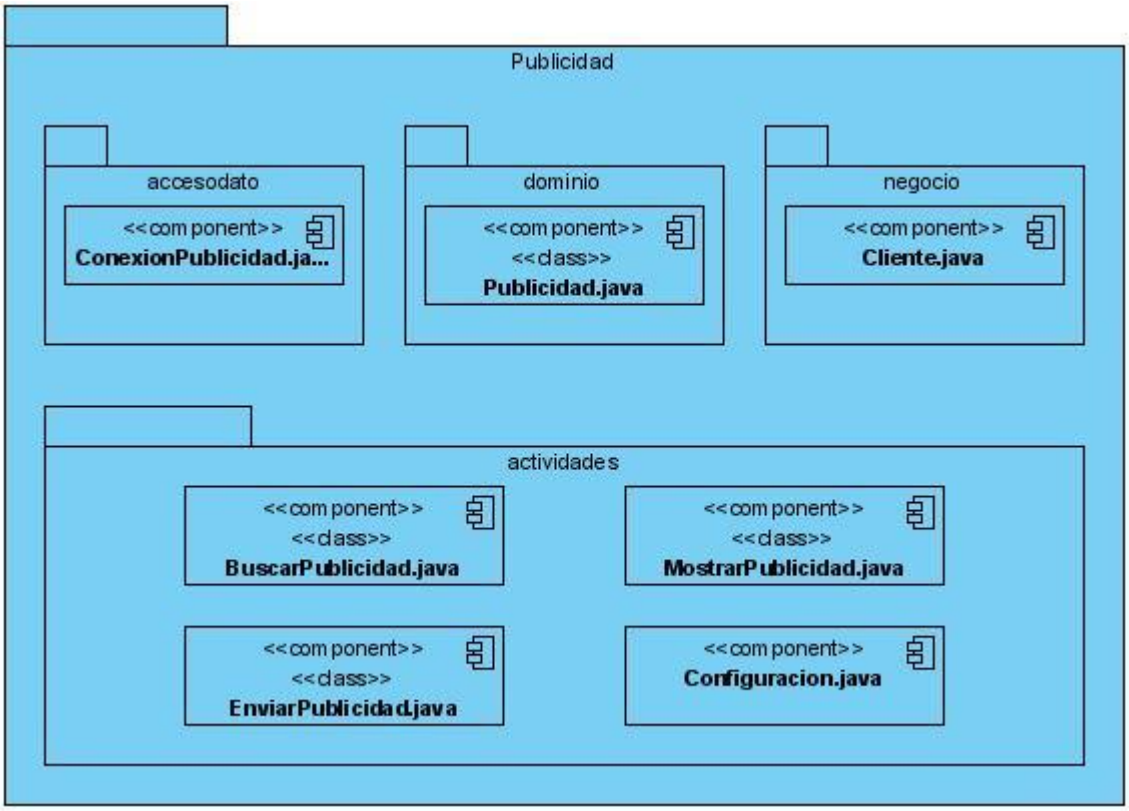


Figura 4.3 Diagrama de Componentes.

Descripción de los elementos:

- accesodato: contiene las clases encargadas de acceder a los datos de la base de datos SQLite.
- dominio: contiene las clases que representan una publicidad.
- negocio: contiene las clases que permiten relacionar las clases controladoras con las clases de acceso a datos.
- actividades: contiene las clases encargas de atender las peticiones de los usuarios y manejan las vistas.

4.3 Pruebas de software

Las pruebas de software son los procesos que permiten verificar la calidad de un producto de software, representando una inspección final de las especificaciones del diseño y la implementación, donde se identificarán posibles fallos. El objetivo de las pruebas es presentar información sobre la calidad del producto a las personas responsables de éste.

4.3.1 Pruebas Unitarias

Inician con las pruebas de cada módulo. Es una forma de probar el adecuado funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de estos funcione correctamente por separado. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el fragmento de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas, fomentan el cambio, simplifica la integración, documenta el código, separa la interfaz del código y hace que los errores estén más acotados y sean fáciles de localizar. Se hiso uso de este nivel de prueba en las clases del negocio para asegurar que cada una de las funcionalidades se desempeñe correctamente por separada. (43)

Clases Probadas	Clases de Prueba
I_Principal	TestActPtincipal
I_Buscar	TestActBuscar
I_MostrarPublicidad	TestActMostrarPublicidad
I_Configuracion	TestActConfiguracion

Tabla 4.1 Pruebas Unitarias.

4.3.2Pruebas de Funcionalidad

Entre las pruebas existentes para comprobar el cumplimiento de los requerimientos, se encuentra la prueba por funcionalidad. Esta pretende evaluar el cumplimiento de los requisitos funcionales, incluye la navegación, entrada de los datos, procesamiento y la obtención de resultados. Se ejecuta en cada funcionalidad corroborando una implementación adecuada. Se realizó este tipo de pruebas en dispositivos virtuales y reales. El principal obstáculo en el dispositivo virtual es que no es capaz de establecer la conexión Bluetooth puesto que la herramienta SDK no provee soporte para emular esta tecnología. A continuación se muestra un listado de los dispositivos en los que se realizó la prueba. (44)

Tipo de dispositivo	Marca	Sistema Operativo
Real	Samsung Galaxy Y	Android v 2.3.6
Emulado	-	Android v 2.3.3
Emulado	Motorola Mobility MT870	Android v 2.3.3
Emulado	HTC OpenSense	Android v 4.03

Tabla 4.2 Resultado de las Pruebas de Aceptación

4.3.3No conformidades detectadas en el sistema

A continuación se muestran la cantidad de no conformidades que han sido detectadas en el sistema, mostrando cuántas de estas proceden, cuántas no proceden y cuántas fueron resueltas; todas fueron no significativas.

Sistema	Total de no conformidades	# de no conformidades que proceden	# de no conformidades que no proceden	# de conformidades resueltas
BluePub Sender V2.0	7	5	1	4

Tabla 4.3 No conformidades detectadas en el sistema

CONCLUSIONES PARCIALES

En este capítulo se han desarrollado los aspectos que muestran la implementación del sistema. Se realizó el diagrama de subsistemas de implementación así como los diagramas de componentes. Se estableció el diagrama de despliegue, el cual muestra cómo estará distribuido el sistema en un plano físico. Se realizaron pruebas para comprobar el correcto funcionamiento del sistema para validar que el mismo cumple con los requisitos establecidos.

CONCLUSIONES

En el presente trabajo se expuso la necesidad de desarrollar un sistema cliente para el sistema de envío de publicidad BluePub Sender orientado a dispositivos con SO Android v2.3 que posean tecnología Bluetooth que permita la gestión de los contenidos publicitarios. Se describieron todos los procesos identificados para el envío y recepción de publicidad haciendo uso de la tecnología Bluetooth. Se seleccionaron las herramientas, tecnologías y metodologías de desarrollo de software que se utilizaron. Se definió la arquitectura a utilizar así como los patrones arquitectónicos para el diseño e implementación del sistema.

Con la elaboración del sistema cliente de envío de publicidad BluePub Sender se le da solución a los problemas existentes en el proceso de realización de las campañas publicitarias. Los archivos publicitarios son recibidos desde el sistema servidor BluePub Sender por la aplicación cliente y pueden ser gestionados de forma óptima, permitiendo la visualización directa de la publicidad, además permite la organización y borrado de los mismos.

Por todo lo expuesto anteriormente se concluye que se cumplieron los objetivos propuestos cumpliendo cada una de las tareas propuestas para desarrollar el sistema cliente para el sistema de envío de publicidad BluePub Sender.

RECOMENDACIONES

Después de culminado el desarrollo del sistema se detectaron algunas necesidades que pueden incluirse como posibles mejoras al mismo. A continuación se detallan las recomendaciones realizadas al sistema:

1. Extender el sistema cliente a dispositivos con otros Sistemas Operativos para ampliar la cobertura de la campaña publicitaria.
2. Incorporar otros servicios al sistema tales como el envío de mensajes SMS.
3. Extender la compatibilidad del sistema a dispositivos móviles que no cuentan con el servicio OBEX de transferencia de archivos.
4. Incluir en el sistema cliente un mecanismo para que funcione como servidor y así crear una red Scatternet en caso de limitarse la capacidad de conexiones permitidas.
5. Incluir una alternativa en el servidor BluePub Sender para enviar la aplicación cliente mediante Bluetooth a los móviles con Android y permitir su instalación inmediata.

REFERENCIAS BIBLIOGRÁFICAS

1. **Gordon, Kim T.** Cómo hacer publicidad efectiva | SoyEntrepreneur. [En línea] [Citado el: 1 de 06 de 2013.] http://www.soyentrepreneur.com/como-hacer-publicidad-efectiva.html.
2. Android. [En línea] [Citado el: 1 de 06 de 2013.] http://www.android.com/.
3. **Ortega Aponte, Graviel y Guerra Reyes, Yailemi.** *BluePub Sender: Sistema de Envío de Publicidad vía Bluetooth.* Habana : s.n., 2012.
4. Marketing de proximidad por bluetooth. [En línea] [Citado el: 23 de 11 de 2012.] http://www.bluems.es/productos-proximity-marketing.
5. Bluetooth Marketing Software. [En línea] [Citado el: 23 de 11 de 2012.] http://www.zonablu.net/bluetooth-proximity-marketing-software.aspx.
6. Bluemessenger. [En línea] [Citado el: 23 de 11 de 2012.] http://www.bluemessenger.es/.
7. Marketing software. [En línea] [Citado el: 12 de 20 de 2012.] http://areabluetooth.com/v3/AreaBluetooth-Light?lang=esp.
8. EntuMovil. [En línea] [Citado el: 20 de 12 de 2012.] http://www.entumovil.cu/textcontent/read/bluespace.
9. **Ortega Aponte, Graviel y Guerra Reyes, Yailemi.** *BluePub Sender: Sistema de Envío de Publicidad vía Bluetooth.* 2012. pág. 22.
10. —. *BluePub Sender: Sistema de Envío de Publicidad vía Bluetooth.* 2012. pág. 24.
11. What is Bluetooth Technology. [En línea] [Citado el: 16 de 12 de 2012.] http://www.bluetooth.com/Pages/what-is-bluetooth-technology.aspx.
12. **Borches, P.** Java 2 Micro Edition: Soporte Bluetooth. [En línea] [Citado el: 09 de 06 de 2013.] http://www.it.uc3m.es/celeste/docencia/j2me/tutoriales/bluetooth/EstudioTecnologico1_0.pdf.
13. Lenguajes de programación. [En línea] [Citado el: 2 de 06 de 2013.] http://guimi.net/descargas/Monograficos/G-Lenguajes_de_programacion.pdf.
14. Características del lenguaje Java. [En línea] [Citado el: 12 de 12 de 2012.] http://www.iec.csic.es/criptonomicon/java/quesjava.html.
15. Lenguaje de Modelamiento Unificado. [En línea] [Citado el: 12 de 12 de 2012.] http://www.slideshare.net/ecastrojimenez/uml-lenguaje-de-modelamiento-unificado-presentation.
16. **Pereira, M.** UML. [En línea] [Citado el: 12 de 12 de 2012.] http://es.scribd.com/doc/2080534/UML.
17. Comparativa de Frameworks. [En línea] [Citado el: 1 de 06 de 2013.] http://seamcity.madeinxpain.com/archives/comparativa-spring.
18. JDBC. [En línea] [Citado el: 01 de 06 de 2013.] http://www.vc.ehu.es/jiwotvim/ISOFT2010-2011/Teoria/BloqueIV/JDBC.pdf.
19. **Amaro, Calderón, y otros, y otros.** *Sockets y su programación en Java.* Peru : Universidad Nacional de Trujillo Facultad de Ciencias Físicas y Matemáticas, 2007.
20. BlueCove. [En línea] [Citado el: 1 de 05 de 2013.] http://bluecove.org/.
21. Estadísticas Android. [En línea] [Citado el: 14 de 03 de 2013.] http://www.android.es/estadisticas-android-022013-jelly-bean-va-comiendo-terreno.html.
22. Android 2.3 (Gingerbread). [En línea] [Citado el: 04 de 12 de 2012.] http://androidweblog.org/android-2-3-gingerbread..
23. **Pressman, Roger S.** *Ingeniería de Software.* s.l. : Hingher Education, 2010.

24. **Gastón, Mousques.** *Metodología FDD.* 2003.
25. Eclipse - The Eclipse Foundation open source community website. [En línea] [Citado el: 21 de 11 de 2012.] http://www.eclipse.org/.
26. Apache Tomcat. [En línea] [Citado el: 21 de 11 de 2012.] http://tomcat.apache.org/.
27. PostgreSQL. [En línea] [Citado el: 21 de 11 de 2012.] http://www.postgresql.org/.
28. SQLite. [En línea] 12 de 06 de 2013. http://gdroid.com.mx/blog/2013/03/05/que-es-sqlite-base-de-datos-android/.
29. UML, BPMN and Enterprise Architecture Tool for Software Development. [En línea] 14 de 12 de 2012. http://www.visual-paradigm.com/..
30. Android SDK | Android Developers. [En línea] [Citado el: 25 de 1 de 2013.] http://developer.android.com/intl/es/sdk/index.html.
31. Publicidad con bluetooth para celulares. [En línea] [Citado el: 15 de 2 de 2013.] http://www.publicidadweb.ws/herramientas/publicidad-con-bluetooth.html.
32. Android Interaction Design Patterns. [En línea] [Citado el: 22 de 1 de 2013.] http://www.androidpatterns.com/.
33. Oposiciones TIC. [En línea] [Citado el: 23 de 05 de 2013.] http://oposicionestic.blogspot.com/2011/06/arquitectura-cliente-servidor.html.
34. Cliente-Servidor. [En línea] [Citado el: 26 de 05 de 2013.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf.
35. [En línea] [Citado el: 26 de 05 de 2013.] http://www.negomobile.es/en/node/221.
36. **Visconti, Marcello y Astudillo, Hernán.** Fundamentos de Ingeniería de Software. [En línea] [Citado el: 26 de 05 de 2013.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.
37. **Gamma, Erich, y otros, y otros.** *Patrones de Diseño Elementos de software orientado a objetos reutilizables.* s.l. : Addison Wesley, 2005.
38. Android Interaction Design Patterns. [En línea] 12 de 12 de 2012. http://www.androidpatterns.com/.
39. Carousel | Android Interaction Design Patterns. [En línea] 12 de 12 de 2012. http://www.androidpatterns.com/uap_pattern/carousel.
40. Context Menu | Android Interaction Design Patterns . [En línea] 12 de 12 de 2012. http://www.androidpatterns.com/uap_pattern/context-menu.
41. Expandable list | Android Interaction Design Patterns. [En línea] 12 de 12 de 2012. http://www.androidpatterns.com/uap_pattern/list-expandable-list.
42. List navigation | Android Interaction Design Patterns. [En línea] 12 de 12 de 2012. http://www.androidpatterns.com/uap_pattern/list-navigation.
43. Android testing. [En línea] 01 de 06 de 2013. http://jmrogado.wordpress.com/2011/03/24/android-testing-con-intellij/.
44. Test de aceptación. [En línea] 03 de 06 de 2013. http://www.4rsoluciones.com/test-de-aceptacion-el-ultimo-paso-para-el-aseguramiento-de-calidad-en-software/ .
45. **Pressman, Roger S.** *Software Engineering.* 2013.
46. SpringSource.org. [En línea] [Citado el: 01 de 06 de 2013.] http://www.springsource.org/.

GLOSARIO DE TÉRMINOS

Activity: Actividad. Las actividades representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana en cualquier otro lenguaje visual. Por lo que se utiliza para manejar y controlar eventos, y para la entrada/salida de datos mediante los layouts.

Arquitectura: Organización de los diversos elementos constitutivos de un sistema informático.

Bluetooth: Es una especificación para Redes Inalámbricas que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia.

CPU: Es el componente principal del ordenador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.

Frameworks: Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.

Marketing: Es una filosofía que sostiene que la clave para alcanzar los objetivos de la organización reside en identificar las necesidades y deseos del mercado objetivo y adaptarse para ofrecer las satisfacciones deseadas por el mercado de forma más eficiente que la competencia.

RAM: Memoria de acceso aleatorio. Se utiliza como memoria de trabajo para el sistema operativo, los programas y la mayoría del software.

ROM: Memoria de solo lectura. Es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite sólo la lectura de la información y no su escritura, independientemente de la presencia o no de una fuente de energía.

SDK: Kit de desarrollo de software. Es un conjunto de herramientas de desarrollo de software que permiten implementar aplicaciones para un sistema concreto.

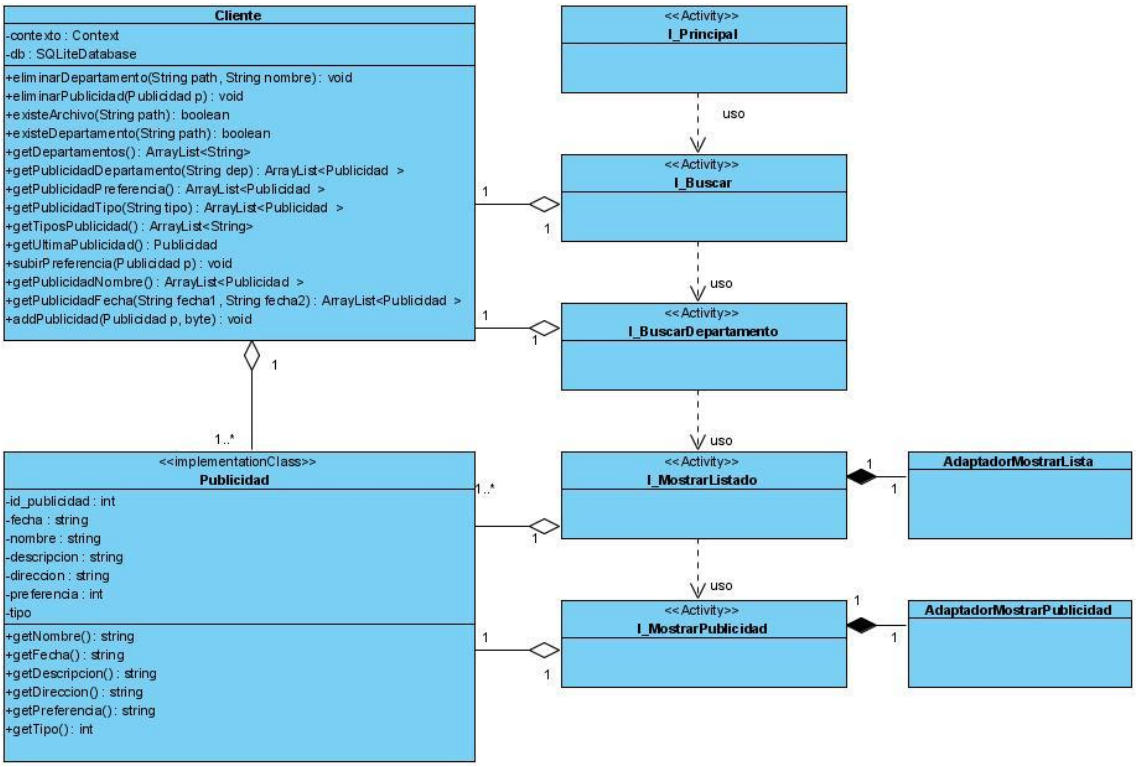
Servlets: Son objetos que corren dentro y fuera del contexto de un contenedor de servlets y extienden su funcionalidad. El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Sistema Operativo: Es un programa o conjunto de programas que en un sistema informático gestiona los recursos de hardware y provee

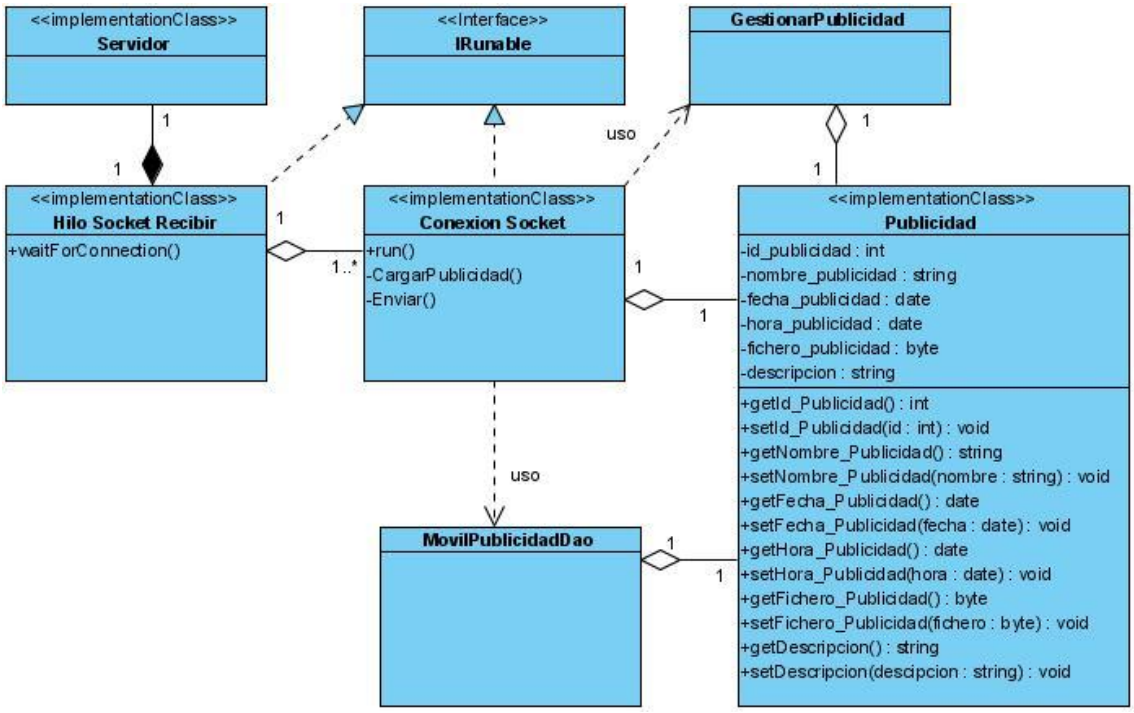
servicios a los programas de aplicación.

Thread: Es una unidad básica de ejecución de una aplicación que se ejecuta de forma concurrente en un contexto determinado.

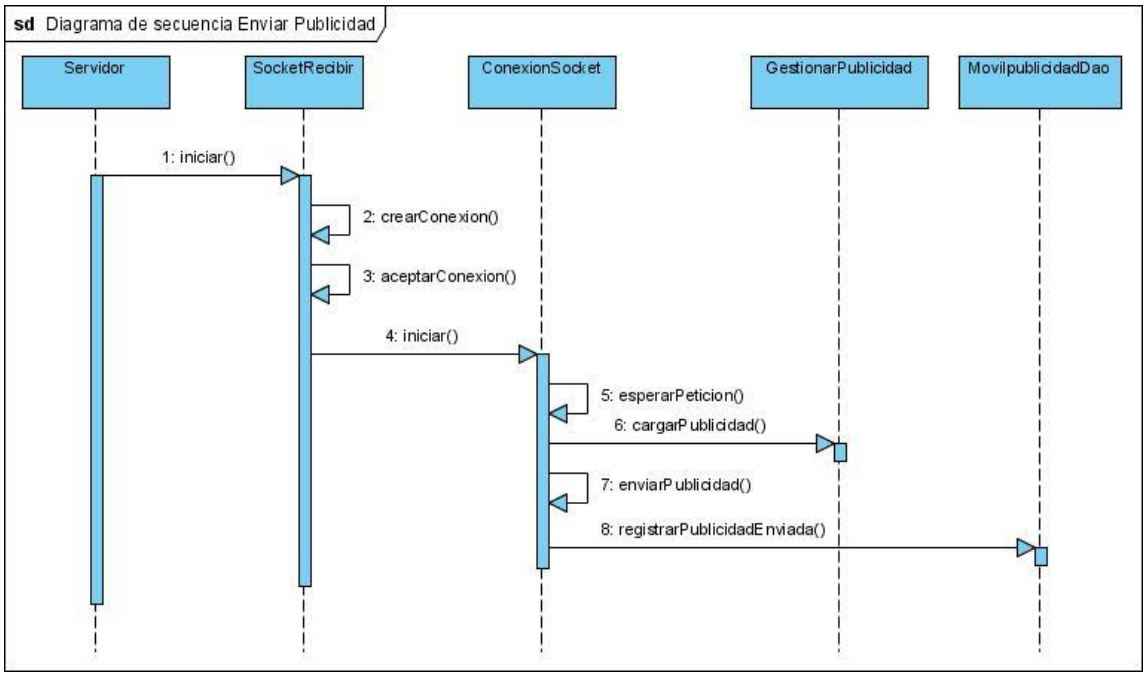
ANEXOS



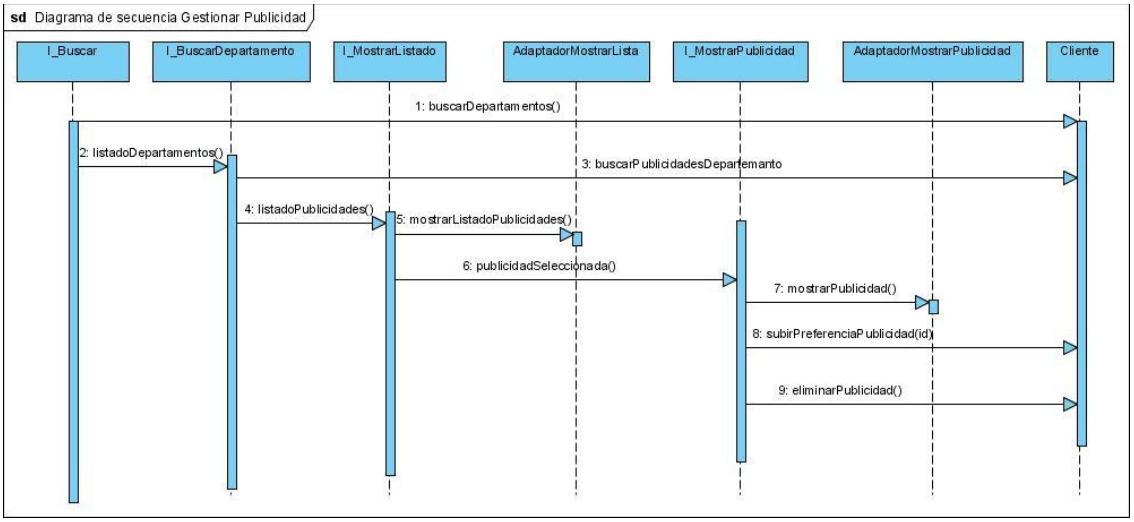
Anexo 1 Diagrama de clases de la funcionalidad Gestionar Publicidad



Anexo 2 Diagrama de clases de la funcionalidad Enviar Publicidad



Anexo 3 Diagrama de secuencia de la funcionalidad Enviar Publicidad



Anexo 4 Diagrama de secuencia de la funcionalidad Gestionar Publicidad