PROTOTIPO DE COMPONENTE SOFTWARE PARA EL DESARROLLO DE APLICACIONES WIRELESS INTERNET EN DISPOSITIVOS MÓVILES CELULARES UTILIZANDO J2ME

JENNY PAOLA MONTILLO GÉLVEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

PROTOTIPO DE COMPONENTE SOFTWARE PARA EL DESARROLLO DE APLICACIONES WIRELESS INTERNET EN DISPOSITIVOS MÓVILES CELULARES UTILIZANDO J2ME

JENNY PAOLA MONTILLO GÉLVEZ

Trabajo de grado para optar al título de
INGENIERA DE SISTEMAS

Director
FERNANDO ANTONIO MORALES ROJAS
Ingeniero de Sistemas, MSc.

Codirector
EDWARD JOSÉ BERTRÁN LOZANO
Ingeniero de Sistemas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

DEDICATORIA

Este logro lo dedico a Dios
porque Él es el motor de mi vida
y porque a Él le debo todo lo que soy
y todo lo que tengo.
Con todo mi amor a mis padres
Ariel Montillo y Jacinta Gélvez Sierra
por su paciencia y apoyo incondicional
y a mi hermano Héctor Mauricio.

Jenny Paola Montillo Gélvez

AGRADECIMIENTOS

Al finalizar esta etapa de mi vida quiero y debo agradecer a todas aquellas personas que me ofrecieron su apoyo desinteresadamente, para que yo pudiera culminar satisfactoriamente este ciclo de formación.

Quiero dar gracias a Dios, porque es Él quien me guía con su luz, me acompaña y me da la fortaleza necesaria para seguir siempre adelante. A mis padres por su gran paciencia y apoyo incondicional.

Al profesor Fernando Antonio Rojas Morales por confiar en mí y por su gran paciencia, al Ing. Edward José Beltrán Lozano por su confianza y apoyo; al Ing. Juan Carlos Escobar Ramírez por su gran colaboración.

Al equipo de desarrollo CloudEISI de la escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander.

Un agradecimiento muy especial a nuestro capellán, el curita Freddy Ramírez por brindarme su colaboración, amistad, apoyo y cariño. A las familias González Gélvez, Parada Gelves, Gélvez Galvis, Anaya Molina y Rojas Sierra por su apoyo y colaboración cuando lo necesité.

A mis amigos Luis Carlos Roa Cantor, Mariutsi Alexandra Osorio Sanabria, Nini Yohana Parada Basto, Rodrigo Rodríguez Faerito, Ferney Farid Fuentes Morelo, Adriana Rocío Pérez Rojas, Margarita Leal Joya, Carlos Alberto Caicedo Simanca, Tuti McCullah y su esposo Matthew, Henry David Brijaldo. A mis amigos de Pastoral Universitaria y demás compañeros con los que compartí este proceso de formación y me brindaron su sincera amistad. A Mario Josué y Astrid Mireya Gélvez Martínez y José Andrés Dávila Martínez por su apoyo desde la distancia.

Y a todas las personas que se preocuparon por mí y de algún modo aportaron su granito de arena para que yo pudiera concluir esta etapa de mi vida.

A todos ellos muchas gracias desde el fondo de mi corazón.

CONTENIDO

pág.

LISTA DE FIGURAS

pág.

LISTA DE CUADROS

pág.

LISTA DE ANEXOS

pág.

GLOSARIO

API: abreviatura inglesa para Application Programming Interface, que en español significa Interfaz de Programación de Aplicaciones. Una API es el conjunto de funciones y métodos que ofrece cierta librería para ser utilizada por otro software como una capa de abstracción, representa una interfaz de comunicación entre componentes software. Uno de los principales propósitos de un API consiste en proporcionar un conjunto de funciones de uso general, de esta forma, los programadores se benefician de las ventajas del API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio.

CGI: abreviatura inglesa para Common Gateway Interface, que en español significa Interfaz de Entrada Común, es una tecnología que permite a un cliente (explorador Web) solicitar datos de un programa ejecutado en un servidor Web. Especifica un estándar para transmitir datos entre el cliente y el programa. Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.

CLDC: abreviatura inglesa para Connected Limited Device Configuration, que en español significa Configuración para Dispositivos con Conexión Limitada, define el conjunto base de APIs y una máquina virtual para dispositivos de recursos limitados como teléfonos móviles o PDAs. Cuando es acoplado con un perfil como el MIDP, provee una plataforma Java sólida para el desarrollo de aplicaciones para correr sobre dispositivos de memoria limitada, poder de procesamiento y capacidades gráficas limitadas. Esta configuración está diseñada para dispositivos con conexiones de red intermitentes, alimentación limitada a menudo basada en batería.

DATAGRAMA: es la estructura interna de un paquete de datos ó fragmento de paquete que es enviado con la suficiente información como para que la red pueda simplemente encaminar el fragmento hacia el equipo terminal de datos receptor (ETD), de manera independiente a los fragmentos restantes. Esto puede provocar una recomposición desordenada o incompleta del paquete en el ETD destino.

JCP: abreviatura inglesa para Java Community Process, que en español significa, Proceso de la Comunidad Java que fue establecido en 1998, es un proceso formalizado el cual permite a las partes interesadas involucrarse en la definición de futuras versiones y características de la plataforma Java.

JSR: abreviatura inglesa para Java Specification Request, que en español significa Petición de Especificación Java y es el proceso que sigue después del JCP. Estos JSR son documentos formales que describen las especificaciones y tecnologías propuestas para que sean añadidas a la plataforma Java y de esta

manera extenderla. Actualmente existen más de 300 JSRs para Java y más de 40 peticiones de especificación relacionadas de algún modo con JME.

KVM: máquina virtual de Java compacta y portable. Toma la K de kilobyte, haciendo referencia al poco tamaño que ocupa la plataforma, un mínimo de 70 kilobytes.

LATENCIA: es el tiempo que toma a una solicitud alcanzar el servidor objetivo.

MIDLET: aplicación Java realizada usando la especificación de MIDP, es decir, un midlet puede usar la funcionalidad aportada por MIDP y por CLDC. Siempre estará compuesto al menos, por una clase principal que hereda directamente de la clase javax.microedition.midlet.MIDlet, contenida en el API MIDP estándar.

MIDP: abreviatura inglesa para Mobile Information Device Profile que en español significa Perfil para Dispositivos Móviles de Información, este perfil se combina con la CLDC para proporcionar un entorno de ejecución para dispositivos móviles. Los usuarios pueden seleccionar las aplicaciones MIDP situadas en el servidor Web. El dispositivo comprueba si puede ejecutarla y la descarga, verifica y compila a bytecode para poder ejecutarla. Las aplicaciones instaladas se pueden ejecutar, actualizar y borrar de forma sencilla. Las aplicaciones MIDP permiten tener aplicaciones intuitivas y gráficas. La IGU (Interfaz Gráfica de Usuario) se ha optimizado para las pequeñas pantallas. Se pueden instalar y ejecutar en local, trabajar en red o de forma desconectada y puede almacenar y gestionar de forma segura datos en local.

SOCKET: designa un concepto abstracto por el cual dos programas (posiblemente situados en computadores distintos) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada. Un socket queda definido por una dirección IP, un protocolo y un número de puerto.

RESUMEN

TITULO:
PROTOTIPO DE COMPONENTE SOFTWARE PARA EL DESARROLLO DE APLICACIONES WIRELESS INTERNET EN DISPOSITIVOS MÓVILES CELULARES UTILIZANDO J2ME[*].

AUTOR:
JENNY PAOLA MONTILLO GELVEZ[**]

PALABRAS CLAVE:
Prototipo, Aplicaciones Wireless Internet, Dispositivos Móviles Celulares, Software Libre, HTTP, HTTPS, J2ME, API, CLDC, MIDP, Sockets, TCP, UDP, NetBeans.

DESCRIPCIÓN:

Las ventajas más grandes de los dispositivos inalámbricos son la conectividad, movilidad de datos, acceso a información desde cualquier lugar y en cualquier momento. La funcionalidad de estos dispositivos ha cambiado significativamente en los últimos años, gracias al incremento en el área de cobertura de las redes, a un mayor ancho de banda y a una mejorada tecnología inalámbrica. Ahora los teléfonos celulares no se usan sólo con el objetivo de establecer una conversación, cada vez más se han convertido en "dispositivos de información móvil" que permiten acceder a información empresarial y personal de manera oportuna. Sin embargo todavía se encuentran algunas áreas, –como la comunicación con servidores Web– que no han sido aprovechadas suficientemente, ya sea por falta de desarrollo por parte de los programadores ó por falta de recursos económicos.

Este prototipo procura aprovechar al máximo los beneficios de Java para ofrecer una herramienta de trabajo a los desarrolladores de software, de manera que les permita crear aplicaciones Wireless Internet de una forma más rápida y sencilla. Con ella se facilita el uso de los protocolos HTTP y HTTPS, además de sockets y datagramas, todo esto con las configuraciones CLDC 1.0, CLDC 1.1 y los perfiles MIDP 1.0 y MIDP 2.0. Esta herramienta pretende facilitar el desarrollo e implementación de este tipo de aplicaciones, ya que es de fácil adquisición (software libre), de esta manera contribuye a impulsar su creación; lo cual implica un incremento en la adquisición de las aplicaciones por parte de las personas interesadas.

Tanto el prototipo como el aplicativo software para las pruebas fueron creados con el lenguaje de programación Java, utilizando el IDE NetBeans; el modelo del proceso del software que se usó para el desarrollo de esta herramienta fue el prototipado evolutivo, apoyado en el Lenguaje Unificado de Modelado (UML).

[*] Trabajo de Grado en la Modalidad de Investigación

[**] Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería de Sistemas e Informática, Director: Ing. de Sistemas, MSc. Fernando Antonio Rojas Morales, Codirector: Ing. de Sistemas Edward José Beltrán Lozano.

SUMMARY

TITLE:
PROTOTYPE OF SOFTWARE COMPONENT FOR THE DEVELOPMENT OF WIRELESS INTERNET APPLICATIONS ON CELLULAR MOBILE DEVICES USING J2ME[*].

AUTHOR:
JENNY PAOLA MONTILLO GÉLVEZ[**]

KEY WORDS:
Prototype, Wireless Internet Applications, Mobile Cellular Divices, Free Software, J2ME, HTTP, HTTPS, J2ME, API, CLDC, MIDP, Sockets, TCP, UDP, NetBeans.

DESCRIPTION:

Biggest advantages of wireless devices are connectivity, data mobility, access to information from any place and any moment. The functionality of these devices has changed significantly in last years, thanks to the increase in area of coverage of the networks, to a major bandwidth and improved wireless technology. Now cellular phones are no used only with the aim to establish a conversation, more and more it is has turned into "mobile information devices" that permit to access to management and personal information in timely way.

The net programs play an important role in the development of wireless applications, which take advantage of the connectivity that these devices offer. However some areas still is find, –like communication with Web servers– that don't have being sufficiently made good use of that, due to programmers aren't developing or because don't exist economic resources.

This prototype attempts to take to the maximum the benefits of Java, to offer a work tool to the software developers, so that allow them to create Internet wireless applications faster and easier. With this tool is facilitate the use of HTTP and HTTPS protocols, sockets and datagrams, all with CLDC 1.0, CLDC 1.1 configurations and MIDP 1.0 and MIDP 2.0 profiles. This tool tries to do easier development and implementation of this type of applications, because is easy to acquire (free software), in such a way contributes to impel its creation; this implies a increment in acquisition of applications on the part of interested people.

The prototype and the software application for tests, was create with the language programming Java, using the NetBeans IDE; the software model process that was use to develop this tool was the evolutionary prototyped, supported in the Unified Modeling Language (UML).

[*] Degree Project in the investigation modality

[**] Physical-Mechanical Engineering Faculty, Systems and Informatics Engineering School, The Director: Systems Engineer MSc. Fernando Antonio Rojas Morales, Codirector: Systems Engineer Edward José Beltrán Lozano.

INTRODUCCIÓN

El crecimiento de la tecnología en los últimos años en áreas como la computación y las comunicaciones, el avance en las redes que ofrecen el servicio de Internet junto con un ancho de banda cada vez mayor, ha hecho que no sólo las empresas, sino también la gente del común vuelva la mirada hacia esta evolución tecnológica, ya sea por entretenimiento o por trabajo. De una u otra manera el mundo de hoy ha creado la necesidad de tener toda clase de información disponible en cualquier momento y en cualquier lugar. Esta situación ha contribuido también a que la adquisición por parte de los usuarios, de esta tecnología sea cada vez más fácil, no sólo de servidores y computadores de escritorio, sino también de computadores portátiles y dispositivos móviles, ya sean celulares o Personal Digital Assistants (PDAs). Últimamente se destaca una preferencia especial por celulares de alta tecnología ó PDAs, debido a todas las alternativas que ofrecen, sobretodo en cuanto a conectividad.

Esta evolución no sólo en las redes de comunicación, sino en los dispositivos móviles ha contribuido a cambiar el estilo de vida de las personas y ha hecho cambiar la forma de hacer las transacciones comerciales. Hoy en día no es extraño ver a una persona consultando el saldo de su cuenta bancaria desde su celular, realizando un pedido ó haciendo cualquier otro tipo de operación de manera electrónica.

Ahora los desarrolladores especializados en elaborar programas para dispositivos móviles, no sólo se encargan de crear aplicaciones para éstos sino que también deben ocuparse en cómo establecer una conexión con los distintos servidores Web de las empresas que ofrecen este servicio. Sin embargo es un área que no ha sido explotada suficientemente ya sea por falta de tiempo, esfuerzo o recursos económicos.

Debido a estas circunstancias se pensó en la elaboración de una herramienta para apoyar a los desarrolladores en la labor de crear aplicaciones Wireless Internet de una manera más rápida y sencilla. Se espera que tenga una amplia acogida ya que puede ser de gran utilidad, en el campo de las telecomunicaciones.

El presente informe se encuentra dividido en capítulos, en los cuales se pueden encontrar los siguientes temas: en el primer capítulo se encuentra la explicación detallada del proyecto; en el segundo se encuentra el marco teórico donde se describen las diferentes tecnologías asociadas a la investigación; en el tercer capítulo se encuentran las fases de diseño e implementación del prototipo y las diferentes pruebas realizadas; en el cuarto la descripción del software aplicativo y por último, en el quinto y sexto capítulo

las conclusiones y recomendaciones, seguidas de las referencias bibliográficas y los anexos.

1. DESCRIPCIÓN DEL PROYECTO

En este capítulo se encuentra la definición del problema del cual fue objeto esta investigación, se describen algunas debilidades en el área que se maneja y los efectos que éstas producen, los objetivos trazados y el impacto estimado con el desarrollo de este proyecto.

1.1 DEFINICIÓN DEL PROBLEMA

"…La tecnología es uno de los aspectos que más incidencia tiene dentro de las organizaciones del futuro y las empresas colombianas la están adquiriendo para su beneficio. La computación móvil es el nuevo panorama de la tecnología; se adapta fácilmente a la mayoría de entornos empresariales, principalmente a aquellos donde el trabajo se realiza fuera de las instalaciones de la organización, permite dar valor agregado a las operaciones empresariales y se integra con el software que poseen las empresas como en el manejo de relaciones con el cliente, mantenimiento de red de servicios, fuerza de ventas, entre otros. En definitiva, la computación móvil extiende el modelo de negocio actual para dar paso a la automatización de los procesos. Con la computación móvil se tiene acceso a la información en tiempo real y capacidad de procesamiento al lugar donde ésta se porte, dejando a un lado aquellos sistemas centralizados y dando mayor uso a la información que reposa estática en los servidores de las empresas…"[7].

El estándar inalámbrico más usado es WAP (Wireless Application Protocol), el cual define las tecnologías para el desarrollo y despliegue de las aplicaciones Wireless Internet. Para el desarrollo de aplicaciones en el cliente, una de las tecnologías que compite en conquistar la atención de los desarrolladores es la plataforma Java ME, ya que puede aplicarse para proyectos que van desde smart phones hasta PDAs, teniendo en cuenta también que es una buena opción para el desarrollo de aplicaciones tipo smart client y cuenta con un entorno de desarrollo bastante confiable.

La plataforma Java Micro Edition (JME) es una colección de APIs[1] en Java orientadas a productos de consumo como PDAs, teléfonos móviles ó electrodomésticos. Java ME se ha convertido en una buena opción debido a que la aplicación se puede emular en un PC durante la fase de desarrollo y luego subirla al dispositivo. Al utilizar tecnologías Java, el desarrollo de aplicaciones o videojuegos con estas APIs resulta bastante económico de portar a otros dispositivos.

[1] API: Application Programming Interface – Interfaz de Programación de Aplicaciones

Sin embargo, las APIs de Java en comunicaciones son muy complejas al momento de utilizarlas, porque no existe una documentación clara y eficiente para su uso en JME y el programador tiene que ir prácticamente a bajo nivel para poder entenderlas. Los diferentes protocolos que existen para comunicaciones, como HTTP, HTTPS, están organizados en diferentes APIs y paquetes dificultando su consulta y utilización. Adicionalmente estas APIs de Java no son suficientes ya que no cuentan con todas las funciones que caracterizan una comunicación completa entre un cliente y un servidor tradicionales, sólo se pueden implementar las funciones más básicas como conectar para visualizar información y desconectar.

Al implementar aplicaciones móviles con servidores Web y cada vez que se desarrolla una de ellas con arquitectura Wireless Internet se debe probar de acuerdo con la configuración del servidor Web y del dispositivo móvil, esto genera un mayor grado de dificultad para el programador, así como mayor cantidad de tiempo invertido en el desarrollo de la aplicación y además problemas de conexión, lo cual produce demoras en los tiempos de entrega.

Actualmente existen algunos componentes para desarrollo de aplicaciones Wireless Internet, pero no tienen una documentación bien definida, algunos no funcionan o no son compatibles en configuraciones CLDC[2] (Connected Limited Device Configuration) anteriores y perfiles MIDP[3] (Midlet Device Profile) de las versiones de la KVM[4] de Java en los dispositivos móviles. Una de las razones para que se presente esta situación, podría ser la escasa investigación en desarrollo de frameworks para dispositivos móviles y porque las pocas aplicaciones que existen e implementan todas las funciones necesarias en una comunicación inalámbrica como descarga de archivos por ejemplo, no son herramientas libres o de fácil adquisición debido a sus elevados costos.

Debido a que hace falta documentación ordenada y no existen manuales prácticos de aplicaciones con arquitectura Wireless Internet, se requiere de mucha experiencia en Java por parte del desarrollador, ya que para utilizar completamente un protocolo, hay que implementar varias APIs en el aplicativo porque no existe una unificada.

De lo anterior nace la idea de crear un componente que facilite el desarrollo, la implementación y el trabajo con comunicaciones en dispositivos móviles basados en JME, que surge como respuesta a algunas necesidades identificadas principalmente en el manejo y documentación de las APIs, la idea de crear un proyecto que reúna gran cantidad de la información necesaria para facilitar la creación de las aplicaciones basadas en la arquitectura Wireless

[2] CLDC: Define el conjunto base de APIs y una máquina virtual para dispositivos de recursos limitados
[3] MIDP: Perfil para dispositivos de información móviles. Se combina con la CLDC para proporcionar un entorno de ejecución para dispositivos móviles.
[4] KVM: Máquina virtual de Java para dispositivos móviles

Internet, así como también la herramienta que permita agilizar el trabajo de los desarrolladores de este tipo de aplicaciones.

1.2 OBJETIVOS

1.2.1 Objetivo General. Diseñar, desarrollar e implementar un prototipo de componente software para la comunicación con servidores Web que facilite el desarrollo de aplicaciones Wireless Internet en J2ME, en dispositivos móviles celulares.

1.2.2 Objetivos Específicos. Estos son los objetivos específicos propuestos en el plan de proyecto [6]:

- Diseñar el prototipo de componente software para aplicaciones Wireless Internet utilizando UML.
- Desarrollar un prototipo de componente software utilizando J2ME, que facilite el desarrollo de aplicaciones Wireless Internet con los protocolos HTTP, HTTPS, las configuraciones CLDC 1.0, CLDC 1.1 y los perfiles MIDP 1.0 y MIDP 2.0.
- Desarrollar un aplicativo software para las pruebas de conexión entre el servidor y el dispositivo móvil celular utilizando J2ME que verifique la interoperabilidad entre ellos.
- Desarrollar el paquete para su posterior instalación y utilización empleando la herramienta NetBeans.

1.3 JUSTIFICACIÓN, IMPACTO Y VIABILIDAD

1.3.1 Justificación. Debido a las necesidades que han surgido en las empresas de instalar software para poder mantener su información lo más actualizada posible y estar siempre a la vanguardia para poder ofrecer a sus clientes mayor calidad en el servicio, se han visto en la necesidad de hacer que esa información sea portable, especialmente en las organizaciones donde la mayoría de los empleados no pueden estar todo el tiempo en las instalaciones de la empresa. Esto ha llevado en los últimos años a un aumento en la implementación de la tecnología móvil y por lo tanto a ciertos adelantos, por este motivo es una tecnología todavía en proceso de desarrollo y maduración.

Como este desarrollo es relativamente nuevo, no hay todavía investigaciones muy conocidas que puedan dar un aporte significativo al desarrollo de frameworks y las que se conocen y son confiables son muy costosas.

Por estos motivos y por las dificultades y necesidades que se han encontrado en el momento de llevar a cabo el desarrollo e implementación de las aplicaciones basabas en la arquitectura Wireless Internet descritas anteriormente, nace la idea de crear un proyecto que permita integrar la información necesaria para dar un mejor uso a las APIs de Java, así como también un prototipo de componente software que permita facilitar y agilizar el trabajo de los desarrolladores de este tipo de aplicaciones.

La idea principal es crear una clase unificada de Java, que permita implementar algunas funciones de comunicaciones entre un servidor Web y un cliente móvil, basadas en el protocolo HTTP, que permitan implementar funciones diferentes a las tradicionales, tales como obtener información de un servidor URL, enviar información a un servidor para obtener una respuesta, transmisión de diferentes clases de archivos, entre otras.

Se proyecta que el desarrollo de este prototipo de componente software se convierta en una herramienta muy útil y de mucha ayuda para los desarrolladores de aplicaciones móviles inalámbricas. Especialmente se haría un aporte bastante significativo, en lo que concierne a ahorrar trabajo y por consiguiente tiempo, en el momento de desarrollar, implementar y hacer las pruebas a este tipo de aplicaciones.

1.3.2 Impacto. A nivel científico con el desarrollo de este proyecto, se espera proporcionar a los desarrolladores de software que hacen énfasis en las aplicaciones móviles inalámbricas y en comunicaciones entre un servidor Web con un cliente móvil, basados en la arquitectura Wireless Internet, una herramienta de carácter tecnológico, que facilite el desarrollo, implementación y puesta en marcha de estas aplicaciones y complementar e integrar los servicios y funciones de las APIs de JME.

En el aspecto económico con la implementación de este prototipo de componente software, se pretende ofrecer una herramienta que sea de fácil adquisición para los desarrolladores, contribuyendo de esta manera a dar un impulso a la generación de un mayor número de aplicaciones móviles, en un menor tiempo y con menor trabajo, es decir, incrementar el rendimiento de las personas interesadas en el desarrollo de estas aplicaciones, ya que tendrían acceso a la herramienta sin ningún costo.

En el aspecto social la idea es incrementar la adquisición de aplicaciones móviles inalámbricas por parte de los empresarios, ya que están optando por implementar este tipo de tecnologías, debido a que pueden tener sus bases de

datos actualizadas en tiempo real y ofrecer mejor calidad de servicio tanto a sus clientes, como a sus proveedores.

1.3.3 Viabilidad. Al analizar la viabilidad de este proyecto, se tuvo en cuenta que:

A nivel científico los recursos tecnológicos para desarrollar este proyecto son habituales en el mercado actual, también se tiene fácil acceso a las herramientas de desarrollo de software.

A nivel económico este proyecto se considera viable debido a la alta demanda de computadores, celulares y el fácil acceso a Internet, que hacen que los costos de estos recursos disminuyan, logrando que la mayoría de personas tengan acceso a ellos, además las tecnologías de desarrollo software a utilizar, no generan costos al ser de libre distribución.

En el aspecto social se pretende con este proyecto promover el uso de las tecnologías móviles e inalámbricas, en las personas del común, para que ellos se familiaricen con ella, con el fin de que puedan ir a la par con la constante evolución de estas tecnologías.

2. MARCO TEÓRICO Y METODOLÓGICO

En este capítulo se encuentra una breve descripción de los diferentes protocolos y tipos de servidores que se trabajaron con el desarrollo de este proyecto, la metodología aplicada, una descripción global del lenguaje Java, una visión general de la tecnología JME y termina con las herramientas utilizadas.

2.1 PROTOTIPO DE SOFTWARE

Un prototipo de software es una versión incompleta del software que se está desarrollando, fácilmente ampliable y modificable, se utiliza para simular aspectos y funcionalidades del producto final. Permite a las partes probarlo en situaciones reales o explorar su uso, creando así un proceso de diseño de iteración que genera calidad. Los prototipos son útiles para comunicar, discutir y definir ideas entre los diseñadores y las partes responsables, también apoyan la evaluación de productos, clarifican requisitos de usuario y definen alternativas. Un prototipo provee los siguientes beneficios:

- Se puede evaluar el diseño e implementación antes de hacerlo.
- Se puede comparar que el producto cumpla con las especificaciones.
- Ayuda a diseñadores y desarrolladores a evaluar propuestas de diseño.
- Ayuda a estimar los plazos de tiempo y recursos necesarios.
- Permite a los usuarios finales probar la interacción con el producto[5].

Los prototipos de baja fidelidad son particularmente útiles en las fases iniciales del desarrollo, durante el diseño conceptual. Los prototipos de alta fidelidad se acercan más a la idea del producto final.

2.2 INGENIERÍA DEL SOFTWARE BASADA EN COMPONENTES

Los componentes de software se pueden emplear cuando éste está en proceso de construcción. Proporcionan funcionalidad dirigida con interfaces bien definidas que permiten que el componente se integre en el software.

El modelo de desarrollo basado en componentes es evolutivo por naturaleza y exige un enfoque iterativo para la creación del software. Conduce a la reutilización del software, lo que reduce el ciclo de desarrollo, los costos del proyecto, incrementa el índice de productividad. Lo cual indica que este

[5] Prototipos de Software [en línea][consultado en 12 diciembre 2012] http://es.scribd.com/doc/53739518/Prototipos-de-Software-Ejemplos-Tutoriales-y-Demo

modelo proporciona ventajas significativas para los ingenieros de software [14].

Una descripción definitiva del término componente, sugiere las siguientes posibilidades[6]:

Componente: parte importante, casi independiente y reemplazable de un sistema que satisface una función clara en el contexto de una arquitectura bien definida.

- *Componente del software en ejecución*: paquete dinámico de unión de uno o más programas gestionados como unidad y a los cuales se tiene acceso por medio de interfaces documentadas que se pueden descubrir en la ejecución.
- *Componente de software*: unidad de composición que sólo tiene dependencias de contexto explícitas y especificadas en forma contractual.

Los componentes de software también se pueden caracterizar con base en sus aplicaciones en el proceso. Además de los componentes, este modelo también produce:
- *Componentes cualificados*: evaluados por ingenieros de software para garantizar que no sólo la funcionalidad, sino el desempeño, la fiabilidad, la facilidad de uso y otros factores de calidad concuerdan con los requisitos del sistema o producto que se construirá.
- *Componentes adaptados*: adaptados para modificar (enmascarar o envolver) características que no se requieren o indeseables.
- *Componentes actualizados*: sustituyen el software existente conforme están disponibles las nuevas versiones de los componentes.

2.3 PROTOCOLOS DE RED

Un protocolo es un método estándar que permite la comunicación entre procesos (que potencialmente se ejecutan en diferentes equipos), es decir, es un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red. El protocolo define cómo se deben comunicar los equipos, es decir, el formato y la secuencia de datos que se van a intercambiar. Existen diversos protocolos de acuerdo a cómo se espera que sea la comunicación.

En Internet, los protocolos utilizados pertenecen a una sucesión de protocolos o a un conjunto de protocolos relacionados entre sí. Este conjunto de protocolos se denomina TCP/IP (Transmission Control Protocol / Internet Protocol). Entre otros, contiene los siguientes protocolos, HTTP (Hypertext

[6] Brawn, A. W., Wallnau K. C. "Engineering of Component-Based Systems", en Component-Based Software Engineering, IEEE Computer Society Press, 1996, pp. 7 - 15.

Transfer Protocol), FTP (File Transfer Protocol), ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol), IP, TCP, UDP (User Datagram Protocol), SMTP (Simple Mail Transfer Protocol), Telnet (Telecommunication Network), NNTP (Network News Transport Protocol).

Generalmente los protocolos se clasifican en dos categorías según el nivel de control de datos requerido:

- **Protocolos orientados a conexión:** estos protocolos controlan la transmisión de datos durante una comunicación establecida entre dos máquinas. En tal esquema, el equipo receptor envía acuses de recepción durante la comunicación, por lo cual el equipo remitente es responsable de la validez de los datos que está enviando. Los datos se envían entonces como flujo de datos. Por ejemplo, TCP es un protocolo orientado a conexión.

- **Protocolos no orientados a conexión:** en éste método de comunicación el equipo remitente envía datos sin avisarle al equipo receptor, y éste recibe los datos sin enviar una notificación de recepción al remitente. Los datos se envían entonces como bloques o paquetes (datagramas). UDP es un protocolo no orientado a conexión.

2.3.1 Protocolo TCP (Transmission Control Protocol) Protocolo de Control de Transmisión. Es uno de los protocolos fundamentales de Internet de la capa de transporte del modelo ISO/OSI y es orientado a conexión, es decir, permite que dos máquinas que están comunicadas controlen el estado de la transmisión. Con el uso del protocolo TCP, las aplicaciones pueden comunicarse en forma segura (gracias al sistema de acuse de recibo del protocolo TCP) independientemente de las capas inferiores. Esto significa que los routers sólo tienen que enviar los datos en forma de datagramas, sin preocuparse por el monitoreo de datos.

Durante una comunicación usando el protocolo TCP, las dos máquinas deben establecer una conexión. La máquina emisora (la que solicita la conexión) se llama cliente, y la máquina receptora se llama servidor. Por eso se dice que se encuentra en un entorno Cliente-Servidor. Las máquinas de dicho entorno se comunican en modo en línea, es decir, que la comunicación se realiza en ambas direcciones. Para posibilitar la comunicación y que funcionen bien todos los controles que la acompañan, los datos se agrupan; es decir, que se agrega un encabezado a los paquetes de datos que permitirán sincronizar las transmisiones y garantizar su recepción.

Otra función del TCP es la capacidad de controlar la velocidad de los datos usando su capacidad para emitir mensajes de tamaño variable. Estos mensajes se conocen como *segmentos*. También proporciona un mecanismo para

distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto[7].

Las principales características del protocolo TCP son las siguientes[8]:

- TCP garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.
- TCP permite el monitoreo del flujo de los datos y así evita la saturación de la red.
- TCP permite que los datos se formen en segmentos de longitud variada para "entregarlos" al protocolo IP.
- TCP permite multiplexar los datos, es decir, que la información que viene de diferentes fuentes (por ejemplo, aplicaciones) en la misma línea pueda circular simultáneamente.

➢ **SOCKETS.** Los sockets se utilizan para establecer comunicación entre dos sistemas, (posiblemente situados en computadores distintos), pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada, estos sockets de red son orientados a conexión, es decir, para que haya comunicación, primero hay que establecer una conexión. Un socket queda definido por una dirección IP, un protocolo y un número de puerto.

2.3.2 Protocolo UDP (User Datagram Protocol). El protocolo UDP (Protocolo de Datagrama a nivel de Usuario), es un protocolo no orientado a conexión de la capa de transporte del modelo ISO/OSI, perteneciente a la familia de protocolos TCP/IP. Este protocolo no es tan fiable como TCP, pues se limita a recoger el mensaje y enviar el paquete por la red. Para garantizar el éxito de la transferencia, UDP hace que la máquina de destino envíe un mensaje de vuelta. Si no es así, el mensaje se envía de nuevo. Con este protocolo no se establece una conexión entre las dos máquinas[9].

Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación, ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o de recepción. Se usa principalmente en protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y video en tiempo real, donde no es posible realizar

[7] http://www.esdebian.org/wiki/manejo-de-protocolo-tcp-con-wireshark
[8] http://es.kioskea.net/contents/internet/tcp.php3
[9] http://www.colombiestad.gov.co/index.php?option=com_glossary&func=view&Itemid=25&catid=114&term=UDP

retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos.

Este protocolo es muy sencillo ya que no proporciona detección de errores, debido a que es no orientado a conexión. Por lo tanto el encabezado del segmento UDP sólo contiene los siguientes campos[10]:

- **Puerto de origen:** es el número de puerto relacionado con la aplicación del remitente del segmento UDP. Este campo representa una dirección de respuesta para el destinatario. Por lo tanto, este campo es opcional, esto significa que si el puerto de origen no está especificado, los 16 bits de este campo se pondrán en cero. En este caso, el destinatario no podrá responder (lo cual no es estrictamente necesario, en particular para mensajes unidireccionales).
- **Puerto de destino:** este campo contiene el puerto correspondiente a la aplicación del equipo receptor al que se envía.
- **Longitud:** este campo especifica la longitud total del segmento, con el encabezado incluido. Sin embargo, el encabezado tiene una longitud de 4 x 16 bits (que es 8 x 8 bits), por lo tanto la longitud del campo es necesariamente superior o igual a 8 bytes.
- **Suma de comprobación:** es una suma de comprobación realizada de manera tal que permita controlar la integridad del segmento.

➢ **DATAGRAMAS.** Están diseñados para enviar paquetes de datos sobre una red y proporcionan una solución mucho mejor que los sockets en los casos en donde no hay una conexión estable entre dos sistemas, debido a que se consideran sockets de red no orientados a conexión. Los datagramas permiten enviar datos sobre una conexión independientemente de que el receptor en el lado opuesto sea capaz de manejar datagramas.

Cuando se usan datagramas la transmisión siempre se asume como exitosa, pero no proporcionan soporte para reorganizar los paquetes de datos, ni hay garantía de que los paquetes lleguen en el mismo orden en que fueron enviados.

La velocidad es la principal razón del uso de datagramas, debido a que no hay control del orden de los paquetes, la comprobación de su integridad o totalidad. En algunas aplicaciones, como las que hacen envío de audio, la pérdida de algunos paquetes podría ser ruido, por lo que no tiene tanta importancia la integridad de los datos. Sin embargo, los datagramas son la opción ideal donde la velocidad juega un papel muy importante, como en la transmisión de audio y video, ya que gracias al protocolo UDP, si una imagen aparece borrosa porque no han llegado todos los paquetes, casi con toda seguridad la siguiente será totalmente nítida.

[10] http://es.kioskea.net/contents/internet/udp.php3

2.4 SERVIDORES WEB

Básicamente, un servidor Web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor, que se comunican el uno con el otro mediante HTTP. Se pueden usar varias tecnologías en el servidor para aumentar su potencia más allá de su capacidad de entregar páginas HTML; éstas incluyen scripts CGI[11], seguridad SSL y páginas activas del servidor (ASP).

Para este proyecto se trabajó con tres tipos de servidores Web:
- WampServer.
- Apache Tomcat.
- Internet Information Server.

2.5 GENERALIDADES DE JAVA 2 MICRO EDITION

El lenguaje Java es utilizado ampliamente debido a las características inherentes al propio lenguaje, donde las más destacadas son la portabilidad, capacidad de crear una aplicación Java y ejecutarla sobre diferentes sistemas operativos, sin necesidad de reconstruirla. Java también se destaca por la seguridad, basada en la integridad del código intermedio que se obtiene al compilar el archivo fuente Java.

Últimamente Java se está destacando por ser una plataforma de programación destinada a entornos grandes de servidores a través del uso de *Java 2 Enterprise Edition* (J2EE) empleando tecnología *Enterprise JavaBeans* (EJB), servlets y *Java Server Pages* (JSP). En un principio Java fue diseñado para aplicaciones sencillas de la parte cliente, destinadas a equipos de consumo. Ahora con *Java Micro Edition* (Java ME), Java vuelve a sus orígenes.

En 1999 en la conferencia JavaOne, *Sun Microsystems* decidió reagrupar toda la tecnología Java en varias ediciones para abarcar desde las grandes aplicaciones distribuidas hasta el pequeño mundo de las aplicaciones ejecutables en dispositivos móviles. A partir de esta decisión, *Sun* definió el lenguaje Java simplemente como Plataforma Java 2 y redistribuyó esta plataforma en tres grandes ramas, cada una con su conjunto de APIs y herramientas de desarrollo propias.

Según *Sun Microsystems* estas tres grandes ramas son:

1. **Java 2 Standard Edition** (J2SE), orientada a computadores de escritorio. Comprende el JDK anteriormente distribuido por *Sun*, donde se han

[11] CGI: Common Gateway Interface

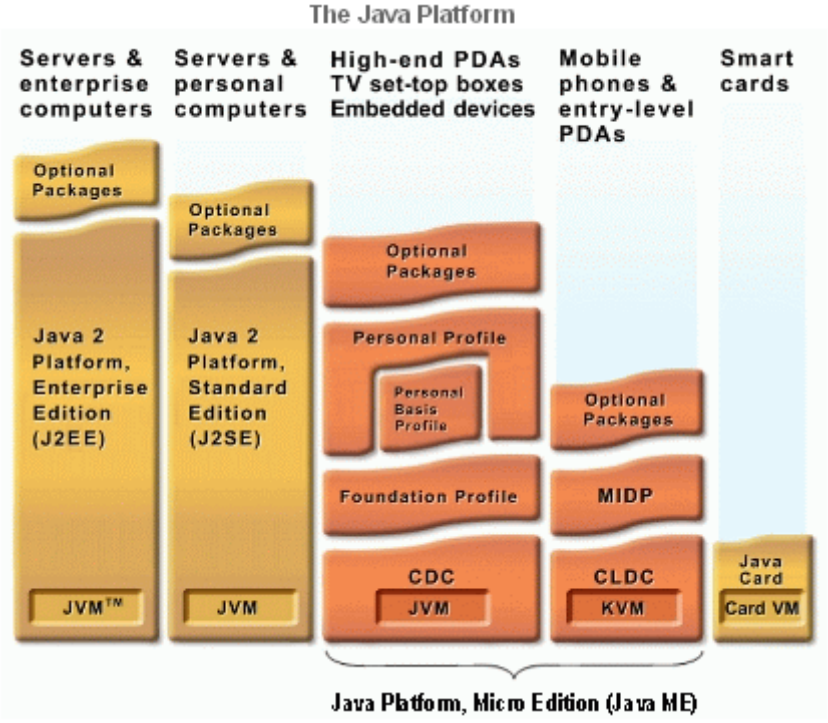
incorporado clases adicionales para facilitar el desarrollo de aplicaciones donde la interfaz de usuario tiene una importancia muy especial.

2. **Java 2 Enterprise Edition** (J2EE), engloba al J2SE y lo potencia añadiéndole clases para el desarrollo de entornos corporativos. Esta edición está orientada al desarrollo de aplicaciones para servidores utilizando Enterprise JavaBeans, aplicaciones Web, Servlets, JavaServer Pages, CORBA y *Extensible Markup Language* (XML). Es decir, esta edición está más orientada al desarrollo de componentes y distribución de aplicaciones.

3. **Java 2 Micro Edition** ahora **Java ME**, es un subconjunto de J2SE orientado al desarrollo de aplicaciones Java destinadas a dispositivos con pocos recursos, con capacidades restringidas, tanto con respecto a la capacidad de memoria disponible, limitaciones de la pantalla gráfica, como con respecto a la capacidad de procesamiento; características típicas no sólo de los teléfonos celulares, sino también de PDAs, buscapersonas y demás equipos de electrónica de consumo [10].

La figura 1 representa una visión general de los componentes de la tecnología Java ME y cómo está relacionada con otras tecnologías Java.

Figura 1. Componentes de Java ME



Fuente: http://java.sun.com/javame/technology/index.jsp#cldc.

Para empezar a hablar detalladamente a cerca de JME, es necesario explicar algunos conceptos.

2.5.1 Configuraciones. Una configuración consiste en un entorno de ejecución Java completo que define el entorno de ejecución básico de JME. Su objetivo es adecuarse a las necesidades de una familia de dispositivos con capacidades similares. Una configuración está formada por tres elementos:

- Una máquina virtual Java para ejecutar el bytecode de la aplicación.
- Código nativo para realizar la interfaz entre Java y el sistema operativo utilizado en el dispositivo.
- Un conjunto de clases Java que constituyen el entorno de ejecución.

Un dispositivo debe cumplir unos requisitos mínimos definidos en la especificación formal de esa configuración para poder utilizarla. Una configuración proporciona un entorno Java completo, pero el conjunto base de clases es muy reducido y debe ser ampliado a través de perfiles o mediante clases propias definidas por el fabricante del dispositivo. Por ejemplo, ninguna configuración tiene clases para implementar la interfaz de usuario, éstas son ofrecidas por los perfiles.

Actualmente existen dos configuraciones: *Connected Device Configuration* (CDC) o Configuración para Dispositivos Conectados y *Connected Limited Device Configuration* (CLDC) o Configuración para Dispositivos con Conexión Limitada. Ambas configuraciones son útiles para dispositivos conectados a redes, ya sean rápidas de tipo LAN o redes lentas inalámbricas.

La configuración CDC está orientada a dispositivos dotados con microprocesadores de 32 bits y que disponen de 2 Mb o más de memoria total, incluyendo memoria RAM y memoria flash o ROM, para la máquina virtual Java y librerías de clases. Deben ser dispositivos con capacidad de conexión a red, generalmente de tipo inalámbrico, lo cual implica inconsistencia en la conexión y un ancho de banda limitado, normalmente 9600 bps o menos. Algunos dispositivos con este tipo de configuración son los *Palmtop-PC* ejecutando Windows-CE[12], Set-Top para televisión digital interactiva, terminales punto de venta, sistemas de navegación para carros, consolas de juegos, cámaras digitales de video o fotografía, impresoras, reproductores de sonido MP3, entre otros. La configuración CDC está asociada al perfil Foundation Profile, que es un API Java orientado a que los distribuidores de estos dispositivos puedan personalizar la interfaz gráfica de las aplicaciones que presentan al usuario.

La configuración CLDC tiene como objetivo mantener en lo posible, las características de Java sobre dispositivos limitados, con procesadores de 16 o

[12] Versión de Windows para sistemas embebidos

32 bits, que tienen entre 8 y 32 MHz y con una cantidad de memoria total disponible para la plataforma Java de al menos 160 – 192 Kb. Para este tipo de dispositivos es imprescindible disponer de una máquina virtual Java adaptada a las restricciones impuestas por las características especiales de los mismos. Esta máquina virtual Java específica es la *Kilo Virtual Machine* (KVM), donde Kilo corresponde a Kilobyte.

Esta configuración es un grupo reducido de APIs (Application Program Interface), útiles para desarrollar las aplicaciones destinadas a un amplio rango de dispositivos. Podría decirse que el CLDC es el conjunto de clases esenciales para construir aplicaciones.

Actualmente existen dos versiones de esta configuración la CLDC 1.0 y la CLDC 1.1. Otros requisitos de hardware que contemplan son:

- Procesador de 16/32 bits.
- Energía limitada, ofrecida por una batería.
- Conectividad a algún tipo de red, aunque posiblemente con ancho de banda limitado (9600 bps o menos).
- Interfaces de usuario con diversos grados de sofisticación.

Los teléfonos celulares, PDAs y terminales de puntos de venta son algunos, pero no todos de los dispositivos que pueden soportar esta configuración. En cuanto a los requisitos de memoria se necesitan 160Kb disponibles para Java, que se utilizan de la siguiente manera:

- 128Kb de memoria no volátil para la máquina virtual Java y para las librerías del API de CLDC.
- 32Kb de memoria volátil para el sistema de ejecución (Java Runtime System).

En cuanto a las limitaciones impuestas por la configuración, CLDC no soporta operaciones en coma flotante. El método *finalize()* fue eliminado de la clase Object, El recolector de basura simplemente recupera un objeto no referenciado, de forma que éstos no se van a resucitar a sí mismos y provoquen una actividad extra al *garbage collector*. El manejo de las excepciones también es limitado en CLDC solamente se definen tres errores en tiempo de ejecución: *Error*, *OutOfMemoryError* y *VirtualMachineError*; la implementación 1.1 agrega *NoClassDefFoundError*. Cualquier otro error será tratado por la máquina virtual Java en función de la implementación, lo cual implica que casi siempre terminará la ejecución [8].

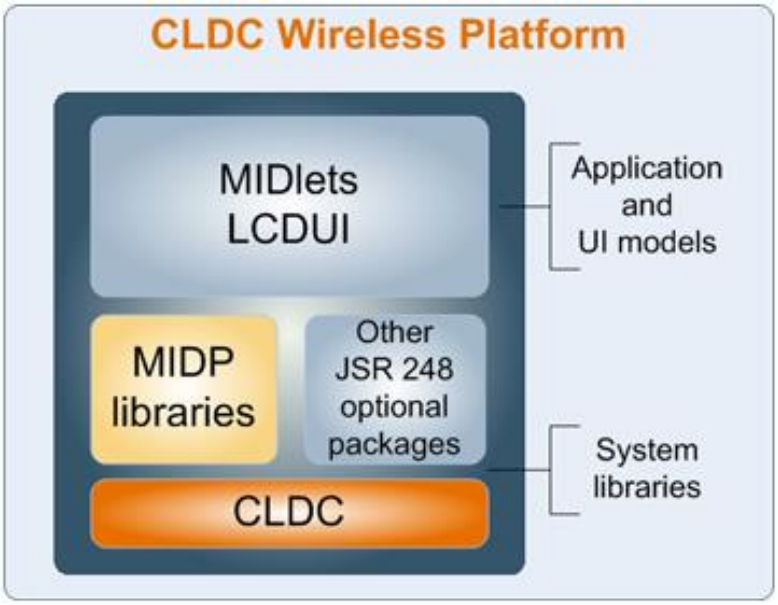
Para este proyecto de investigación se trabajó específicamente con las configuraciones para dispositivos con conexión limitada (CLDC).

2.5.2 Perfiles. El perfil es un grupo más específico de APIs, desde el punto de vista del dispositivo, es decir, la configuración se ajusta a una familia de dispositivos y el perfil se orienta hacia un grupo determinado de dispositivos dentro de dicha familia. El perfil añade funcionalidades adicionales a las proporcionadas por la configuración. Para poder utilizar un perfil, el dispositivo debe cumplir los requisitos mínimos de la configuración en que se basa y también los requisitos adicionales definidos en la especificación formal del perfil.

El perfil MIDP (Mobile Information Device Profile), es el principal y el más utilizado por haber sido el primero del cual se ha proporcionado una implementación, está basado en la configuración CLDC para ejecutar aplicaciones en teléfonos celulares, buscapersonas, dispositivos con pantallas reducidas, conexión HTTP inalámbrica y memoria limitada. El perfil MIDP está promulgado en la especificación JSR-37 y es mejorado en la especificación JSR-118 donde se incorpora conectividad mediante sockets y datagramas, soporte para protocolo HTTPS y SSL, inclusión de la distribución *over-the-air* (OTA) y APIs para el desarrollo de juegos y reproducción de sonido y video.

La figura 2 ilustra la configuración para pequeños dispositivos, es decir, cómo se encuentran estructurados las configuraciones CLDC y los perfiles MIDP.

Figura 2. Configuración para pequeños dispositivos



Fuente: http://java.sun.com/javame/technology/index.jsp#cldc.

2.5.3 MIDLets. Las aplicaciones JME desarrolladas bajo las especificaciones CLDC y MIDP, se denominan MIDLets. Las clases de un MIDLet, son almacenadas en bytecodes Java, dentro de un fichero .class. Estas clases deben ser verificadas antes de su "puesta en marcha", para garantizar que no realizan ninguna operación no permitida. Esta preverificación se debe hacer debido a las limitaciones de la máquina virtual (KVM) para dispositivos embebidos. Los MIDLets generalmente, son juegos y aplicaciones que corren en un teléfono celular.

2.5.4 APIs de CLDC y de MIDP. Los principales elementos involucrados en el proceso de desarrollo con Java, son el lenguaje Java propiamente dicho y el grupo de APIs (Application Programming Interface) que proporcionan el soporte para el software desarrollado.

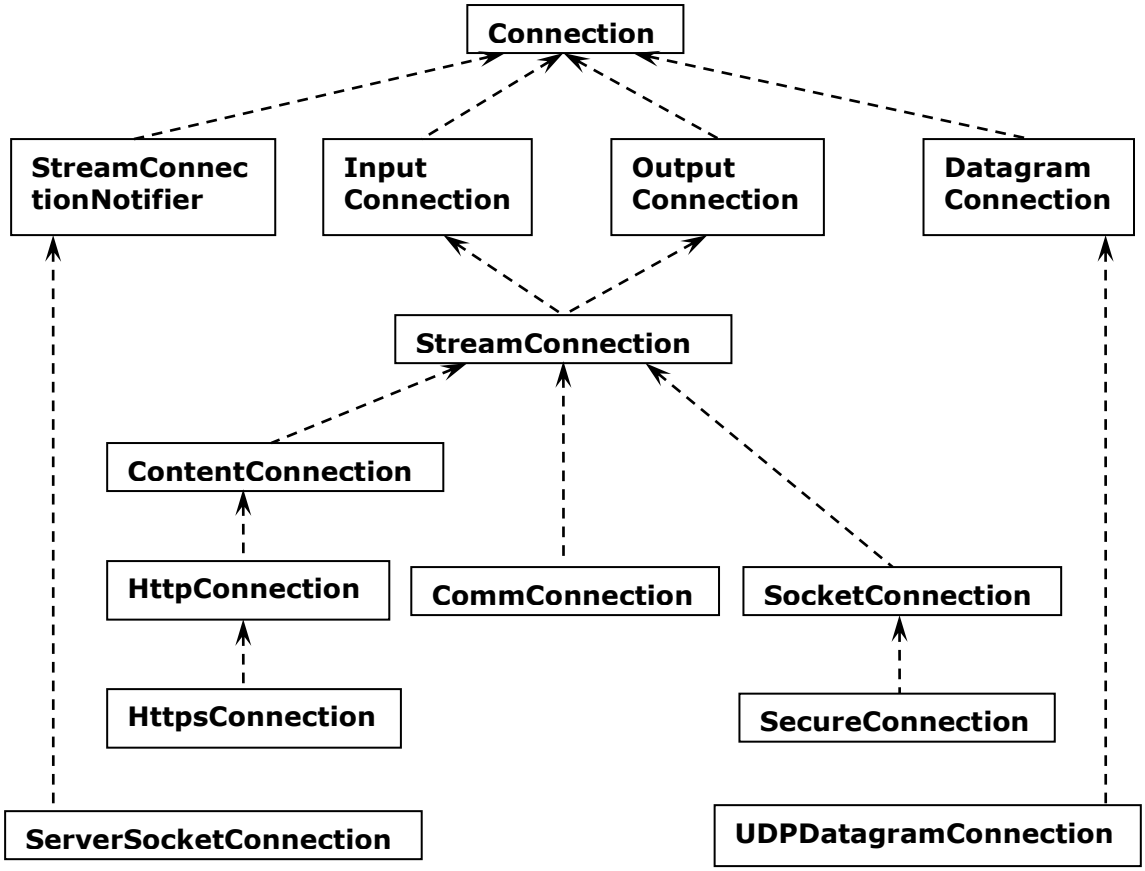
Los APIs específicos para el desarrollo de MIDLets, están descritos en las especificaciones de CLDC y MIDP, que definen la configuración y el perfil para los dispositivos móviles inalámbricos. Estas APIs constan de clases e interfaces presentes en el estándar J2SE así como con clases e interfaces únicos del desarrollo de MIDLets.

El API de CLDC es un pequeño subgrupo del API de J2SE. A parte de estas clases e interfaces, el API de CLDC contiene una serie de interfaces propias, dedicadas a los servicios de red.

2.5.5 KVM. Los conceptos KVM y CVM, acrónimos correspondientes a *Kilo Virtual Machine* y *Compact Virtual Machine*, respectivamente, son los nombres de las máquinas virtuales para la configuración CLDC (KVM) y CDC (CVM), desarrolladas específicamente para su funcionamiento en el entorno restringido de los dispositivos móviles o para ser embebidas en este tipo de dispositivos y para ser fácilmente portables a cualquier otra plataforma diferente de desarrollo.

2.5.6 El GCF (Generic Connection Framework) del CLDC. Como las configuraciones poseen dificultades, para proveer soporte para funciones de red, debido a la variedad de dispositivos, el CLDC delega este trabajo a los perfiles. Para realizar esta operación de forma satisfactoria, el CLDC ofrece un marco general de trabajo en red, conocido como el GCF (Generic Connection Framework). El GCF está compuesto básicamente por una serie de interfaces de conexión, junto con una clase "Connector" que es usada para establecer las diferentes conexiones. La figura 3 ilustra esta jerarquía de interfaces.

Figura 3. Relación de Jerarquía entre las Interfaces del GCF



Fuente: Jenny Paola Montillo Gélvez

2.5.7 La clase Connection. Es el tipo más básico de conexión. Sólo está definido el método *close()*, no está definido el método *open()*, porque esta operación siempre se lleva a cabo usando el método *Connector.open()*.

2.5.8 La clase Connector. Esta clase se utiliza para crear una instancia de un determinado protocolo de conexión, proporcionando métodos que permiten obtener objetos de tipo Connection pasándole una cadena en formato URL. La especificación CLDC proporciona varios tipos de conexiones dependiendo de la cadena que se pase, permitiendo indicar direcciones URL en formato HTTP, pero también permite cadenas menos estándar como *socket:* o *comm:*, que representan sockets o puertos serie, respectivamente.
Todas las conexiones se crean acogiéndose al método estático único *open()* de la clase Connector que utiliza como único argumento cualquier forma de las cadenas que se presentan a continuación:

Forma general	<protocolo>://<destino>;<parámetros>
Archivo	file://prueba.txt
Conexión HTTP	http://www.yahoo.com
Socket	socket://192.168.111.222:7000
Datagrama	datagram://127.0.0.1:8090
Puerto Serie	comm://9600:18N

2.6 METODOLOGÍA DE DESARROLLO

A continuación se hace una breve descripción de las tecnologías aplicadas al proyecto. Para el prototipo se aplicó la metodología de prototipado evolutivo.

2.6.1 Metodología aplicada al proyecto. El modelo de proceso del software, se escogió después de haber realizado un análisis de los modelos más utilizados en el desarrollo de proyectos software.

Los modelos evolutivos son modelos iterativos que permiten desarrollar versiones de software cada vez más completas y funcionales. Dentro de estas metodologías se encuentra el modelo incremental que combina el modelo en cascada pura y el de construcción de prototipos a través de la entrega de incrementos durante el tiempo de desarrollo del proyecto, otros modelos que se encuentran dentro de esta categoría son el prototipado simple y el evolutivo, el modelo en espiral y el de desarrollo concurrente.

El modelo de proceso del software seleccionado para la elaboración de este proyecto, es el Prototipado Evolutivo basado en UML (Lenguaje Unificado de Modelado), para lo que se tuvieron en cuenta las siguientes características del proyecto:

- Los requerimientos y especificaciones sólo están descritos globalmente al inicio del proyecto.
- Las especificaciones deben poder ajustarse a posibles modificaciones y adaptaciones que se hagan en el transcurso del desarrollo del proyecto.
- Funciona con incrementos, los cuales arrojan como resultado un prototipo, para luego agregar nuevas funcionalidades hasta alcanzar lo que el cliente desea.
- Se adapta a la mayoría de las solicitudes de modificación de los clientes.
- No es tan complicado ni requiere de tanta sofisticación como otros modelos, como por ejemplo el espiral.

El prototipado evolutivo comienza con un análisis general de los requerimientos para crear un prototipo inicial. La experiencia del desarrollo del prototipo y su evaluación deben permitir la definición de especificaciones más completas y

seguras para el producto definitivo. Para lograr esto, lo primero que hay que hacer es realizar una especificación del sistema, a partir de los requisitos globales que se pueden conocer inicialmente. En este caso no es necesario realizar una definición completa de los requisitos. Luego se procede a realizar un diseño rápido a partir del cual se construye un prototipo que será evaluado por el cliente quien suministra una realimentación que facilitará una validación y especificación más exacta de lo que desea, con lo que es posible refinar los requisitos para diseñar y construir el siguiente prototipo tomando como base los diseños y el código que ya se tiene.

Algunas de las ventajas que ofrece este modelo son:

- Trabaja con poca identificación de los requerimientos.
- Genera un sistema altamente fiable.
- Permite modificaciones a medio camino.
- Ofrece signos visibles de progreso.
- Al final del proyecto se entrega un software que cumple con todas las exigencias del cliente.

La figura 4, describe un esquema de flujo de trabajo a través de las fases que se ejecutan para obtener un producto final. El proceso parte de un concepto inicial donde se definen el análisis y especificaciones de los requerimientos iniciales.

Figura 4. Proceso del Modelo Prototipado Evolutivo

Fuente: Ingeniería del Software – Un enfoque práctico, Roger PRESSMAN. Sexta Edición. McGraw Hill, Mexico, 2005.

➢ **UML (Unified Modeling Language).** El UML es un lenguaje gráfico creado para ofrecer una solución a la hora de construir o planificar un sistema software o cualquier otro modelo de negocio.

Es un lenguaje de modelado y no un método. Los métodos consisten en su mayoría en un leguaje y un proceso para modelar. El lenguaje de modelado es la notación principalmente gráfica, de que se valen los métodos para expresar

los diseños. El proceso es la orientación que nos dan sobre los pasos a seguir para hacer el diseño. El lenguaje de modelado es la parte más importante del método, es la clave para la comunicación.

Lo que UML permite es crear un modelo a partir de ciertos elementos y ciertas técnicas, que representan el sistema. Los modelos creados hacen uso de notación gráfica que representa principalmente la información, los procesos y el comportamiento del sistema. Además pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

También permite una abstracción del sistema y sus componentes; UML se usa para especificar, visualizar, construir y documentar los componentes de un sistema software, también para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguajes de programación, así como construir modelos por ingeniería inversa a partir de programas existentes [6].

El rol de los modelos UML se puede sintetizar en:

- Ayudar a entender la información, la función y el comportamiento del sistema, haciendo por tanto más fácil y sistemática la tarea de analizar requerimientos.
- Convertirse en el punto de comparación entre lo logrado y lo planificado.
- Fundamentar el diseño, proporcionando una representación lógica, de la implementación.

2.6.2 Herramientas de desarrollo. Como herramienta de desarrollo se trabajó con el IDE NetBeans 7.4. Entre otras herramientas que se describen en seguida.

IDE NetBeans 7.4. Es un exitoso proyecto que cuenta con una gran base de usuarios, una comunidad en constante crecimiento y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y actualmente Oracle patrocina estos proyectos.

Hoy en día se encuentran disponibles dos productos: NetBeans IDE (Entorno de Desarrollo Integrado) y NetBeans Platform.

NetBeans, el IDE Java de código abierto es una herramienta para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe también un

número importante de módulos para extender el NetBeans IDE, además corre sobre Windows, Linux, Mac OS X y Solaris. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

Ambos productos son de código abierto y gratuitos, para uso tanto comercial como no comercial. El código fuente está disponible para su reutilización de acuerdo con la Common Development and Distribution License (CDDL) v1.0 y la GNU (General Public License) GPL (General Public License) v2 [3].

- **WampServer 2.1.** Es un instalador para Windows que contiene el servidor Apache 2.2.17, la base de datos MySQL 5.5.8 y el lenguaje de programación PHP (Hypertext Preprocessor) versión 5.3.5, de esta manera se pueden visualizar las páginas desarrolladas en este lenguaje. Incluye también las herramientas PHPMyAdmin 3.3.9 y el administrador de servicios Wampserver. WAMP los configura automáticamente para que trabajen juntos. El nombre WAMP viene de las iniciales de Windows, Apache, MySQL y PHP.

- **PHP (PHP Hypertext Pre-processor).** Es un lenguaje de programación interpretado, de propósito general, diseñado originalmente para la creación de páginas Web dinámicas y puede ser embebido dentro de código HTML. Generalmente se ejecuta en un servidor Web, al tomar el código en PHP como su entrada y crear páginas Web como salida. Puede ser desplegado en la mayoría de los servidores Web y en casi todos los sistemas operativos y plataformas sin costo alguno. Fue publicado bajo la PHP License, la Free Software Foundation[13] considera esta licencia como software libre.

- **MySQL.** Es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Tiene un esquema de licenciamiento dual, es decir, por un lado se ofrece bajo la GNU[14] GPL[15] para cualquier uso compatible con esta licencia. Por otro lado si las empresas quieren incorporarlo en productos privativos pueden comprar una licencia específica que permita su uso. MySQL es propiedad y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

[13] Free Software Foundation. Organización creada en octubre 1985 para difundir el software libre
[14] GNU Proyecto iniciado con el objetivo de crear un sistema operativo completamente libre
[15] GNU GPL Licencia Pública General de GNU creada por la Free Software Foundation

- **Apache Tomcat/5.5.25.** Apache Tomcat es una implementación de las tecnologías Java Servlet[16] y JavaServer Pages (JSP). Estas tecnologías son desarrolladas bajo el *Java Community Process* (JCP - Proceso de la Comunidad Java*)*. También es conocido como Jakarta Tomcat o simplemente Tomcat, fue desarrollado bajo el proyecto Jakarta[17] en la Apache Software Foundation[18]. Apache Tomcat es desarrollado en un ambiente abierto y participativo y liberado bajo la *Apache Software License* (Licencia de Software Apache). Se propone ser una comunidad colaborativa de los mejores desarrolladores alrededor del mundo. Potencializa numerosas aplicaciones Web a gran escala, de misión crítica a través de diversas industrias y organizaciones.

- **Java Server Pages (JSP).** Es una tecnología Java que permite generar contenido dinámico para Web, en forma de documentos HTML, XML o de otro tipo. Esta tecnología es un desarrollo de la compañía *Sun Microsystems*. Actualmente está disponible la especificación JSP 2.1.

Las JSPs permiten la utilización de código Java por medio de scripts, además es posible utilizar algunas acciones JSP predefinidas, mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de librerías de etiquetas, también llamadas TagLibs externas e incluso personalizadas.

- **Microsoft Internet Information Server (IIS) 5.1.** El IIS simplifica la publicación de información en Internet o en la Intranet, tanto local como remotamente, incluye una amplia gama de funciones administrativas para controlar sitios Web y el servidor Web. Los servicios que ofrece el IIS son FTP, SMTP, NNTP y HTTP/HTTPS. Con funciones de programación como Páginas Active Server (ASP). o ASP.Net puede crear e implementar aplicaciones Web flexibles y escalables. También pueden ser incluidos los módulos de otros fabricantes, como PHP o Perl [2].

- **Active Server Pages (ASP).** Es una tecnología de Microsoft del tipo "lado del servidor" para páginas Web generadas dinámicamente, que ha sido comercializada como un anexo a IIS. La tecnología ASP está estrechamente relacionada con el modelo tecnológico de su fabricante. Intenta ser solución para un modelo de programación rápida. Lo interesante de este modelo tecnológico es poder utilizar diversos componentes ya desarrollados como algunos controles ActiveX, así como componentes del lado del servidor, tales como "CDONTS", por ejemplo, que permite la interacción de los scripts con el servidor SMTP que integra IIS.

---

[16] Servlet Programa que se ejecuta en un servidor

[17] Proyecto Jakarta es el que se encarga de crear y mantener todas las soluciones open Source creadas para la plataforma Java.

[18] Apache Software Foundation. Organización no lucrativa de una comunidad descentralizada de desarrolladores, que trabajan en proyectos de código abierto y creada para dar soporte a los proyectos de software bajo la denominación *Apache*, incluyendo el servidor HTTP Apache.

3. DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO

El presente capítulo expone cómo se desarrolló el prototipo de software y su evolución, así como las fases de diseño e implementación y las pruebas realizadas al prototipo. Haciendo uso de la metodología de desarrollo seleccionada, el proyecto se llevó a cabo en cuatro iteraciones. En la primera iteración se desarrolló un prototipo que cumplió con un conjunto de requerimientos generales planteados en la primera fase del proyecto. Posteriormente se realizó un refinamiento del prototipo basado en la realimentación, con esto se actualizaron los requerimientos y se cumplió con el objeto de desarrollo del proyecto.

3.1 PROTOTIPO INICIAL – CONEXIÓN BÁSICA

En esta fase se realizaron las siguientes etapas:

3.1.1 Revisión de requerimientos y funcionalidades del prototipo. En esta etapa se determinó el grado en que el componente a producir cumpliría con los requisitos que fueron contemplados en la primera fase. Así mismo, se recopilaron y analizaron los requisitos específicos que surgieron durante esta fase. Para el concepto inicial se decidió nombrar la clase que se iba a desarrollar como *HTTP_EXTENSION*; para este primer prototipo se buscaba establecer una sencilla conexión entre un celular y un servidor Web.

3.1.2 Diseño UML del prototipo inicial. A continuación se presentan los casos de uso, diagramas de secuencia, diagrama de actividades y diagrama de clases del prototipo inicial. Es importante resaltar que todas las figuras y cuadros correspondientes al diseño UML de los cuatro prototipos, son de autoría personal.

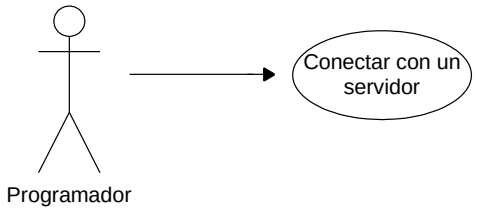
- **Casos de Uso.** Un caso de uso es la descripción de las acciones de un sistema desde el punto de vista del usuario. Para los desarrolladores del sistema, ésta es una herramienta valiosa, ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el punto de vista del usuario. Esto es importante si la finalidad es crear un sistema que pueda ser utilizado por la gente en general (no sólo por expertos en computación) [11].

Las figuras 5 a 8 ilustran los casos de uso del prototipo inicial. Los cuadros 1 a 4 contienen la descripción de los respectivos casos de uso.

- **Diagramas de secuencia.** Un diagrama de secuencias muestra la mecánica de la interacción de un sistema con base en tiempos.

Las figuras 9 a 16 presentan los diagramas de secuencia del prototipo inicial, correspondientes a los casos de uso CU01, CU02, CU03 y CU04.

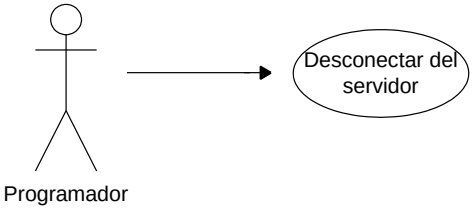
Figura 5. Caso de Uso CU01 – Conectar con Servidor



Cuadro 1. Descripción Caso de Uso CU01 – Conectar con Servidor

Caso de uso	CU01 – Conectar con un servidor.
Actores	Programador
Propósito	Permitir a una aplicación conectarse con un servidor Web.
Descripción caso de uso	Establece una conexión entre un servidor Web y un celular.
Precondición	El programador debe instanciar la clase.
Secuencia normal	El programador instancia la clase. La aplicación capta una URL. La aplicación verifica la URL, el tamaño de la información y el tipo de servidor. La aplicación hace la conexión con el servidor. La aplicación visualiza la información.
Post-condición	La conexión se realizó exitosamente.
Eventos excepcionales	Mensaje de error porque el servidor no está disponible ó porque falla la conexión.

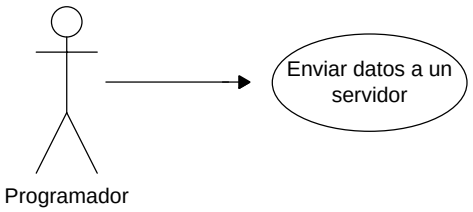
Figura 6. Caso de Uso CU02 – Desconectar del Servidor



Cuadro 2. Descripción Caso de Uso CU02 – Desconectar del Servidor

Caso de uso	CU02 – Desconectar del servidor.
Actores	Programador
Propósito	Permitir a una aplicación desconectarse de un servidor.
Descripción caso de uso	Luego de tener una conexión establecida entre un servidor Web y un celular, permite hacer la respectiva desconexión.
Precondición	Debe haber una conexión establecida.
Secuencia normal	El programador instancia la clase. La aplicación verifica la URL. La aplicación hace la desconexión del servidor.
Post-condición	La desconexión se realizó exitosamente.
Eventos excepcionales	Mensaje de error porque el servidor no está disponible ó porque falla la desconexión.

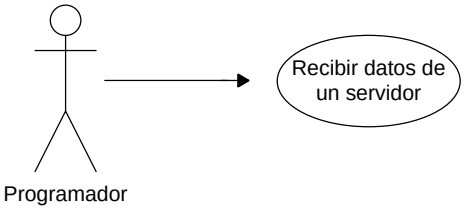
Figura 7. Caso de Uso CU03 – Enviar datos a un Servidor



Cuadro 3. Descripción Caso de Uso CU03 – Enviar datos a un Servidor

Caso de uso	CU03- Enviar datos a un Servidor
Actores	Programador
Propósito	Permitir a una aplicación enviar datos a un servidor.
Descripción caso de uso	Establece conexión con un servidor Web y luego se envía información desde el celular.
Precondición	El programador debe instanciar la clase.
Secuencia normal	El programador instancia la clase. La aplicación capta una URL. La aplicación verifica la URL, el tamaño de la información y el tipo de servidor. La aplicación hace la conexión con el servidor. La aplicación envía la información.
Post-condición	La conexión se realizó exitosamente.
Eventos excepcionales	Mensaje de error porque el servidor no está disponible ó porque falla la conexión.

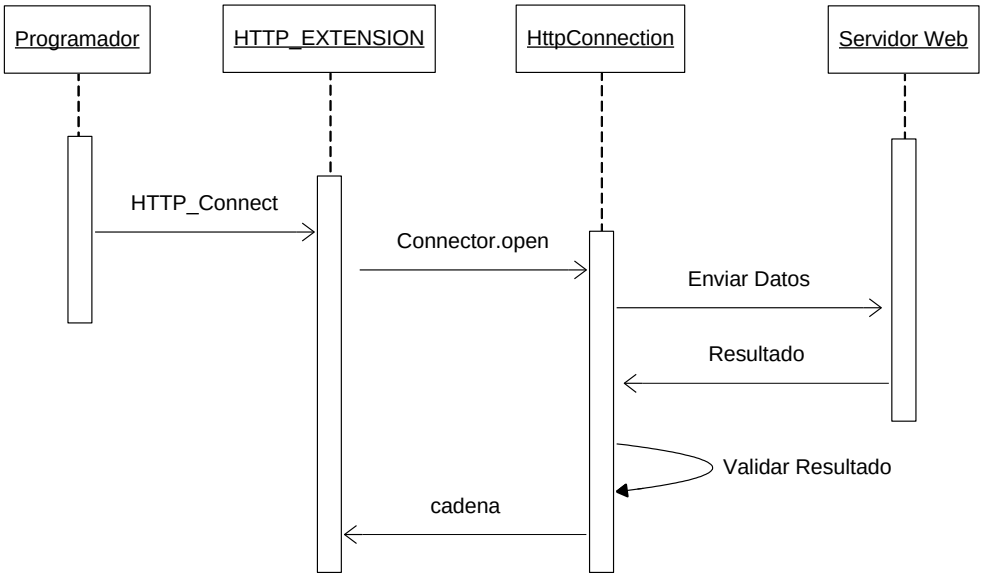
Figura 8. Caso de Uso CU04 – Recibir datos de un Servidor



Cuadro 4. Descripción Caso de Uso CU04 – Recibir datos de un Servidor

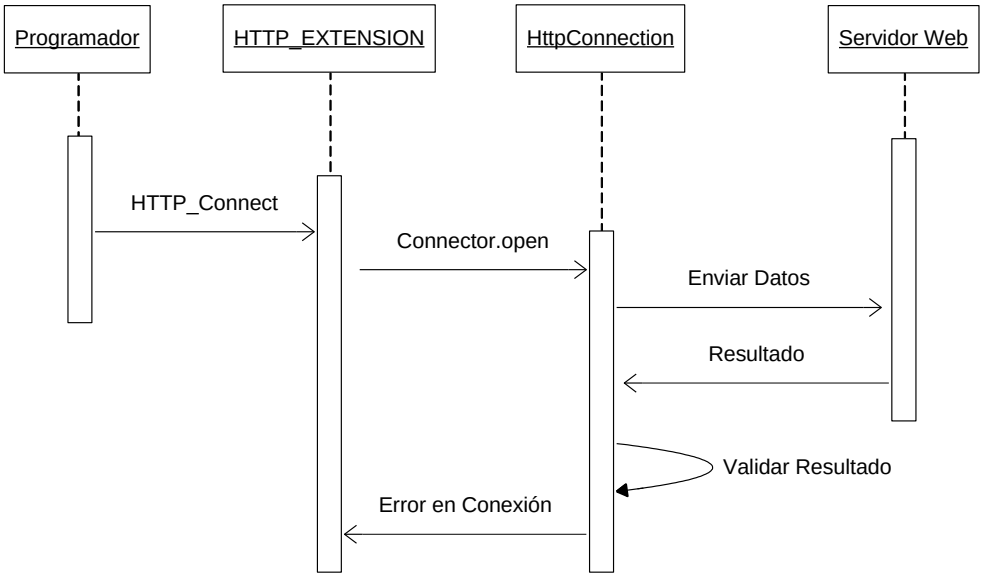
Caso de uso	CU01-Recibir datos de un Servidor
Actores	Programador
Propósito	Permitir a una aplicación recibir datos de un servidor Web.
Descripción caso de uso	Establece conexión con un servidor Web y luego visualiza en la pantalla del celular la información recibida.
Precondición	El programador debe instanciar la clase.
Secuencia normal	El programador instancia la clase. La aplicación capta una URL. La aplicación verifica la URL, el tamaño de la información y el tipo de servidor. La aplicación hace la conexión con el servidor. La aplicación visualiza la información.
Post-condición	La conexión se realizó exitosamente.
Eventos excepcionales	Mensaje de error porque el servidor no está disponible ó porque falla la conexión.

Figura 9. Escenario conexión exitosa del CU01 – Conectar con Servidor



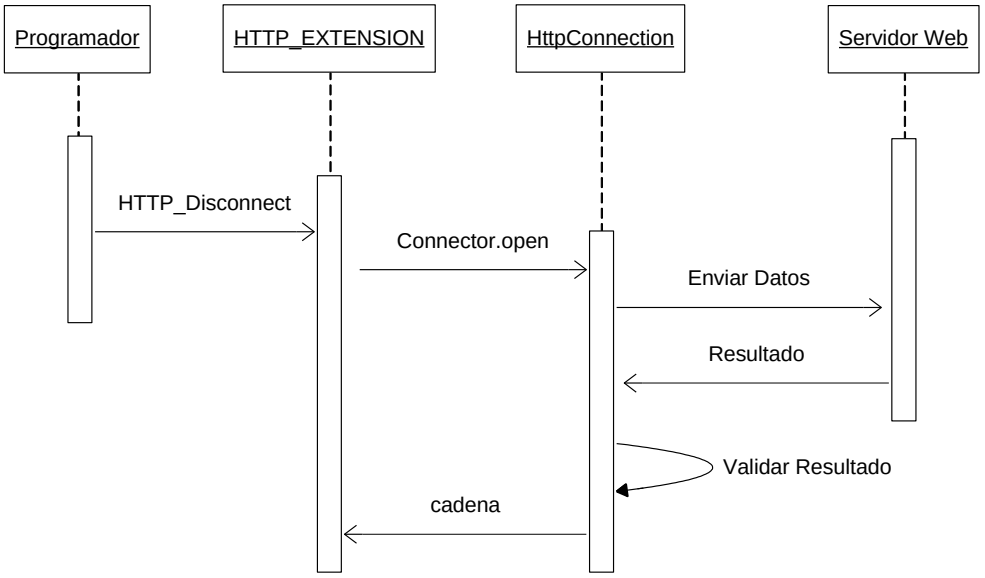
Fuente: Jenny Paola Montillo Gélvez

Figura 10. Escenario conexión fallida del CU01 – Conectar con Servidor



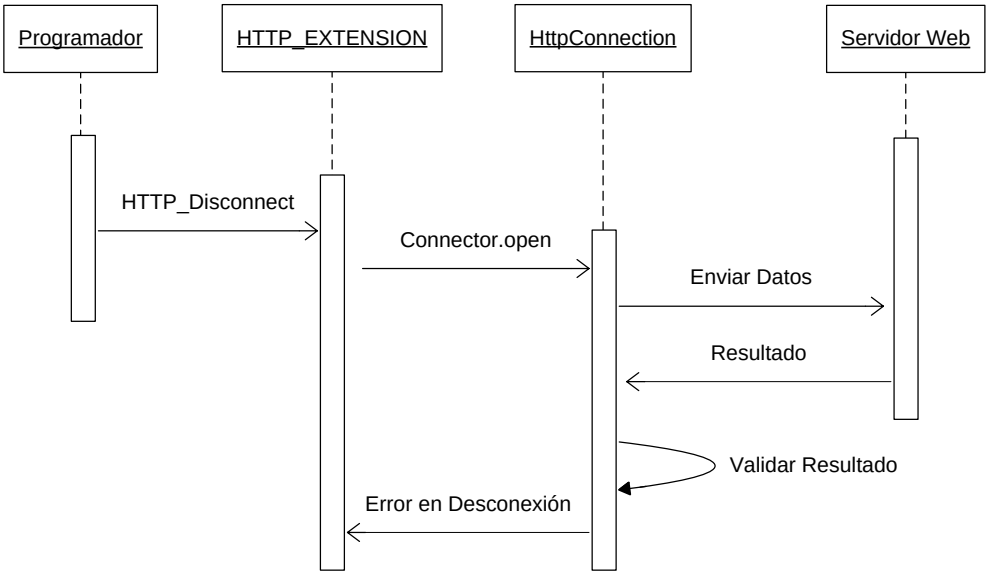
Fuente: Jenny Paola Montillo Gélvez

Figura 11. Escenario desconexión exitosa del CU02 – Desconectar del Servidor



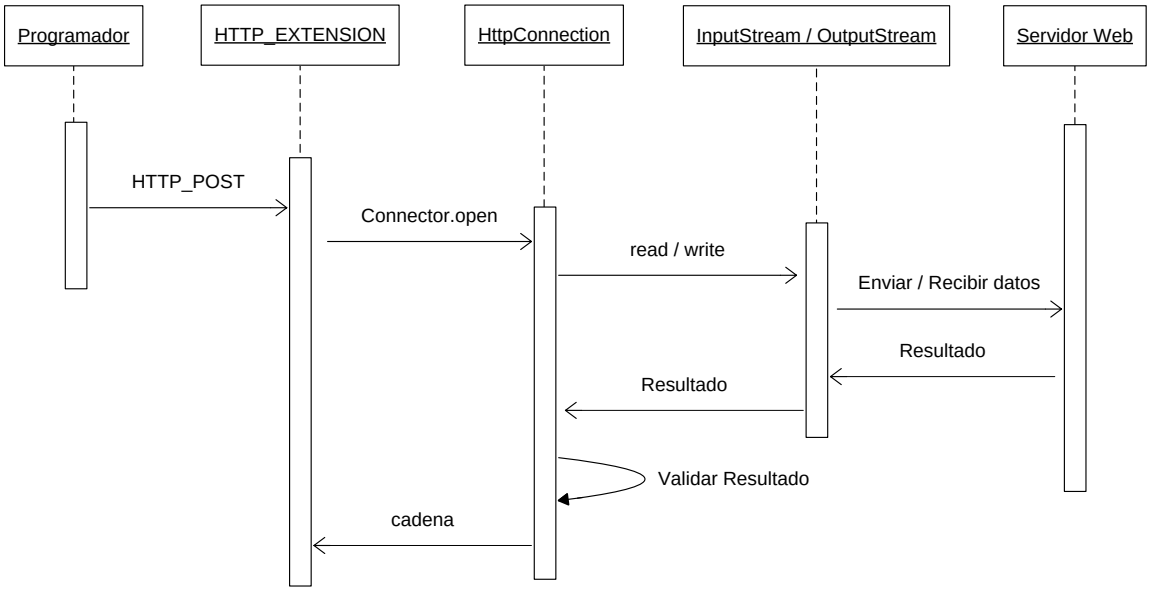
Fuente: Jenny Paola Montillo Gélvez

Figura 12. Escenario desconexión fallida del CU02 – Desconectar del Servidor



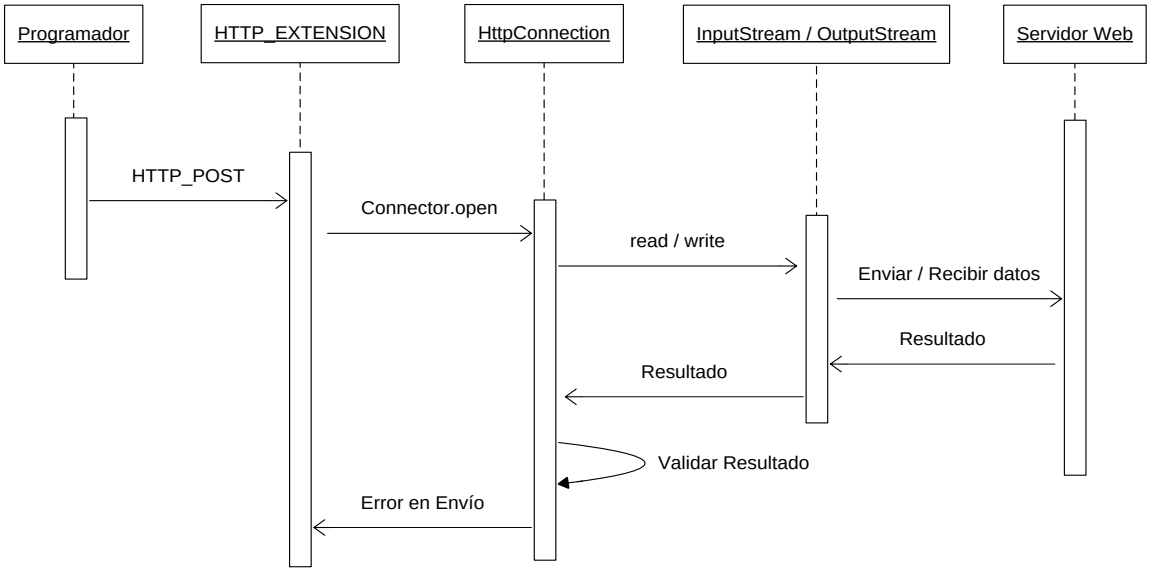
Fuente: Jenny Paola Montillo Gélvez

Figura 13. Escenario conexión exitosa del CU03 – Enviar datos a un Servidor



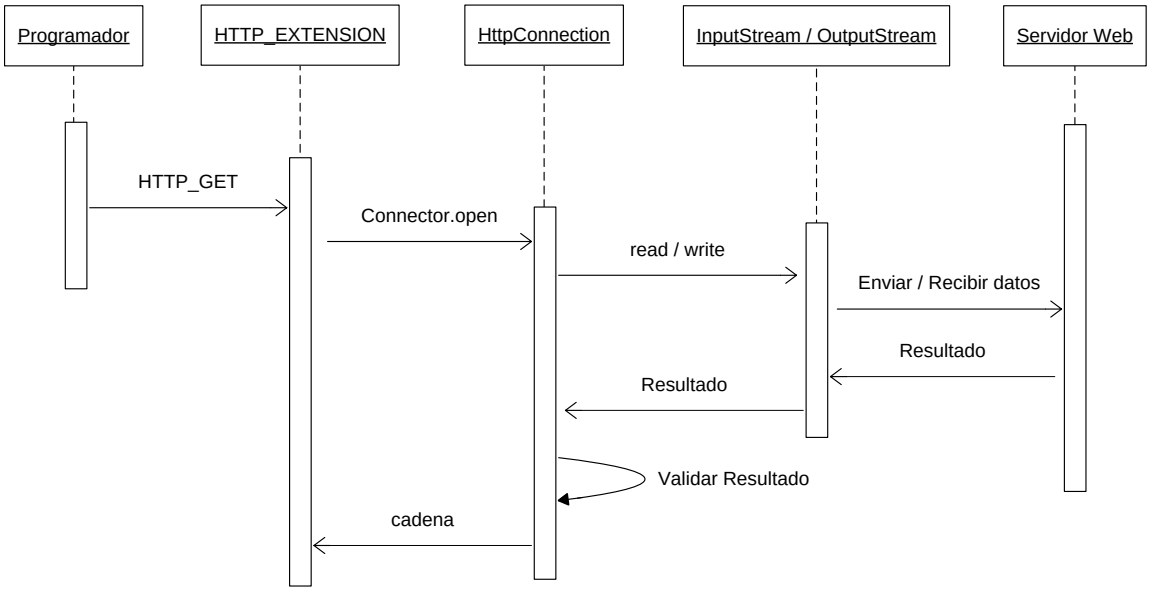
Fuente: Jenny Paola Montillo Gélvez

Figura 14. Escenario conexión fallida del CU03 – Enviar datos a un Servidor



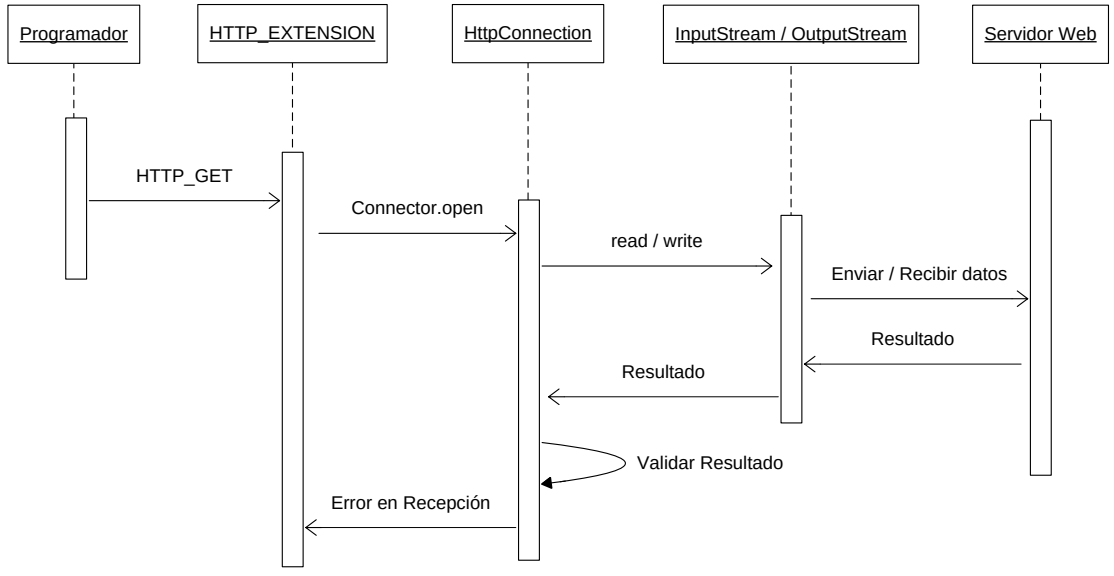
Fuente: Jenny Paola Montillo Gélvez

Figura 15. Escenario conexión exitosa del CU04 - Recibir datos de un Servidor



Fuente: Jenny Paola Montillo Gélvez

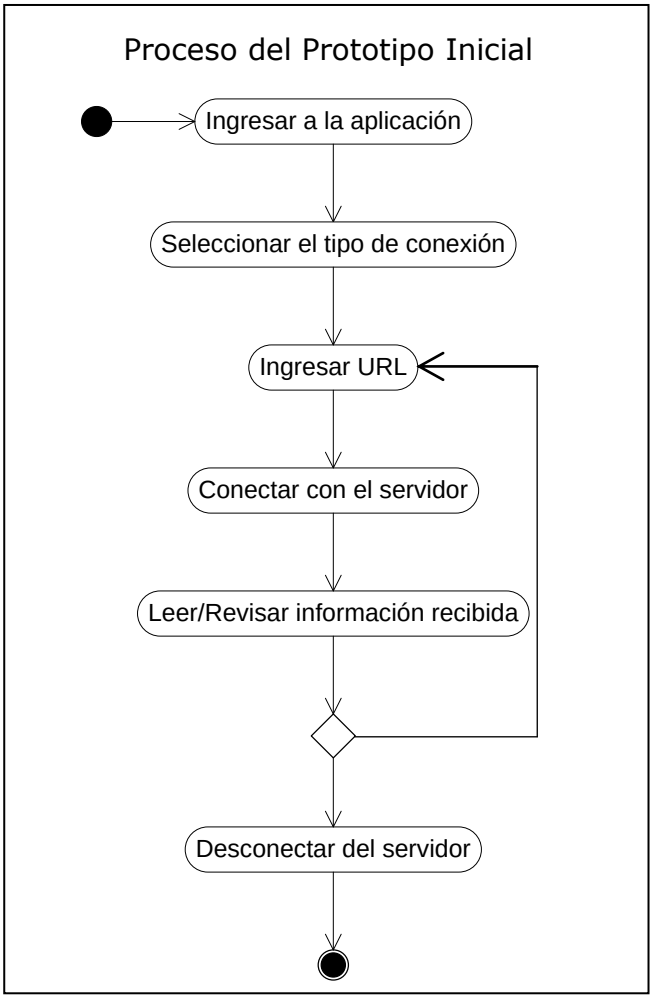
Figura 16. Escenario conexión fallida del CU04-Recibir datos de un Servidor



Fuente: Jenny Paola Montillo Gélvez

➢ **Diagrama de actividades.** El diagrama de actividades describe la secuencia del flujo de trabajo desde el punto inicial, hasta el punto final del proceso. En la figura 17 se presenta el diagrama de actividades del prototipo inicial, correspondiente a los casos de uso CU01 y CU02.

Figura 17. Diagrama de Actividades del prototipo inicial – Conexión Básica

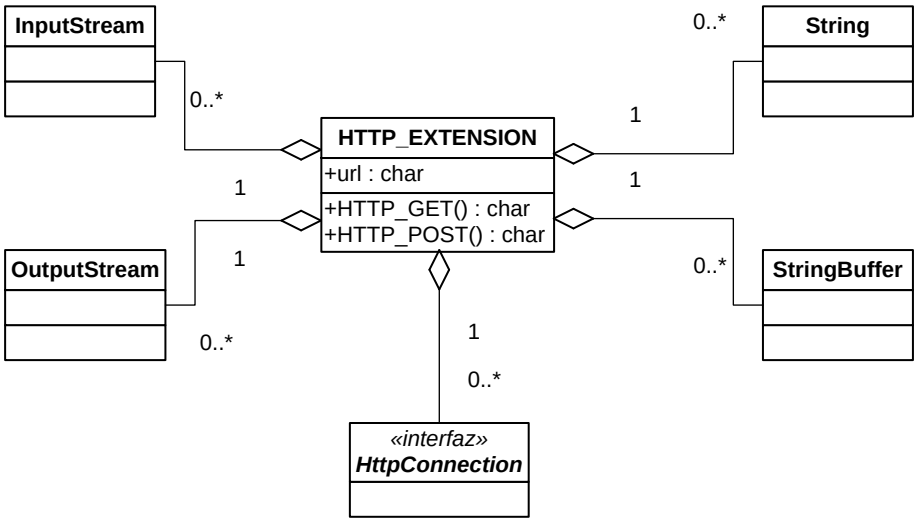


Fuente: Jenny Paola Montillo Gélvez

➢ **Diagramas de clases.** Una clase es una categoría o grupo de cosas que tienen atributos y acciones similares.

La figura 18 ilustra el diagrama de clases del prototipo inicial y el cuadro 5 describe el primer prototipo de la clase HTTP_EXTENSION – Conexión Básica.

Figura 18. Diagrama de clases del prototipo inicial – Conexión Básica



Fuente: Jenny Paola Montillo Gélvez

Cuadro 5. Descripción del primer prototipo de la clase HTTP_EXTENSION – Conexión Básica

Clase: HTTP_EXTENSION
Descripción: Clase para establecer conexiones entre celulares y servidores Web a través del protocolo HTTP.
Módulo: Principal
Propiedades: Concreta
Atributos: url
HTTP_GET(String, int, String) Método para establecer una conexión entre un servidor y un celular, a través del método estándar GET.
HTTP_POST(String, int, String) Método para establecer una conexión entre un servidor y un celular, a través del método estándar POST.

Fuente: Jenny Paola Montillo Gélvez

3.1.3 Desarrollo del prototipo inicial. En esta etapa se realizó la implementación del prototipo definido para el ciclo actual, fundamentándose en el diseño realizado en la etapa anterior.

El producto de esta etapa arrojó como resultado los métodos HTTP_GET y HTTP_POST para la clase HTTP_EXTENSION. El método HTTP_GET sirve para

establecer una conexión al implementar el método estándar Get y el método HTTP_POST, para establecer la conexión con el método estándar POST. Se trabajó con servidores que alojan código PHP, JSP y ASP.
Una vez construido el prototipo inicial, se realizaron las respectivas pruebas y se pudo comprobar su funcionamiento y también que se cumplieran los requisitos especificados en esta fase. Además, se realizó una evaluación de los resultados obtenidos, produciendo información que sirvió de realimentación para refinar los requerimientos y especificaciones.

En las figuras 19 y 20 se puede apreciar parte del código del prototipo inicial.

Figura 19. Código del método HTTP_GET

```
/**
 * Método que devuelve una cadena de texto que contiene el resultado de la conexión
 *
 * @param url Strig que contiene dirección con la cual se quiere establecer la conexión
 * @param tamano Entero que establece el tamaño de los datos que se van a transmitir
 * @param tipoServidor String que define el tipo de servidor con el que se va a establecer la conexión;
 * puede ser PHP, JSP o ASP
 * @return Objeto tipo String que contiene el resultado de la conexión
 * @throws IOException Se genera al producirse un fallo o interupción en operaciones de I/O (Entrada/Salida)
 */
public String HTTP_GET(String url, int tamano, String tipoServidor) throws IOException {

        HttpConnection connection = null;
        InputStream is = null;
        OutputStream os = null;
        StringBuffer stringBuffer = new StringBuffer();
        String cadenas;
        cadenas = new String();

        if ((tipoServidor.compareTo("PHP") == 0) || (tipoServidor.compareTo("JSP") == 0)) {

            try {
                    connection = (HttpConnection)Connector.open(url);
                    connection.setRequestMethod(HttpConnection.GET);
                    connection.setRequestProperty("IF-Modified-Since","20 Jan 2001 16:19:14 GMT");
                    connection.setRequestProperty("User-Agent","Profile/MIDP-2.0 Confirguration/CLDC-1.0");
                    connection.setRequestProperty("Content-Language", "es-ES");
                    connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
```

Fuente: Jenny Paola Montillo Gélvez

En el anexo A se puede apreciar la secuencia gráfica de los resultados obtenidos con el prototipo inicial.

3.2 SEGUNDO PROTOTIPO – VISUALIZAR ARCHIVOS

En las siguientes fases se debe refinar el prototipo hasta que sea aceptable. Se toma el prototipo del resultado de la etapa anterior para ser evaluado por el cliente y será usado para refinar los requisitos del software a desarrollar. Se

continúa con esta secuencia hasta que se satisfacen las necesidades del cliente.

Figura 20. Código del método HTTP_POST

```
/**
 * Método que devuelve una cadena de texto que contiene el resultado de la conexión
 *
 * @param url Strig que contiene dirección con la cual se quiere establecer la conexión
 * @param tamano Entero que establece el tamaño de los datos que se van a transmitir
 * @param tipoServidor String que define el tipo de servidor con el que se va a establecer la conexión;
 * puede ser PHP, JSP o ASP
 * @return Objeto tipo String que contiene el resultado de la conexión
 * @throws IOException Se genera al producirse un fallo o interupción en operaciones de I/O (Entrada/Salida)
 */
public String HTTP_POST(String url, int tamano, String tipoServidor) throws IOException {

    HttpConnection connection = null;
    InputStream is = null;
    OutputStream os = null;
    StringBuffer stringBuffer = new StringBuffer();
    String cadenas;
    cadenas = new String();

    if ((tipoServidor.compareTo("PHP") == 0) || ((tipoServidor.compareTo("JSP") == 0))) {

        try {

                connection = (HttpConnection)Connector.open(url);
                connection.setRequestMethod(HttpConnection.POST);
                connection.setRequestProperty("IF-Modified-Since","20 Jan 2001 16:19:14 GMT");
                connection.setRequestProperty("User-Agent","Profile/MIDP-2.0 Confirguration/CLDC-1.0");
                connection.setRequestProperty("Content-Language", "en-CA");
                connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
```

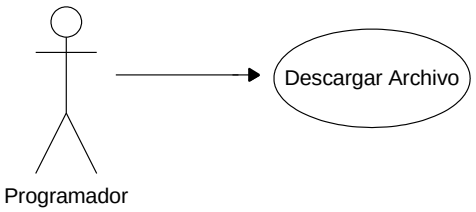
Fuente: Jenny Paola Montillo Gélvez

3.2.1 Revisión de requerimientos y funcionalidades. El segundo prototipo añade nuevas funcionalidades a la clase *HTTP_EXTENSION*; al final de esta fase la aplicación debe permitir descargar y visualizar archivos de texto e imágenes.

3.2.2 Diseño UML del segundo prototipo. A continuación se presentan los casos de uso, diagramas de secuencia, diagrama de actividades y diagrama de clases del segundo prototipo.

- **Casos de uso.** La figura 21 representa el caso de uso del segundo prototipo y, el cuadro 6 presenta la descripción del caso de uso CU05.

Figura 21. Caso de Uso CU05 – Descargar archivo



Fuente: Jenny Paola Montillo Gélvez

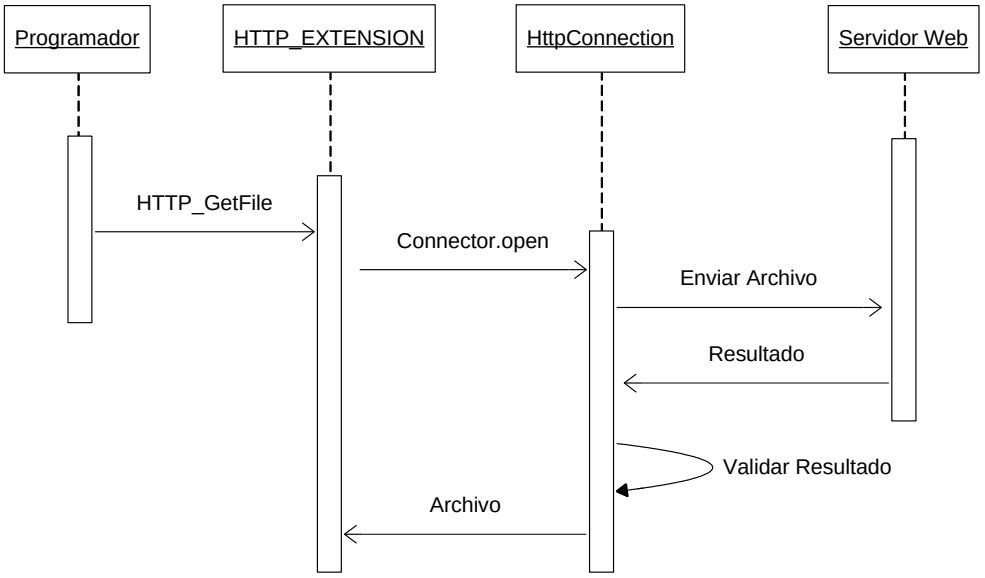
Cuadro 6. Descripción Caso de Uso CU05 – Descargar archivo

Caso de uso	CU03 – Descargar archivo
Actores	Programador
Propósito	Permitir a una aplicación descargar archivos de un servidor.
Descripción caso de uso	Establece conexión con un servidor Web y luego visualiza y guarda en el celular un archivo.
Precondición	El programador debe instanciar la clase.
Secuencia normal	El programador instancia la clase. La aplicación capta una URL. La aplicación verifica la URL, el tamaño de la información, un nombre local y el tipo de servidor. La aplicación hace la conexión con el servidor. La aplicación visualiza el archivo.
Post-condición	La conexión se realizó exitosamente.
Eventos excepcionales	Mensaje de error porque el servidor no está disponible ó porque falla la conexión.

Fuente: Jenny Paola Montillo Gélvez

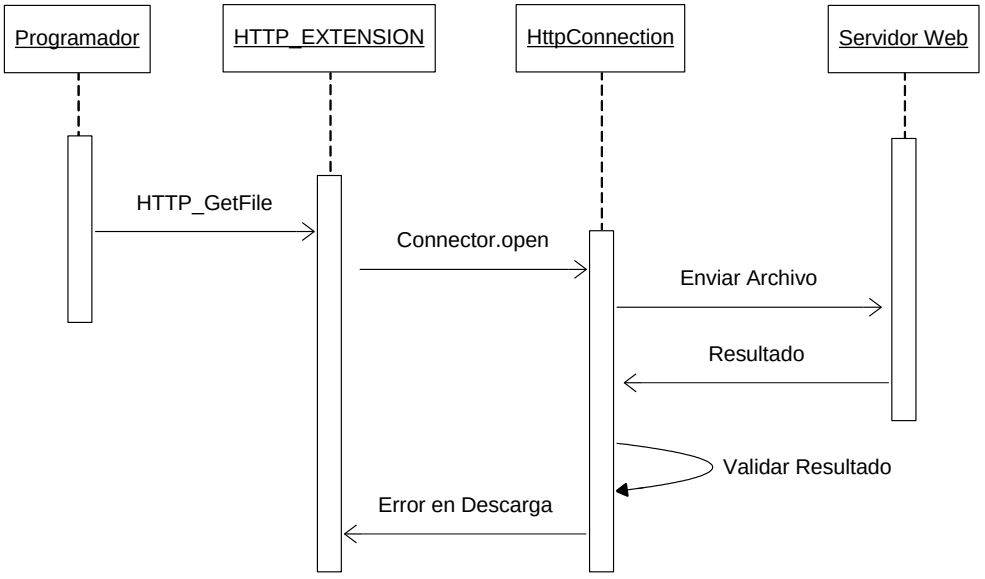
- **Diagramas de secuencia.** Las figuras 22 y 23 representan los diagramas de secuencia del segundo prototipo.

Figura 22. Escenario descarga exitosa del CU05 – Descargar Archivo



Fuente: Jenny Paola Montillo Gélvez

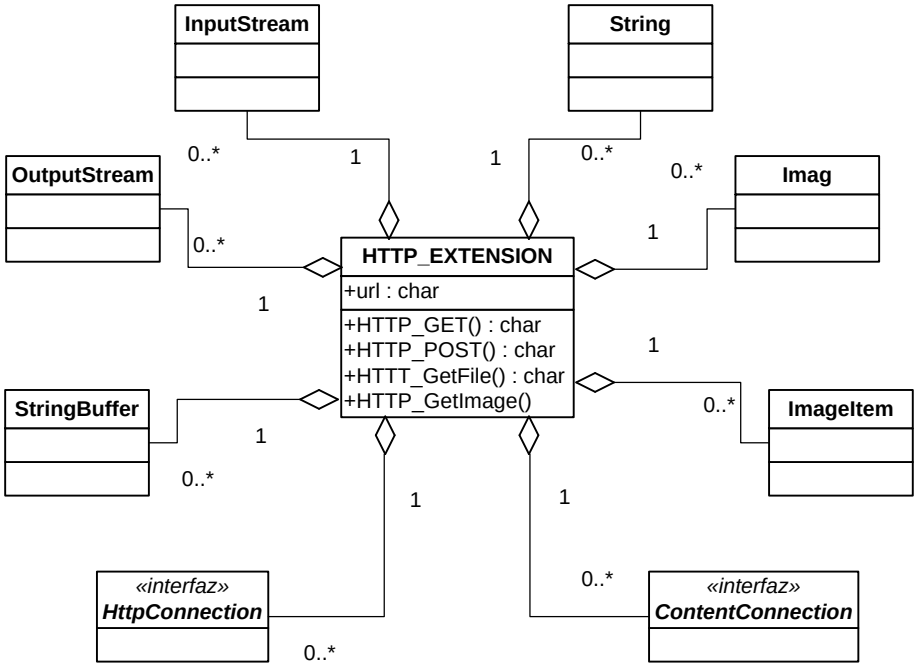
Figura 23. Escenario descarga fallida del CU05 – Descargar Archivo



Fuente: Jenny Paola Montillo Gélvez

➢ **Diagrama de clases.** La figura 24 presenta el diagrama de clases del segundo prototipo y, el cuadro 7 presenta su descripción.

Figura 24. Diagrama de clases del segundo prototipo – Visualizar Archivos

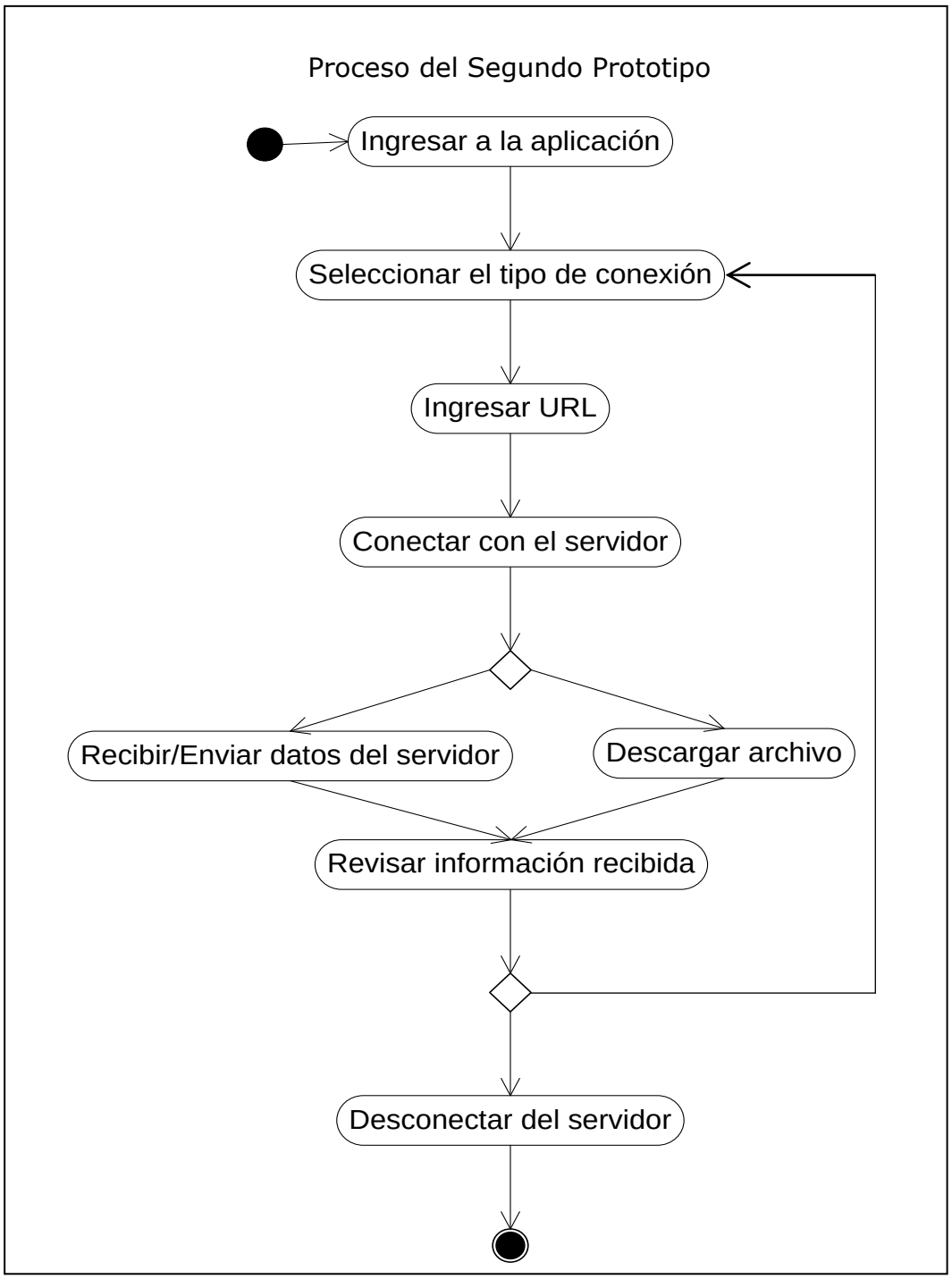


Fuente: Jenny Paola Montillo Gélvez

➢ **Diagrama de actividades.** En la figura 25 se presenta el diagrama de actividades del segundo prototipo, correspondiente al caso de uso CU05.

3.2.3 Desarrollo del segundo prototipo. En el segundo prototipo se agregaron dos nuevos métodos a la clase HTTP_EXTENSION, que fueron HTTP_GetFile, el cual permite visualizar a través de la pantalla del celular, un archivo de tipo txt que esté en el servidor. Y el método HTTP_GetImage que permite visualizar imágenes en los formatos jpg, gif, png y bmp, aunque estas últimas se podrán visualizar dependiendo de las características del dispositivo móvil en el cual se ejecute el aplicativo.

Figura 25. Diagrama de Actividades del segundo prototipo – Visualizar Archivo



Fuente: Jenny Paola Montillo Gélvez

Cuadro 7. Descripción del segundo prototipo de la clase HTTP_EXTENSION – Visualizar Archivo

Clase: HTTP_EXTENSION
Descripción: Clase para establecer conexiones entre celulares y servidores Web a través del protocolo HTTP.
Módulo: Principal
Propiedades: Concreta
Atributos: url
HTTP_GET(String, int, String) Método para establecer una conexión entre un servidor y un celular, a través del método estándar GET.
HTTP_POST(String, int, String) Método para establecer una conexión entre un servidor y un celular, a través del método estándar POST.
HTTP_GetFile(String, int, String, String) Método para establecer conexión con un servidor Web y visualizar el contenido de un archivo.
HTTP_GetImage(String) Método para establecer conexión con un servidor Web y visualizar una imagen determinada.

Fuente: Jenny Paola Montillo Gélvez

En las figuras 26 y 27 se presenta parte del código de los métodos HTTP_GetFile y HTTP_GetImage, respectivamente.

En el anexo A se pueden consultar los resultados gráficos del segundo prototipo.

3.3 TERCER PROTOTIPO – CONEXIÓN TCP/UDP

3.3.1 Revisión de requerimientos y funcionalidades. En esta tercera iteración se agregaron nuevos métodos a la clase HTTP_EXTENSION, estos métodos son el connectSocket, getInput, getOutput, disconnectSocket y el connectUDP, que implementan la conexión utilizando sockets y datagramas respectivamente.

3.3.2 Diseño UML del tercer prototipo. Para el diseño del tercer prototipo no se vio la necesidad de crear nuevos casos de uso ni diagramas de secuencia, pero si se agregaron nuevas clases y funcionalidades al prototipo.

Figura 26. Código del método HTTP_GetFile

```
/**
 * Método que permite visualizar un archivo de texto, que se encuentra alojado en un servidor
 *
 * @param url Strig que contiene dirección con la cual se quiere establecer la conexión
 * @param tamano Entero que establece el tamaño de los datos que se van a transmitir
 * @param nombreLocal String que especifica el nombre del archivo
 * @param tipoServidor String que define el tipo de servidor con el que se va a establecer la conexión;
 * puede ser PHP, JSP o ASP
 * @return Objeto tipo String que contiene el resultado de la conexión
 * @throws IOException Se genera al producirse un fallo o interupción en operaciones de I/O (Entrada/Salida)
 */
public String HTTP_GetFile(String url, int tamano, String nombreLocal, String tipoServidor) throws IOException {

    HttpConnection connection = null;
    InputStream is = null;
    StringBuffer stringBuffer = new StringBuffer();


    long len = 0;
    int ch = 0;

    connection = (HttpConnection)Connector.open(url);
    is = connection.openInputStream();
    len = connection.getLength();

        if(len != -1){
            for(int i = 0; i < len; i++) {
                if((ch = is.read()) != -1) {
                stringBuffer.append((char) ch);
                }
            }
```

Fuente: Jenny Paola Montillo Gélvez

Figura 27. Código del método HTTP_GetImage

```
/**
 * Método que permite visualizar una imagen (jpg, png, gif o bmp), que se encuentra alojada en un servidor
 *
 * @param url Strig que contiene la dirección con la cual se quiere establecer la conexión
 * @return Objeto tipo image que contiene la imagen
 * @throws IOException Se genera al producirse un fallo o interupción en operaciones de I/O (Entrada/Salida)
 */
public Image HTTP_GetImage (String url) throws IOException {

    ContentConnection con = null;
    InputStream is = null;
    byte[] datos = null;
    Image imagen = null;
    ImageItem imgItem = null;

    try {
            con = (ContentConnection)Connector.open(url);
            is = con.openInputStream();
            int longitud = (int)con.getLength();

            if (longitud > 0){
                int offset = 0;
                datos = new byte[longitud];

                while (offset < longitud) {
                    int leer = 1024;

                    if ((longitud-offset) < 1024) {
                            leer = longitud-offset;
```

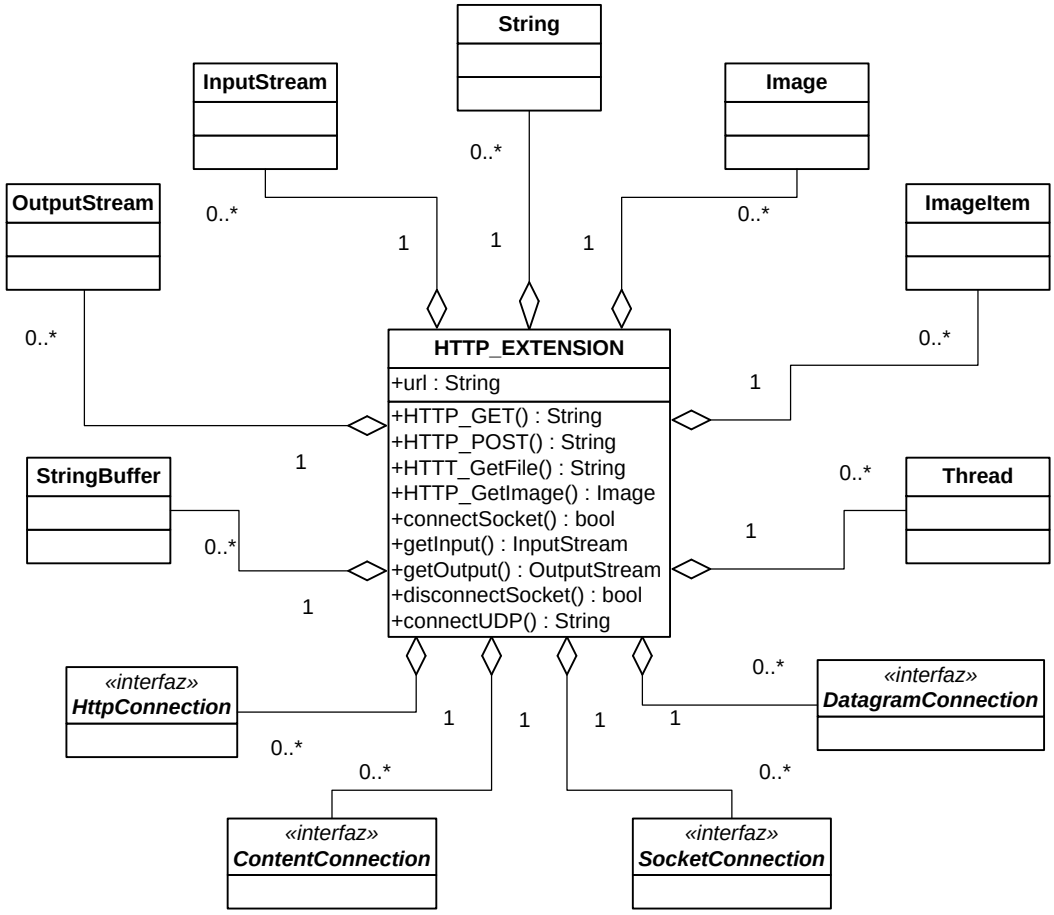
Fuente: Jenny Paola Montillo Gélvez

➢ **Diagrama de actividades.** En la figura 28 se puede apreciar el diagrama de actividades del tercer prototipo. Se agregaron las nuevas opciones que tiene el usuario.

➢ **Diagrama de clases.** En el diagrama de clases del tercer prototipo se agregaron cinco nuevos métodos; el método connectSocket(), getInput(), getOutput(), disconnectSocket() y el método connectUDP().

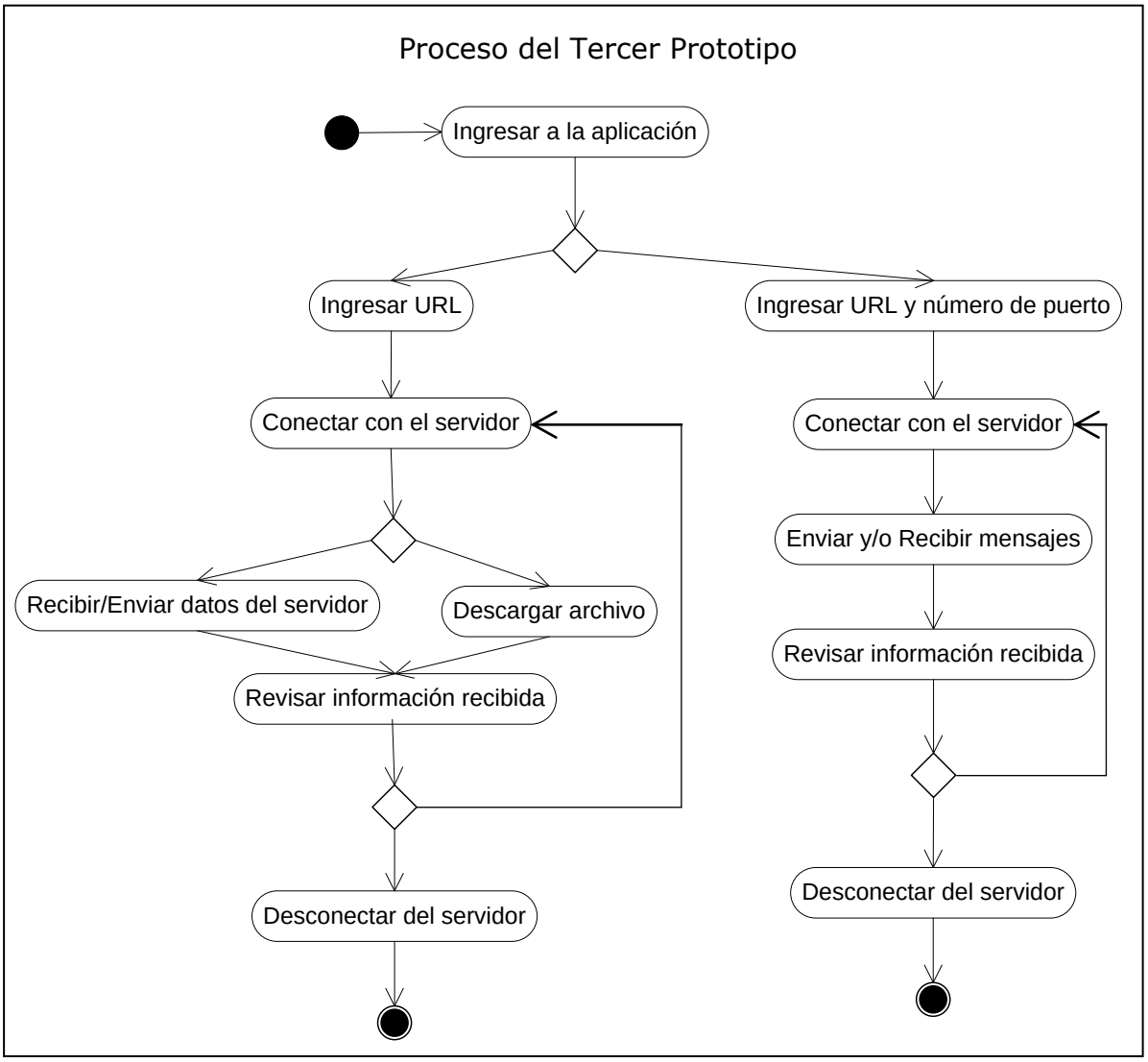
En la figura 29 se puede apreciar el diagrama de clases del tercer prototipo de la clase HTTP_EXTENSION y, en el cuadro 8 se encuentra su descripción.

Figura 29. Diagrama de clases del tercer prototipo – Conexión TCP/UDP



Fuente: Jenny Paola Montillo Gélvez

Figura 28. Diagrama de actividades del tercer prototipo – Conexión TCP/UDP



Fuente: Jenny Paola Montillo Gélvez

3.3.3 Desarrollo del tercer prototipo. Para el tercer prototipo se implementaron los métodos connectSocket, getInput, getOutput y disconnectSocket, que sirven para establecer una comunicación entre un celular y un servidor a través de sockets; y el método conncectUDP que establece una conexión entre un celular y un servidor utilizando datagramas.

Una vez construido el tercer prototipo, se realizaron las respectivas pruebas para comprobar su funcionamiento, para ello se utilizaron programas servidores tanto TCP, como UDP, desarrollados en JME.

Cuadro 8. Descripción del tercer prototipo de la clase HTTP_EXTENSION – Conexión TCP/UDP

Clase: HTTP_EXTENSION
Descripción: Clase para establecer conexiones entre celulares y servidores Web a través del protocolo HTTP.
Módulo: Principal
Propiedades: Concreta
Atributos: url
HTTP_GET(String, int, String) Método para establecer una conexión entre un servidor y un celular, a través del método estándar GET.
HTTP_POST(String, int, String) Método para establecer una conexión entre un servidor y un celular, a través del método estándar POST.
HTTP_GetFile(String, int, String, String) Método para establecer conexión con un servidor Web y visualizar el contenido de un archivo.
HTTP_GetImage(String) Método para establecer conexión con un servidor Web y visualizar una imagen determinada.
connectSocket(Boolean) Método para establecer conexión con un servidor utilizando sockets.
getInput() Método que permite el ingreso de una cadena de datos.
getOutput() Método que permite la salida de una cadena de datos.
disconnectSocket() Método que se utiliza para realizar la desconexión del programa servidor.
connectUDP(String) Método para establecer conexión con un servidor utilizando datagramas.

Fuente: Jenny Paola Montillo Gélvez

En la figura 30 se presenta parte del código del tercer prototipo.

En las figuras del anexo A, se puede apreciar una secuencia gráfica de los resultados obtenidos con el desarrollo del tercer prototipo.

Figura 30. Código del método connectSocket

```
/**
 * Método que permite establecer una conexión a través de sockets
 *
 * @param urlTCP String que contiene el host o dirección IP donde corre el programa servidor
 * @return Booleano true
 * @throws IOException Se genera al producirse un fallo o interupción en operaciones de I/O (Entrada/Salida)
 */
public boolean connectSocket(String urlTCP) throws IOException {

    sc = (SocketConnection)Connector.open(urlTCP);
    is = sc.openInputStream();
    os = sc.openOutputStream();

    return true;
}

/**
 * Método que almacena la información recibida a través de connectSocket
 * @return Objeto tipo InputStream
 */
public InputStream getInput (){
    return is;
}

/**
 * Método que almacena la información enviada a través de connectSocket
 * @return Objeto tipo OutputStream
 */
public OutputStream getOutput (){
    return os;
}
```

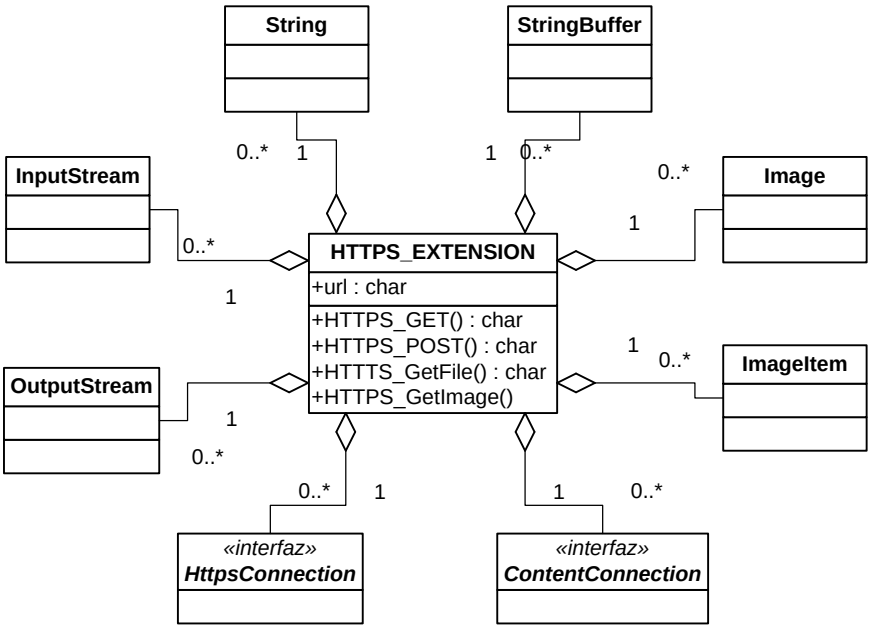
Fuente: Jenny Paola Montillo Gélvez

3.4 CUARTO PROTOTIPO – CONEXIÓN HTTPS

3.4.1 Revisión de requerimientos y funcionalidades. El cuarto prototipo consiste en el diseño y desarrollo de la clase HTTPS_EXTENSION, que como su nombre lo indica, consiste en implementar las mismas funciones que la clase HTTP_EXTENSION, pero usando el protocolo HTTPS.

3.4.2 Diseño UML del cuarto prototipo. Para el diseño del cuarto prototipo no se vio la necesidad de crear nuevos casos de uso ni diagramas de secuencia, debido a que en esta etapa se implementaron las mismas funciones que en el prototipo anterior, pero para el protocolo HTTPS. En la figura 31 se presenta el diagrama de la clase HTTPS_EXTENSION y en el cuadro 9 se presenta la descripción de dicha clase.

Figura 31. Diagrama de la clase HTTPS_EXTENSION



Fuente: Jenny Paola Montillo Gélvez

Cuadro 9. Descripción del cuarto prototipo. Clase HTTPS_EXTENSION

Clase: HTTPS_EXTENSION
Descripción: Clase para establecer conexiones entre celulares y servidores Web a través del protocolo HTTPS.
Módulo: Principal
Propiedades: Concreta
Atributos: url
HTTPS_GET(String, int, String) Método para establecer una conexión entre un servidor y un celular, a través del método estándar GET.
HTTPS_POST(String, int, String) Método para establecer una conexión entre un servidor y un celular, a través del método estándar POST.
HTTPS_GetFile(String, int, String, String) Método para establecer conexión con un servidor Web y visualizar el contenido de un archivo.
HTTPS_GetImage(String) Método para establecer conexión con un servidor Web y visualizar una imagen determinada.

Fuente: Jenny Paola Montillo Gélvez

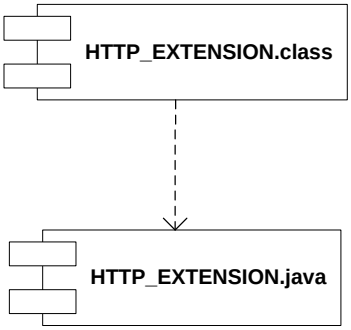
➢ **Modelos de despliegue.** También son conocidos como diagramas de componentes y representan la parte física de un sistema:

- Muestra los enlaces de comunicación física entre elementos hardware (máquinas y otros recursos, tales como impresoras).

- También muestra las relaciones entre máquinas físicas y procesos, es decir, qué se ejecuta y dónde se ejecuta.

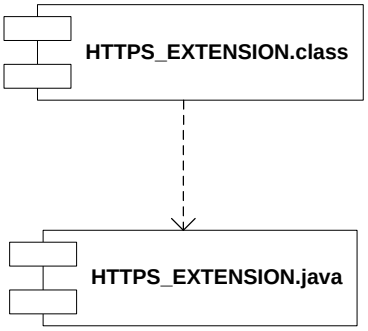
En las figuras 32 y 33 se pueden apreciar los diagramas de despliegue del prototipo.

Figura 32. Modelo de despliegue de la clase HTTP_EXTENSION



Fuente: Jenny Paola Montillo Gélvez

Figura 33. Modelo de despliegue de la clase HTTPS_EXTENSION



Fuente: Jenny Paola Montillo Gélvez

3.4.3 Desarrollo del cuarto prototipo. En este prototipo se implementó la clase HTTPS_EXTENSION que contiene los métodos HTTPS_GET, HTTPS_POST, HTTPS_GetFile y HTTPS_GetImage.

En la figura 34 se presenta una parte del código del cuarto prototipo.

Figura 34. Código del cuarto prototipo

```
/**
 * Constructor de la clase
 */
public HTTPS_EXTENSION() {

}


public String HTTPS_GET(String urls, int tamano, String tipoServidor) throws IOException {

    HttpsConnection connection2 = null;
    InputStream iss = null;
    OutputStream oss = null;
    StringBuffer stringBuffer = new StringBuffer();
    String cadenas2;
    cadenas2 = new String();

    if ((tipoServidor.compareTo("PHP") == 0) || (tipoServidor.compareTo("JSP") == 0)) {

        try {
                connection2 = (HttpsConnection)Connector.open(urls);
                connection2.setRequestMethod(HttpsConnection.GET);
                connection2.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
                connection2.setRequestProperty("METH_CONNECT", "CONNECT");
                connection2.setRequestProperty("IF-Modified-Since","20 Jan 2001 16:19:14 GMT");
                connection2.setRequestProperty("User-Agent","Profile/MIDP-2.0 Confirguration/CLDC-1.1");
                connection2.setRequestProperty("User-Agent","App de Prueba");
                connection2.setRequestProperty("Content-Language", "en-CA");
```

Fuente: Jenny Paola Montillo Gélvez

Completar y entregar el prototipo. En esta fase se realizan pruebas completas del funcionamiento del sistema con diferentes usuarios, para descubrir qué hace falta al prototipo, para complementarlo y hacer la entrega del producto final.

3.5 DESARROLLO APLICATIVO DE LAS PRUEBAS

En esta fase se realizaron pruebas de verificación y de validación al prototipo. Cabe mencionar que la etapa de pruebas se llevó a cabo durante todo el proceso de desarrollo del prototipo.

3.5.1 Pruebas de verificación. El objetivo de las pruebas de verificación es buscar discrepancias entre los requerimientos y la ejecución del software[19]. Estas pruebas fueron realizadas por la desarrolladora durante el proceso de codificación a medida que se iba incrementando el prototipo. La fuente de todos los cuadros que describen el desarrollo de las pruebas es personal.

Inicialmente las pruebas se realizaron en un computador de escritorio que cuenta con las siguientes características:

- **Hardware**

- Procesador Pentium 4 HT de 3.20 GHz.
- Memoria RAM de 3,25 GB.
- Disco duro de 465,6 GB.

- **Software**

- Sistema Operativo Windows XP.
- Máquina Virtual de Java JDK 1.7.0_21.
- NetBeans IDE 7.4.
- Wamp Server 2.1.
- Apache Tomcat/5.5.25.
- Internet Information Server 5.1.

Luego se realizaron pruebas en los siguientes dispositivos móviles:

- Teléfono celular LG KP500.
- Teléfono celular 9700 MTK 6235.
- Teléfono celular Nokia Asha 311.

Para llevar a cabo estas pruebas, se deben trasladar al dispositivo móvil los archivos .jar y .jad del aplicativo desarrollado, a través de cable USB. Y una vez instalado, se puede empezar a usar.

➢ **Procedimiento de la prueba.** El procedimiento que se llevó a cabo para la realización de las pruebas en los métodos de la clase HTTP_EXTENSION, se describe en el cuadro 10.

Es importante tener en cuenta que antes de establecer la conexión se deben iniciar los servicios del servidor con el que se vaya a trabajar.

[19] http://www.ldc.usb.ve/~teruel/ci4713/clases2001/testReqs.html

Cuadro 10. Procedimiento de pruebas para la clase HTTP_EXTENSION

Paso	Descripción
1	*Conexión con el método HTTP_GET:* en la aplicación que se va a ejecutar en el celular, se debe instanciar la clase HTTP_EXTENSION y elegir el método HTTP_GET. Luego se debe escribir la URL del servidor con el que se quiere establecer la conexión, luego se selecciona la opción "Conectar" y se espera la respuesta del servidor.
2	*Conexión con el método HTTP_POST:* en la aplicación que se va a ejecutar en el celular, se debe instanciar la clase HTTP_EXTENSION y elegir el método HTTP_POST. Luego se debe escribir la URL del servidor con el que se quiere establecer la conexión, luego se selecciona la opción "Conectar" y se espera la respuesta del servidor.
3	*Descargar archivo con el método HTTP_GetFile:* en la aplicación que se va a ejecutar en el celular, se debe instanciar la clase HTTP_EXTENSION y elegir el método HTTP_GetFile. Luego se debe escribir la URL del servidor con el que se quiere establecer la conexión, luego se selecciona la opción "Conectar" y se espera la respuesta del servidor.
4	*Visualizar imagen con el método HTTP_GetImage:* en la aplicación que se va a ejecutar en el celular, se debe instanciar la clase HTTP_EXTENSION y elegir el método HTTP_GetImage. Luego se debe escribir la URL del servidor con el que se quiere establecer la conexión, luego se selecciona la opción "Conectar" y se espera la respuesta del servidor.
5	*connectSocket:* Primero se debe iniciar un programa que sirva como servidor para la comunicación con sockets. En la aplicación que se va a ejecutar en el celular se debe escribir un host y un número de puerto por el cual se va a establecer la comunicación, se espera a que el servidor confirme la conexión y luego se procede a ingresar el texto que se va a enviar al servidor, nuevamente se espera la respuesta del servidor y así sucesivamente hasta que el socket se desconecte del servidor.
6	*connectUDP:* Primero se debe iniciar un programa que sirva como servidor para la comunicación con datagramas. En la aplicación que se va a ejecutar en el celular se debe escribir un host y un número de puerto por el cual se va a establecer la comunicación, se espera a que el servidor confirme la conexión y luego se procede a ingresar el paquete de datos que se va a enviar al servidor, nuevamente se espera la respuesta del servidor y así sucesivamente hasta que seleccione la opción de desconectarse del servidor.

Fuente: Jenny Paola Montillo Gélvez

El mismo procedimiento se llevó a cabo con los tres tipos de servidores con los que se hicieron las pruebas para el caso de los métodos HTTP_GET, HTTP_POST, HTTP_GetFile y HTTP_GetImage.

- **Resultados de las pruebas.** El cuadro 11 describe los resultados de las pruebas realizadas con la clase HTTP_EXTENSION.

La comunicación a través de datagramas no se pudo implementar adecuadamente dentro de la clase HTTP_EXTENSION, debido a la jerarquía de interfaces propia del Generic Connection Framework (GCF). Como se puede apreciar en la figura 3, la interfaz *DatagramConnection* hereda directamente de la interfaz *Connection*; por esta razón los datagramas no soportan el tratamiento de los datos a través de un stream de bytes, por medio de los métodos propios de la interfaz *StreamConnection* que son los adecuados para el manejo de la información. Por lo que su implementación sólo se puede llevar a cabo a nivel del perfil [17].

En otras palabras, los datagramas no soportan los métodos *OpenInputStream, OpenDataInputStream, OpenOutputStream* y *OpenDataOutputStream*, que son los que se necesitan para manipular la información que es enviada y recibida. Sin embargo, la interface DatagramConnection propia de Java, posee unos sencillos métodos para establecer este tipo de conexiones y por lo tanto es muy fácil establecerlas.

Cuadro 11. Resultados de la prueba con la clase HTTP_EXTENSION

No.	Descripción	Respuesta	
		Si	No
1	¿Fue posible establecer una conexión con el método HTTP_GET?	X	
2	¿Fue posible establecer una conexión con el método HTTP_POST?	X	
3	¿Fue posible visualizar un archivo con el método HTTP_GetFile?	X	
4	¿Fue posible visualizar una imagen con el método HTTP_GetImage?	X	
5	¿Fue posible establecer una comunicación utilizando sockets?	X	
6	¿Fue posible establecer una comunicación utilizando datagramas?		X

Fuente: Jenny Paola Montillo Gélvez

➢ **Procedimiento de la prueba.** En el cuadro 12 se hace la descripción del procedimiento que se llevó a cabo para la realización de las pruebas con los métodos de la clase HTTPS_EXTENSION.

Cuadro 12. Procedimiento de pruebas para la clase HTTPS_EXTENSION

Paso	Descripción
1	*Conexión con el método HTTPS_GET:* en la aplicación que se va a ejecutar en el celular, se debe instanciar la clase HTTPS_EXTENSION y elegir el método HTTPS_GET. Luego se debe escribir la URL del servidor con el que se quiere establecer la conexión, luego se selecciona la opción "Conectar" y se espera la respuesta del servidor.
2	*Conexión con el método HTTSP_POST:* en la aplicación que se va a ejecutar en el celular, se debe instanciar la clase HTTPS_EXTENSION y elegir el método HTTPS_POST. Luego se debe escribir la URL del servidor con el que se quiere establecer la conexión, luego se selecciona la opción "Conectar" y se espera la respuesta del servidor.
3	*Descargar archivo con el método HTTPS_GetFile:* en la aplicación que se va a ejecutar en el celular, se debe instanciar la clase HTTPS_EXTENSION y elegir el método HTTPS_GetFile. Luego se debe escribir la URL del servidor con el que se quiere establecer la conexión, luego se selecciona la opción "Conectar" y se espera la respuesta del servidor.
4	*Visualizar imagen con el método HTTPS_GetImage:* en la aplicación que se va a ejecutar en el celular, se debe instanciar la clase HTTPS_EXTENSION y elegir el método HTTPS_GetImage. Luego se debe escribir la URL del servidor con el que se quiere establecer la conexión, luego se selecciona la opción "Conectar" y se espera la respuesta del servidor.

Fuente: Jenny Paola Montillo Gélvez

➢ **Resultados de la prueba.** En el cuadro 13 se describen los resultados de las pruebas realizadas con la clase HTTPS_EXTENSION.

La clase HTTPS_EXTENSION se desarrolló, pero no se pudo implementar, debido a que las conexiones entre servidores seguros (HTTPS) y dispositivos móviles celulares, tienen un alto grado de complejidad, no en el código Java en sí, sino porque este tipo de conexiones exige cumplir una serie de requisitos para lograrlo. Uno de ellos es que sólo es posible establecer la conexión, cuando el servidor cuenta con certificados emitidos por las empresas Verisign o Thawte, de lo contrario se recibe una excepción en tiempo de ejecución del

tipo *CertificateException*, esto hace que este tipo de conexiones se tornen prácticamente imposibles utilizando certificados autofirmados.

Otra razón es debido a que siempre hay un significativo proceso de cómputo comprometido en este tipo de conexiones, además los procesos de encriptación con llave privada, la generación de un par llave RSA[20], demandan un consumo alto de tiempo y energía, y estos algoritmos se llevan a cabo mediante funciones y detalles de bajo nivel [16].

Cuadro 13. Resultados de la prueba con la clase HTTPS_EXTENSION

No.	Descripción	Respuesta	
		Si	No
1	¿Fue posible establecer una conexión con el método HTTPS_GET?		X
2	¿Fue posible establecer una conexión con el método HTTPS_POST?		X
3	¿Fue posible visualizar un archivo con el método HTTPS_GetFile?		X
4	¿Fue posible visualizar una imagen con el método HTTPS_GetImage?		X

Fuente: Jenny Paola Montillo Gélvez

3.5.2 Pruebas de validación. Este tipo de prueba pretende lograr una revisión final por parte de la organización que solicitó el sistema, lo cual, a menudo, significa validación del sistema [12]. En esta fase se asume que el software ha cumplido la etapa de verificación y se prueba el código final de la aplicación, es decir, el programa completo ya que no existe un programa de prueba, se deben hacer varios procedimientos de prueba por cada requisito o caso de uso especificado.

Estas pruebas fueron realizadas durante todo el proceso de desarrollo de los prototipos.

[20] Algoritmo criptográfico de clave pública creado por Rivest, Shamir y Adleman para cifrar y firmar digitalmente.

4. SOFTWARE APLICATIVO

Uno de los objetivos específicos de este trabajo de investigación, es desarrollar un aplicativo software para las pruebas de conexión entre el servidor y el dispositivo móvil celular, utilizando JME, que verifique la interoperabilidad entre ellos. En este capítulo se presenta el aplicativo.

4.1 DESCRIPCIÓN

Este software aplicativo utiliza el prototipo desarrollado y pretende dar a conocer su funcionalidad. La aplicación consiste en una sencilla herramienta que se instala en un celular y sirve para hacer una consulta o registro en una base de datos que se encuentra en un servidor, para esto es necesario hacer previamente la conexión correspondiente. También permite agregar nueva información a la base de datos, modificar y eliminar registros así como también visualizar imágenes.

Esta aplicación es una herramienta para capturar imágenes de personas que cometen algún tipo de infracción. La persona encargada de aplicar la sanción, debe registrar la información de quien comete la falta. Los datos que se registran son el número de documento de identificación del infractor, fecha y hora en la que se comete la infracción, detalle, valor, permite registrar si la multa ya fue pagada y la imagen (ver descripción de los campos en el anexo A). Luego debe enviar esta información a través de una conexión con un servidor, para que los datos queden registrados en la base de datos de la entidad.

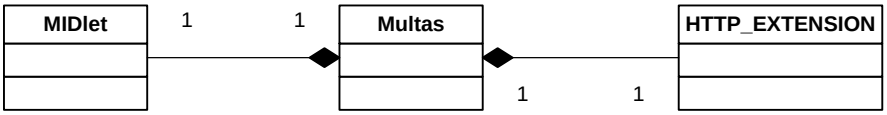
El aplicativo fue desarrollado con el IDE NetBeans 7.4 y utilizando el prototipo HTTP_EXTENSION, ambos elaborados en JME. La base de datos se creó con HeidiSQL 6.0 y para el servidor PHP se trabajó con WampServer 2.1 de manera local. Para la realización de las pruebas remotas se trabajó con el portal de administración phpMyAdmin, gracias a la colaboración del Equipo de Desarrollo CloudEISI. Cabe mencionar que todas estas herramientas son de uso libre.

Es preciso aclarar que esta herramienta se desarrolló con fines únicamente ilustrativos; es una simulación de casos de multas que pueden aplicar a cualquier tipo de caso ya sea ambiental, escolar o de tránsito y las entidades encargadas de desempeñar estas funciones no tienen ningún conocimiento de esta aplicación.

Todas las imágenes que se encuentran en este capítulo son de autoría personal.

En la figura 35 se puede apreciar el diagrama de clases del software aplicativo.

Figura 35. Diagrama de clases del software aplicativo

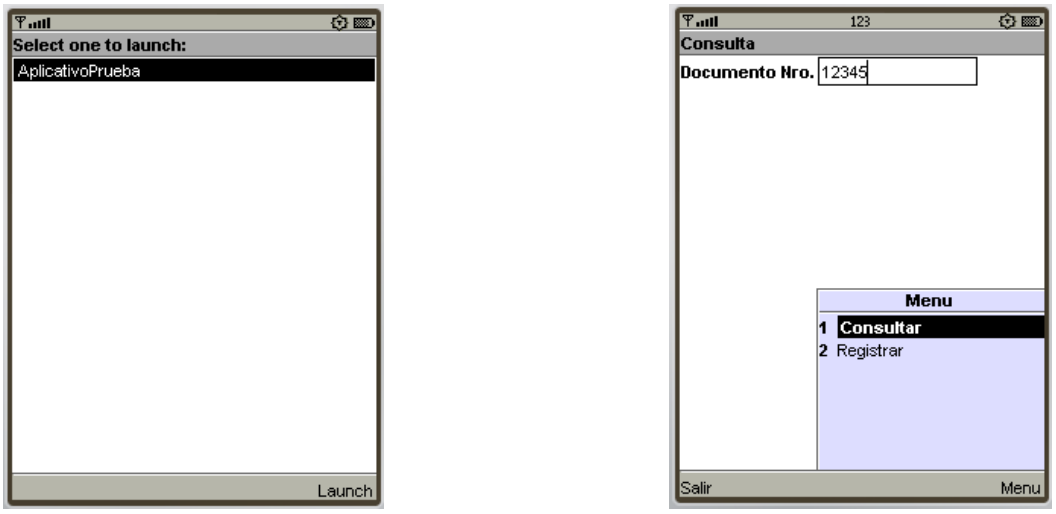


Fuente: Jenny Paola Montillo Gélvez

4.2 EJECUTANDO LA APLICACIÓN

Para empezar a manejar la aplicación, se debe seleccionar y abrir la aplicación oprimiendo la opción Launch, luego aparece el formulario que presenta las alternativas de Menú ó Salir que se encuentran en la parte inferior de la pantalla del celular; en la parte superior aparece una caja de texto para digitar un número de documento. El menú presenta dos alternativas, una es Consultar, que funciona con datos ya registrados; si el registro no existe, aparecerá el respectivo mensaje. La otra opción es Registrar, que permite diligenciar un nuevo formulario, como se puede apreciar en la figura 36.

Figura 36. Inicio del MIDlet AplicativoPrueba

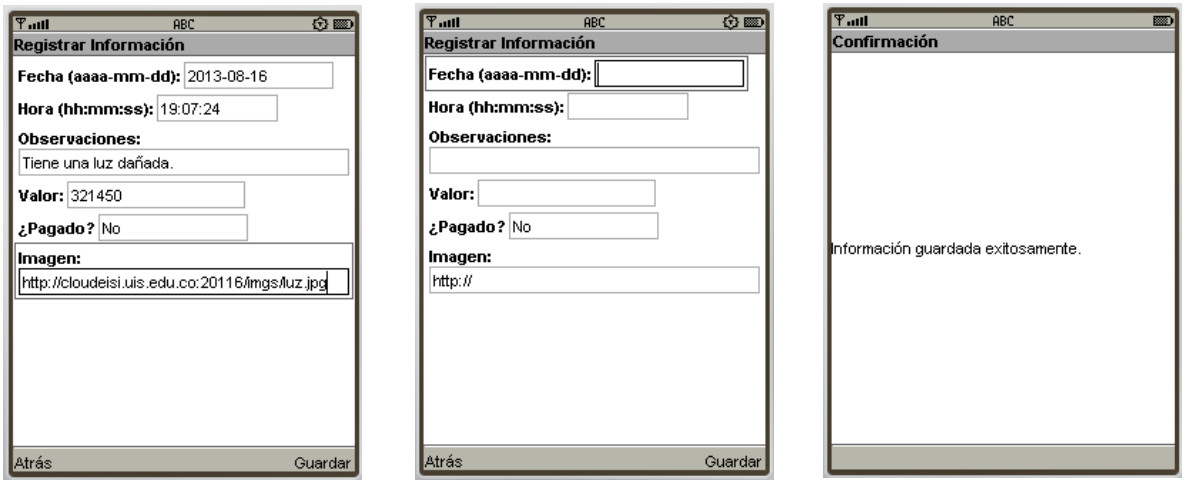


Fuente: Jenny Paola Montillo Gélvez

Al escoger la opción Registrar, se debe digitar la información de seis campos que tienen los siguientes nombres, Fecha, Hora, Observaciones, Valor, Pagado e Imagen. El campo Pagado indica si la multa ya fue pagada o no y en el campo Imagen se debe registrar la URL donde se encuentra guardada; sólo el

campo Imagen es opcional, de lo contrario saldrá un mensaje que solicita los datos completos. Luego se selecciona una de las opciones que aparecen en la parte inferior de la pantalla, Guardar la información o Volver a la pantalla anterior. Al seleccionar la opción Guardar aparece una confirmación de la operación efectuada. Esta secuencia aparece en la figura 37.

Figura 37. Opción Registrar del MIDlet AplicativoPrueba

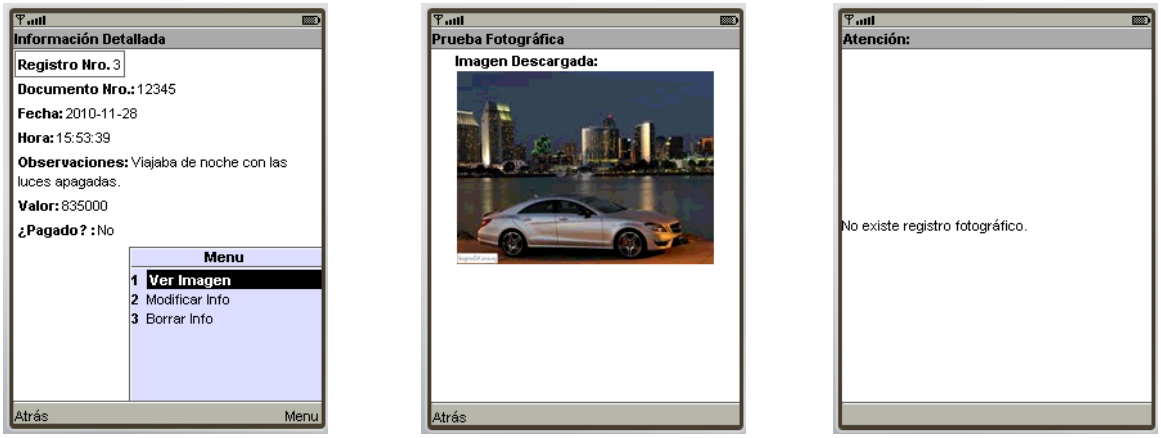


Al seleccionar la opción Consultar, si no existen registros correspondientes al número de identificación digitado, aparecerá un mensaje dando esta información; de lo contrario, aparece un listado con los registros encontrados, dando información básica como el número de registro y la fecha en la cual se realizó, como se muestra en la figura 38.

Figura 38. Resultados de la opción Consultar

Esta sección del MIDlet permite desplazarse por cada uno de los elementos de la lista y al resaltarse el que se quiere observar, se debe oprimir el botón Ver Detalle, el cual despliega el registro seleccionado y ofrece un menú que permite ver la imagen que evidencia la infracción cometida, de lo contrario se verá un mensaje que indica que no existe registro fotográfico, como se puede apreciar en la figura 39.

Figura 39. Resultados de la opción Ver Imagen



Las otras opciones de este menú son modificar la información registrada o borrar el registro que se está visualizando. Al guardar o modificar información si ésta está incompleta, saldrá un mensaje que advierte este hecho. Antes de borrar definitivamente el registro aparece un mensaje para confirmar la acción y una vez borrado, aparece el mensaje que afirma la operación, como se muestra en la figura 40.

Figura 40. Mensaje de las opciones Modificar y Borrar

5. CONCLUSIONES Y RECOMENDACIONES

Este capítulo ofrece las conclusiones a las que se llegó después de haber desarrollado el presente trabajo de investigación, también se exponen una serie de recomendaciones que permitirán mejorar los resultados de una posible segunda versión.

5.1 CONCLUSIONES

A nivel de la investigación se profundizó el aprendizaje del lenguaje Java, más específicamente en la rama JME, se ensayó la conectividad entre celulares y servidores Web, se llevó a cabo un proceso de aprendizaje de la plataforma NetBeans IDE tanto con el desarrollo del prototipo, como con la aplicación de prueba, esto permitió comprobar el gran potencial del lenguaje de programación Java, especialmente en lo relacionado con conectividad; también con las características que tiene este lenguaje referentes al uso de sockets y datagramas.

Con esta investigación se ha realizado un aporte significativo a nivel académico, no sólo en la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander, sino en las demás escuelas que tengan afinidad con este tema, para enriquecer y fortalecer la investigación en el área de desarrollo de aplicaciones en dispositivos móviles y más exactamente en lo que concierne a conectividad con servidores utilizando Internet, especialmente cuando se cuenta con los recursos materiales y con el potencial humano para continuar con el desarrollo de este tipo de herramientas.

La aplicación del modelo de proceso del software prototipado evolutivo apoyado en el UML, permitió conocer avances del prototipo y recibir la realimentación adecuada en cada una de las etapas del proyecto.

El prototipo posee una alta adaptabilidad ya que NetBeans permite cambiar las configuraciones CLDC entre la 1.0 y la 1.1, así como también los perfiles MIDP entre las versiones 1.0, 2.0 y 2.1. Otra ventaja del prototipo es que es muy fácil de usar para cualquier persona que posea conocimientos básicos de programación orientada a objetos.

Teniendo en cuenta los recursos limitados que manejan los celulares, el prototipo tiene un tamaño que no supera los 25 Kb. El tamaño de las aplicaciones depende de las características de la aplicación misma, sin

embargo, una aplicación como la que se desarrolló para el presente proyecto tiene aproximadamente 64 Kb. De esta manera se comprueba que en cuanto a capacidad, tanto el prototipo, como el software aplicativo pueden ser utilizados sin ningún tipo de inconvenientes en los dispositivos para los que fueron diseñados. Los tiempos de respuesta también son relativamente rápidos, especialmente si se tiene en cuenta que cada vez los celulares tienen mayor capacidad tanto de almacenamiento, como de procesamiento.

La implementación de este prototipo ayuda a reducir el tiempo de desarrollo de las aplicaciones, ya que el programador no tiene que detenerse en hacer consultas de tipo técnico que le proporcionen la información necesaria, para establecer adecuadamente una conexión.

5.2 RECOMENDACIONES

El estudio y las actividades realizadas durante esta investigación permiten hacer algunas recomendaciones para futuros trabajos como las siguientes:

Las páginas de los diferentes servidores con los que se trabajó contienen partes de código PHP, JSP ó ASP con HTML, al visualizar la información en el celular, éste no interpreta el código HTML, sólo el PHP, JSP ó el ASP, respectivamente. Esta herramienta podría enriquecerse significativamente, al crear un parser ó intérprete de HTML para su próxima versión, de manera que cualquier página pueda ser visualizada en su totalidad con los celulares.

Se podría pensar en la posibilidad de incluir esta herramienta en algún tipo de framework, para potencializar los resultados tanto de éste como del prototipo.

REFERENCIAS

[1] Apache Tomcat [On line].
Disponible en Internet : <http://tomcat.apache.org/index.html>

[2] Microsoft [En línea]. Disponible en Internet :
< http://www.microsoft.com/spain/technet/productos/iis/default.mspx >

[3] NetBeans.org [On line].
Disponible en Internet: <http://www.netbeans.org>,
<http://www.netbeans.org/index_es.html>

[4] MENÉNDEZ-BARZANALLANA ASENCIO, Rafael. Capítulo 9 Redes y comunicaciones. [En línea]. España. Universidad de Murcia. Fecha de actualización: Julio 19 de 2008.
Disponible en Internet: <http://www.um.es/docencia/barzana/II/Ii09.html>

[5] NationMaster.com [On line].
Disponible en Internet : < http://www.nationmaster.com/encyclopedia/Https>

[6] MONTILLO GÉLVEZ, Jenny Paola. Plan de Proyecto de Grado. Prototipo de componente software para el desarrollo de aplicaciones Wireless Internet en dispositivos móviles celulares utilizando J2ME. Universidad Industrial de Santander. Agosto de 2007.

[7] CASTRO MORA, Giovanny Andrés, LEAL GÓMEZ, José Luís. MOVILCRED – Aplicación en Computación Móvil para la Captura de Datos en Campo como Soporte al Catastro de Redes de Acueducto. Universidad Industrial de Santander. Colombia. 2004.

[8] Java Community Process – Community Development of Java Technology Specifications [On line].
Disponible en Internet: <http://jcp.org/en/home/index>

[9] MALLICK, Martyn. Mobile and Wireless Desing Essentials. Wiley Publishing, Inc. Indianapolis, United States of America.

[10] FROUFE QUINTAS, Agustín, JORGE CÁRDENES, Patricia. J2ME Java 2 Micro Edition. Manual de usuario y tutorial. Alfaomega Grupo Editor RA-MA, México D.F., 2004.

[11] SCHMULLER, Joseph. Aprendiendo UML en 24 horas. Pearson Educación, México D.F., 2000.

[12] WEITZENFELD Alfredo. Ingeniería de Software Orientada a Objetos con UML, Java e Internet. Thomson Editores, S.A. de C.V., México, 2005.

[13] MONTILVA C., Jonás A., ARAPÉ Nelson, COLMENARES Juan Andrés. Desarrollo de Software Basado en Componentes. Universidad de los Andes, Mérida, Venezuela. Universidad del Zulia, Maracaibo, Venezuela [On line]. Disponible en Internet: <http://webdelprofesor.ula.ve/ingenieria/jonas/Productos/Publicaciones/Congresos/CAC03%20Desarrollo%20de%20componentes.pdf>

[14] PRESSMAN, Roger. Ingeniería del Software – Un enfoque práctico, Sexta Edición. McGraw Hill, México, 2005.

[15] ARIZA ROJAS Maribel, MOLINA GARCÍA Juan Carlos. Introducción y Principios Básicos del Desarrollo de Software Basado en Componentes. Pontificia Universidad Javeriana. [On line]. Disponible en Internet: <http://pegasus.javeriana.edu.co/~jcpymes/Docs/DSBC.pdf>

[16] CARRERA ERAZO Enrique. El costo de la seguridad en dispositivos móviles. Revista Eídos 3, Tercera Edición, Diciembre 2013. Universidad Tecnológica Equinoccial, Quito, Ecuador.

[17] GÁLVEZ ROJAS, Sergio, ORTEGA DÍAZ, Lucas. Java a tope: J2ME (Java 2 Micro Edition). Edición Electrónica. Universidad de Málaga, España, 2003. Disponible en Internet: http://www.lcc.uma.es/~galvez/ftp/libros/J2ME.pdf

BIBLIOGRAFÍA

ARIZA ROJAS Maribel, MOLINA GARCÍA Juan Carlos. Introducción y Principios Básicos del Desarrollo de Software Basado en Componentes. Pontificia Universidad Javeriana. 2004. Disponible en Internet: http://pegasus.javeriana.edu.co/~jcpymes/Docs/DSBC.pdf

BOBADILLA S., Jesús, SANCHO H., Adela. Comunicaciones y bases de datos con JAVA a través de ejemplos. Alfaomega Grupo Editor, México, 2003.

CASTRO MORA, Giovanny Andrés, LEAL GÓMEZ, José Luis. MOVILCRED – Aplicación en Computación Móvil para la Captura de Datos en Campo como Soporte al Catastro de Redes de Acueducto. Bucaramanga, 2004. Trabajo de grado (Ingeniero de Sistemas). Universidad Industrial de Santander. Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática.

FOWLER, Martin y SCOTT, Kendall. UML gota a gota. Primera edición. Editorial Pearson, México, 1999.

FROUFE QUINTAS, Agustín, JORGE CÁRDENES, Patricia. J2ME Java 2 Micro Edition. Manual de usuario y tutorial. Alfaomega Grupo Editor RA-MA, México D.F., 2004.

GÁLVEZ ROJAS, Sergio, ORTEGA DÍAZ, Lucas. Java a tope: J2ME (Java 2 Micro Edition). Edición Electrónica. Universidad de Málaga, España, 2003. Disponible en Internet: http://www.lcc.uma.es/~galvez/ftp/libros/J2ME.pdf

GRECH M., Pablo. Introducción a la Ingeniería, Un enfoque a través del Diseño, Primera Edición. Pearson Education, Colombia, 2001.

INSTITUTO COLOMBIANO DE NORMAS TÉCNICAS Y CERTIFICACIÓN. Norma Técnica Colombiana 1486. Documentación. Presentación de Tesis, Trabajos de Grado y Otros Trabajos de Investigación. Bogotá D.C. Edición actualizada 2008.

MALLICK, Martyn. Mobile and Wireless Desing Essentials. Wiley Publishing, Inc. Indianapolis, United States of America.

MARTIN, Robert C. UML para programadores Java. Pearson Educación, S.A., Madrid, 2004.

McCONNELL, Steve. Desarrollo y gestión de proyectos informáticos. McGraw Hill, España, 1997.

MONTILVA C., Jonás A.; ARAPÉ, Nelson y COLMENARES, Juan Andrés. Desarrollo de Software Basado en Componentes. En: Congreso de Automatización y Control (4: noviembre: Mérida, Venezuela). Universidad de los Andes. Actas. 2003. Disponible en Internet: http://webdelprofesor.ula.ve/ingenieria/jonas/Productos/Publicaciones/Congresos/CAC03%20Desarrollo%20de%20componentes.pdf

PRESSMAN, Roger. Ingeniería del Software – Un enfoque práctico, Sexta Edición. McGraw Hill, México, 2005.

PRIETO MARTÍN, Manuel Jesús. Desarrollo de juegos con J2ME Java 2 Micro Edition, Primera Edición. Alfaomega Grupo Editor, México, 2005.

STEVENS P., POOLEY R. Utilización de UML en Ingeniería del Software con Objetos y Componentes. Pearson Educación, S.A., Madrid, 2002.

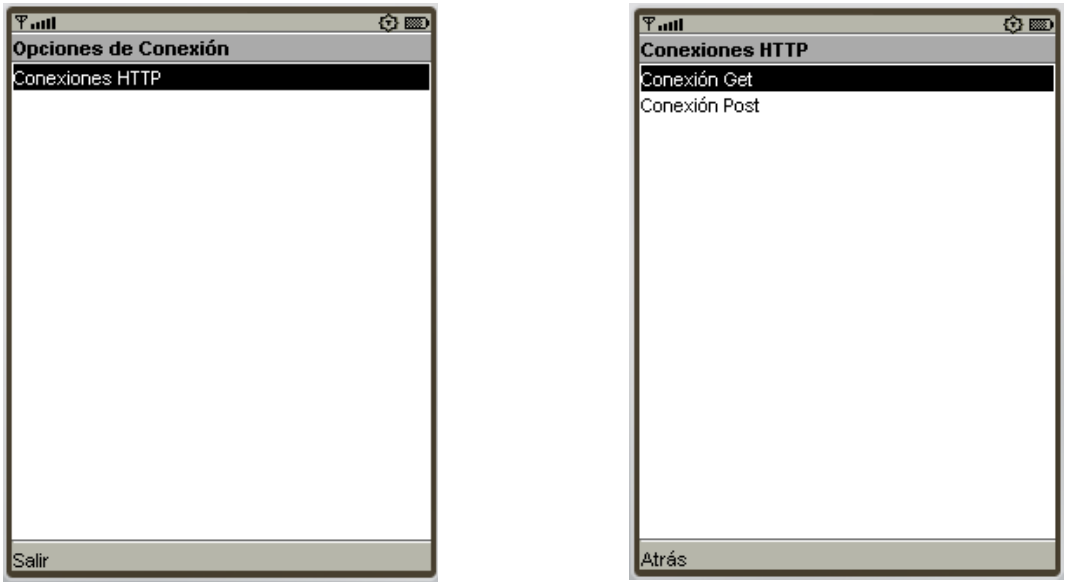
TOPLEY, Kim. J2ME in a Nutshell. O'Reilly, Sebastopol, CA, United States of America, Edition March 2002.

WEITZENFELD, Alfredo. Ingeniería de Software Orientada a Objetos con UML, Java e Internet. Thomson Editores, S.A. de C.V., México, 2005.

WU, Thomas C. Introducción a la programación orientada a objetos con Java, Segunda Edición. McGraw Hill, España, 2001.

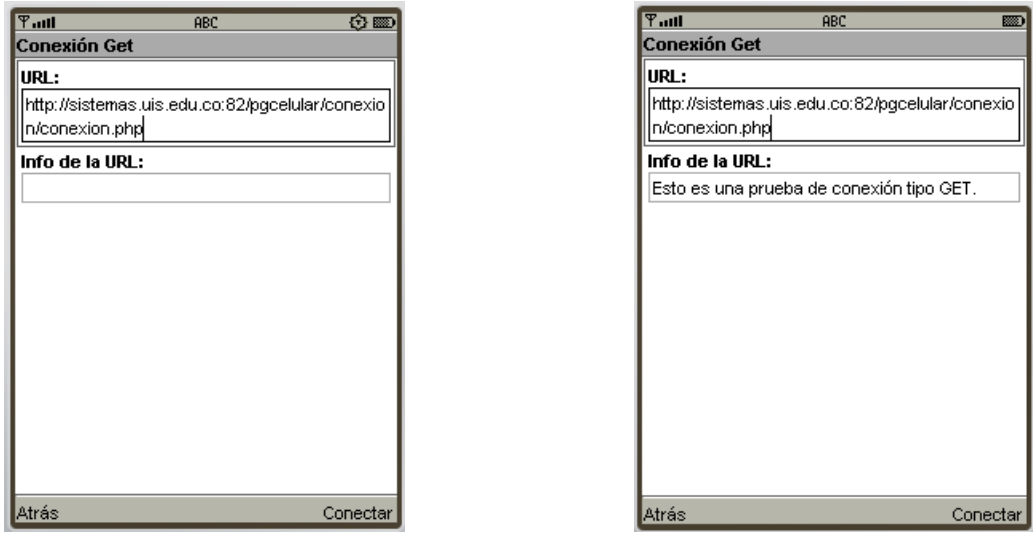
ANEXO A. RESULTADOS GRÁFICOS DEL PROTOTIPO

Figura A.1. Primera y segunda pantalla del primer prototipo



Fuente: Jenny Paola Montillo Gélvez

Figura A.2. Resultados al ejecutar la opción "Conexión Get"

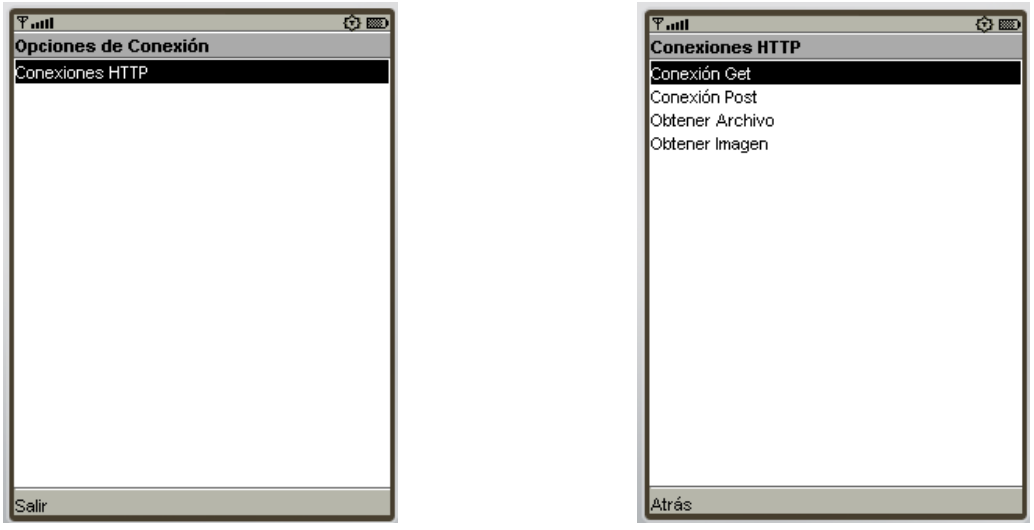


Fuente: Jenny Paola Montillo Gélvez

Figura A.3. Resultados al ejecutar la opción "Conexión Post"

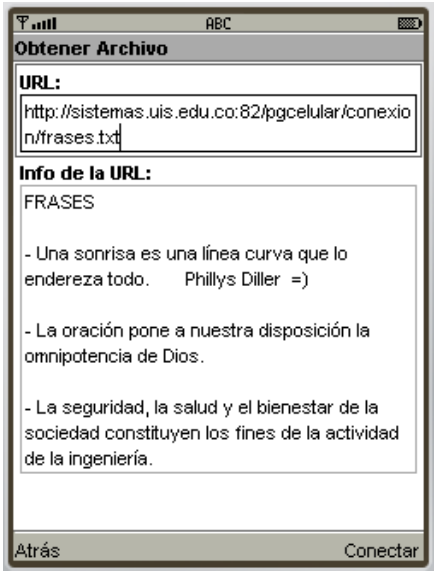
Fuente: Jenny Paola Montillo Gélvez

Figura A.4. Primera y segunda pantalla del segundo prototipo



Fuente: Jenny Paola Montillo Gélvez

Figura A.5. Resultados gráficos del segundo prototipo



La imagen que aparece con pantalla de fondo azul, corresponde al equipo que trabaja como servidor y el equipo con pantalla de fondo blanco es el que trabaja como cliente y en donde se ejecuta el tercer prototipo del aplicativo. El equipo servidor se encuentra a la espera de una conexión a través del puerto 5000; en el programa cliente se digita el host ó número IP y el puerto, que para este caso son 127.0.0.1 y 5000 respectivamente. Una vez se ha pulsado la opción "Conectar", aparece el mensaje de confirmación tanto en el equipo servidor, como en el cliente.

Figura A.6. Primer estado de la conexión TCP

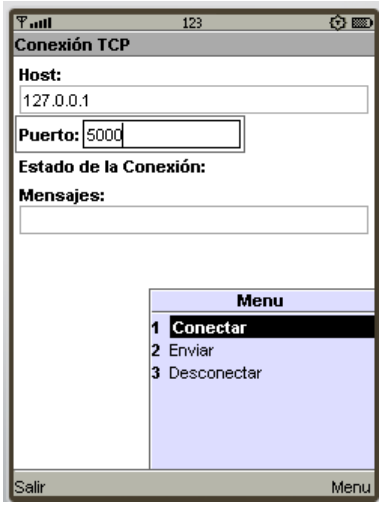
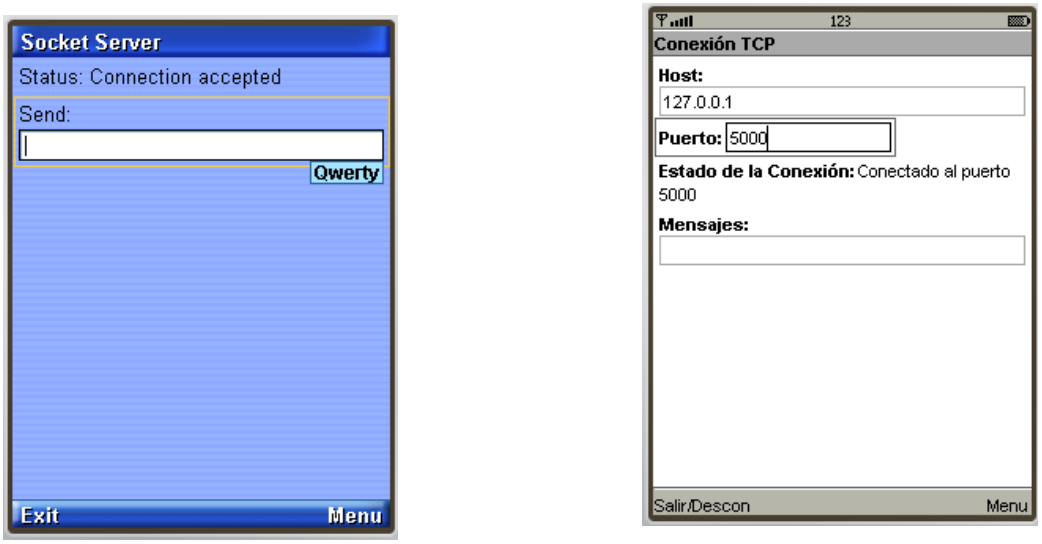


Figura A.7. Segundo estado de la conexión TCP



Fuente: Jenny Paola Montillo Gélvez

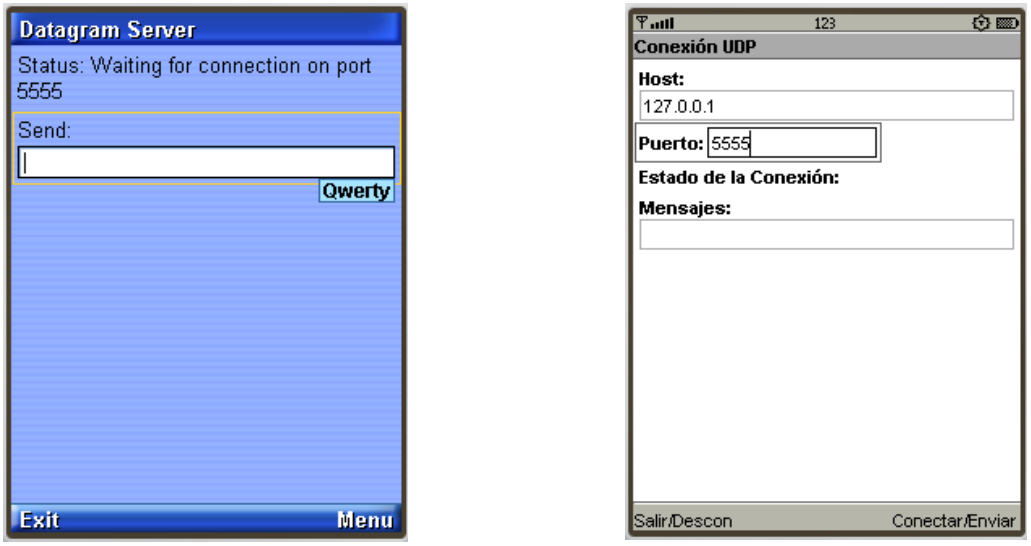
Una vez establecida la conexión entre el cliente y el servidor, se puede proceder con el envío y recepción de mensajes, hasta que el cliente decida desconectarse.

Figura A.8. Envío y recepción de mensajes vía TCP

Fuente: Jenny Paola Montillo Gélvez

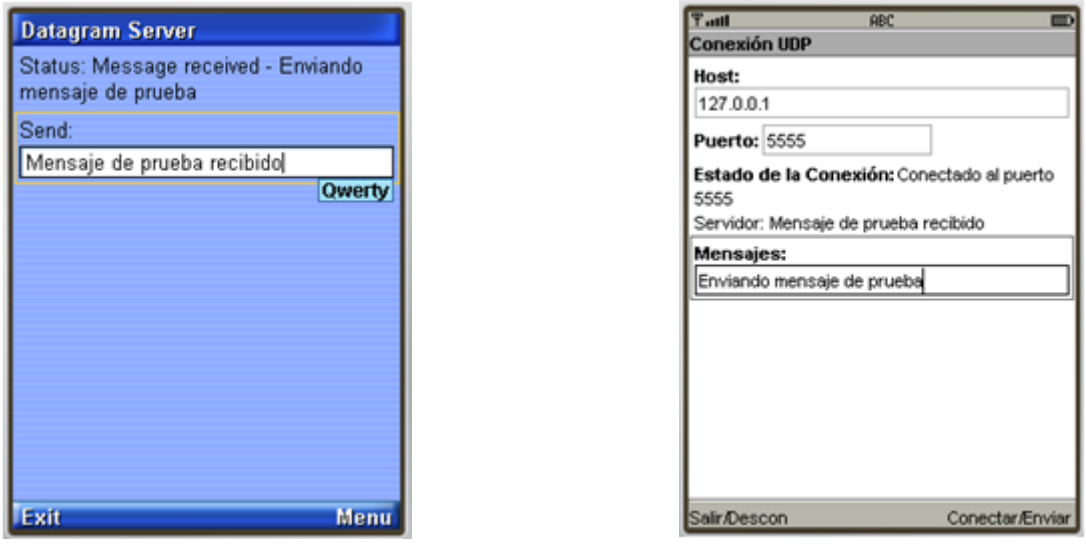
Conexión vía UDP a través del puerto 5555.

Figura A.9. Primer estado de la conexión UDP



Fuente: Jenny Paola Montillo Gélvez

Figura A.10. Envío y recepción de mensajes vía UDP



Fuente: Jenny Paola Montillo Gélvez

ANEXO B. DISEÑO DE LA BASE DE DATOS

A continuación se presenta el diseño de la base de datos que se utilizó en el aplicativo software.

B.1 Diccionario de datos del software aplicativo

- **Nombre Tabla:** tbl_registros
- **Descripción:** Almacena los datos necesarios para identificar la información de los registros.

tbl_registros			
Campo	**Tipo**	**Long**	**Descripción**
Id_Registro	INT	3	Identifica cada uno de los registros
Documento	INT	10	Identifica el número de documento
Fecha	DATE	10	Fecha en la que se realizó el registro
Hora	TIME	8	Hora en la que se realizó el registro
Detalle	VARCHAR	500	Describe el tipo de multa
Valor	INT	11	Valor de la multa
Pagado	VARCHAR	2	Indica si la multa ya fue pagada o no
Imagen	VARCHAR	255	Imagen de la falta cometida (Evidencia)

B.2 Modelo de datos del software aplicativo

tbl_registros		
PK	**Id_Registro**	**INT(3)**
	Documento	INT(10)
	Fecha	DATE(10)
	Hora	TIME(8)
	Detalle	VARCHAR(500)
	Valor	INT(11)
	Pagado	VARCHAR(2)
	Imagen	VARCHAR(255)