CONTROLPC

ACCESO A ARCHIVOS MULTIMEDIA Y EJECUCION DE COMANDOS EN UN PC POR MEDIO DEL CELULAR

JULIAN ANDRES FLOREZ GALEANO

ANDRES FELIPE QUINTERO ESCOBAR

UNIVERSIDAD TECNOLOGICA DE PEREIRA

FACULTAD DE INGENIERIAS

DEPARTAMENTO DE SISTEMAS

PEREIRA

2008

CONTROLPC

ACCESO A ARCHIVOS MULTIMEDIA Y EJECUCION DE COMANDOS EN UN PC POR MEDIO DEL CELULAR

JULIAN ANDRES FLOREZ GALEANO

ANDRES FELIPE QUINTERO ESCOBAR

Tesis de grado para optar al titulo de

Ingeniero de Sistemas y Computación

Director

ANGEL AUGUSTO AGUDELO ZAPATA

Ingeniero Electrico

UNIVERSIDAD TECNOLOGICA DE PEREIRA

FACULTAD DE INGENIERIAS

DEPARTAMENTO DE SISTEMAS

PEREIRA

2008

Nota de aceptación

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

A mi familia, inspiradora de grandes logros.

Julián

A mi madre que siempre motivó para estudiar y salir adelante.

Felipe

CONTENIDO

Pag.

DEFINICIÓN DEL PROBLEMA

En diversas situaciones, especialmente cuando estamos lejos de casa, nos damos cuenta de la urgente necesidad de acceder a los recursos de nuestro equipo de computo, para obtener algún archivo almacenado allí y exponerlo en una reunión importante, solucionar un problema ejecutando un comando del sistema operativo o descargar al PC las fotos, sonidos y videos cuando llenan la memoria de nuestro celular y limitan la captura de nuevo contenido multimedia.

JUSTIFICACIÓN

Actualmente, se ha masificado el uso del celular hasta el punto donde “siete de cada diez colombianos tiene un teléfono móvil”, los cuales dejaron de usar su teléfono sólo para hacer llamadas y ahora están aprovechando los nuevos servicios como las mejoradas cámaras fotográficas, reproducción de sonidos y video, uso de aplicaciones, navegación por Internet y descargas de canciones, imágenes y timbres. Las compañías fabricantes de celulares están mejorando y extendiendo la funcionalidad de sus modelos y de la mano de esta evolución ha estado java con su versión para móviles J2ME, quien por medio de diversas especificaciones denominadas JSR ha estandarizado el desarrollo de aplicaciones para acceder a los recursos de los nuevos teléfonos.

Algunos teléfonos móviles tienen implementados los paquetes definidos en la especificación JSR 172 para J2ME, el cual le permite a los teléfonos acceder a servicios Web basados en XML.

Se desarrollará este proyecto para conocer acerca de la especificación java (JSR 172) que permite a un móvil la conexión remota con un servidor para solucionar los problemas planteados. Así un usuario móvil tendría la facilidad de acceder a los recursos de dicho servidor (en este caso un computador personal) desde cualquier lugar.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar una solución que permita dar órdenes de consola desde un celular para ser ejecutadas en el PC y compartir archivos multimedia, utilizando servicios Web, aprovechando la implementación de la especificación JSR 172 de algunos celulares.

OBJETIVOS ESPECÍFICOS

- Estudiar y comprender el funcionamiento de la especificación JSR 172 la cual permite la creación de clientes de servicios Web para celulares.
- Realizar un resumen de dicha especificación que permita a los estudiantes tener una referencia para desarrollar proyectos similares.
- Implementar una aplicación cliente para celulares que tengan la JSR 172 implementada.
- Implementar una aplicación servidor para PCs con sistema operativo Windows, que preste a celulares el servicio Web:
    - Intercambio de archivos multimedia.
    - Ejecución remota de los siguientes comandos de consola: Xcopy, Attrib, Cd, Chkdsk, Del, Dir, Driverquery, Format, Getmac, Hostname, Ipconfig, Mkdir, Move, Ping, Rename, Shutdown, Systeminfo, Taskkill, Tasklist, Rd, Runas, Ver, Vol.
- Satisfacer las necesidades planteadas en la definición del problema

LISTA DE FIGURAS

Pag.

LISTA DE TABLAS

Pag.

INTRODUCCION

El éxito de internet ha hecho que las aplicaciones de escritorio tengan la tendencia a desaparecer, ahora empiezan a tener mayor aceptación las aplicaciones web porque están disponibles en cualquier lugar donde haya acceso a internet. El presente trabajo busca aprovechar la capacidad de los celulares para acceder a internet y la ventaja de la movilidad para proveer una solución que permita el acceso a la información de un computador personal desde cualquier lugar donde haya cobertura de las redes celulares.

1. MARCO TEORICO

1.1. JAVA

En el lenguaje de programación java, el código fuente es un archivo de texto plano con extensión .java, el cual es compilado usando el compilador javac generando un archivo .class. Un archivo .class no contiene código nativo al procesador; en su lugar contiene bytecodes, el lenguaje de la máquina virtual de java. La herramienta "java launcher tool" ejecuta la aplicación con una instancia de la máquina virtual de java.

La máquina virtual de java, está disponible para diferentes sistemas operativos, así un mismo archivo .class se puede ejecutar sobre Windows, Solaris, Linux o Mac OS.

Debido a nuestra experiencia en desarrollo de software hemos elegido el lenguaje de programación java, ya que nos ofrece excelentes herramientas para la codificación, pruebas y documentación de los proyectos realizados. Además posee una versión reducida para funcionar en dispositivos con escasos recursos, denominada J2ME, para la cual se ha realizado la especificación JSR 172 para acceder a servicios Web.

Las características más importantes de java son las siguientes:

- Simple y orientado a objetos

Para programar en java no es necesario de un extenso curso de programación, por su simplicidad le permite a los programadores entender los conceptos fundamentales rápidamente y ser productivos desde el principio.

Java está diseñado para ser orientado a objetos. Las necesidades de sistemas distribuidos cliente-servidor coinciden con el encapsulamiento y el paradigma de intercambio de mensajes del software basado en objetos. La tecnología Java provee una limpia y eficiente plataforma de desarrollo orientado a objetos.

- Robusto y Seguro

Java está diseñado para crear software altamente confiable. Provee dos niveles de comprobación un tiempo de compilación y otro tiempo de ejecución. Las características del lenguaje guían al programador hacia buenos hábitos de programación.

El modelo de administración de memoria es extremadamente simple: los objetos son creados con el operador new y la memoria se libera automáticamente gracias al garbage collector.

Java está diseñado para operar en entornos distribuidos y crear aplicaciones que no pueden ser atacadas desde afuera, como acceso de intrusos que utilizan código maléfico como virus o invasión al sistema de archivos.

- Arquitectura neutral y portable

Java está diseñado para soportar aplicaciones que serán desplegadas en entornos de red heterogéneos, donde las aplicaciones se deben ejecutar en variedad de hardware y sistemas operativos e interactuar con múltiples lenguajes de programación. El compilador de java genera bytecodes que pueden ser ejecutados en múltiples plataformas.

La arquitectura neutral es una parte de un sistema realmente portable. Sus programas son iguales sobre cada plataforma, porque no hay incompatibilidades de tipos de datos por diferencias entre arquitecturas hardware y software.

- Interpretado y multihilo

En una plataforma interpretada, tal como java, la fase de link de un programa es simple, incremental y liviana. Haciendo más rápidos los ciclos de desarrollo, prototipado y experimentación; contra el proceso pesado tradicional de compilación, link y ciclos de prueba.

La tecnología multihilos de java permite construir aplicaciones con múltiples hilos de actividad logrando un alto grado de interactividad con los usuarios finales. Las librerías de sistema de java han sido escritas para proveer la funcionalidad sin conflictos de concurrencia entre múltiples hilos de ejecución.

1.1.1. JAVA J2ME

J2ME es el acrónimo de Java 2 *Micro Edition*. J2ME es la versión de Java orientada a los dispositivos móviles. Debido a que los dispositivos móviles tienen una potencia de cálculo baja e interfaces de usuario pobres, es necesaria una versión específica de Java destinada a

estos dispositivos, ya que el resto de versiones de Java, J2SE o J2EE, no encajan dentro de este esquema. J2ME es por tanto, una versión "reducida" de J2SE.

J2SE (Java 2 Standard Edition) es la base de la tecnología Java. Permite el desarrollo de applets (aplicaciones que se ejecutan en un navegador web) y aplicaciones independientes (standalone). J2EE (Java 2 Enterprise Edition) está basado en J2SE, pero añade una serie de características necesarias en entornos empresariales, relativos a redes, acceso a datos y entrada/salida que requieren mayor capacidad de proceso, almacenamiento y memoria. La decisión de separarlos es debida a que no todas estas características son necesarias para el desarrollo de aplicaciones standard.

Diferente de J2SE, J2ME no es una pieza de software, no es una sola especificación. En lugar de eso, J2ME es una plataforma, una colección de tecnologías y especificaciones que son diseñadas para diferentes partes del mercado de dispositivos móviles.

J2ME, por lo tanto, está dividido en configuraciones, perfiles y paquetes opcionales. Las configuraciones son especificaciones que describen una máquina virtual y un conjunto base de APIs que pueden ser usadas con una cierta clase de dispositivos. Una configuración, por ejemplo, puede ser diseñada para dispositivos que tienen menos que 512 KB de memoria y conexión intermitente de red. El conjunto de APIs es habitualmente un subconjunto de las APIs de J2SE.

Un perfil se construye sobre una configuración, pero adiciona más APIs específicas para hacer un entorno completo para construcción de aplicaciones. Una configuración no especifica por sí misma con suficiente detalle la construcción de aplicaciones completas. Los perfiles usualmente incluyen APIs para el ciclo de vida de la aplicación, interface de usuario, y almacenamiento permanente.

Un paquete opcional provee funcionalidad que puede no ser asociada con una configuración específica o un perfil. Un ejemplo de un paquete opcional es el API de servicios web (JSR 172), el cual provee una API estándar para acceder a servicios web. Este paquete opcional podría ser implementado sobre cualquier combinación de configuraciones y perfiles.

1.2. SERVICIOS WEB

La promesa de Servicios Web es facilitar un ambiente distribuido en el cual cualquier número de aplicaciones, o componentes de aplicaciones, puedan interactuar de igual forma en un lenguaje y plataforma neutral. Esta interacción lleva heterogeneidad al mundo de la computación distribuida.

1.2.1. ¿QUÉ SON SERVICIOS WEB?

Un servicio Web es una pieza de la lógica de negocios, localizada en algún sitio en internet, que es accesible a través de protocolos estándares de internet como HTTP o SMTP.

Los servicios web se diferencian de tecnologías como J2EE, CORBA, tecnología win32 y scripts CGI por su estandarización. Esta nueva tecnología está basada en XML estandarizado (opuesto al estándar binario reservado) y es soportada globalmente por la mayoría de empresas tecnológicas. XML provee un lenguaje neutral para representar datos.

La visión de los servicios Web es que será posible crear aplicaciones complejas al instante o con el mínimo tiempo de desarrollo – combinando bits y fragmentos de datos y servicios que están distribuidos en la Web. El lema de Sun es "la red es la computadora" y esa visión se está haciendo realidad.

1.2.2. CARACTERÍSTICAS DE LOS SERVICIOS WEB

- Basado en XML

Gracias al uso de XML como capa de representación de datos para todos los protocolos y tecnologías, estas tecnologías pueden ser interoperables en su núcleo. Como el transporte de datos, XML elimina cualquier dependencia de red, sistema operativo o plataforma que el protocolo tenga.

- Acoplamiento Débil

Un consumidor de un servicio web no está relacionado con el servicio directamente; la interface del servicio web puede cambiar sobre el tiempo sin comprometer la habilidad del cliente de interactuar con el servicio. Un sistema fuertemente acoplado implica que la lógica del cliente y el servidor están estrechamente relacionados, implicando que si una interfaz cambia, la otra debe ser actualizada. Adoptando una arquitectura de acoplamiento débil tiende a hacer sistemas de software más manejables y permite una integración más sencilla entre diferentes sistemas.

- Grano-Grueso

Las tecnologías orientadas a objetos como Java exponen sus servicios a través de métodos individuales. Un método individual es también una operación para proveer alguna capacidad útil a nivel corporativo. Construyendo un programa Java desde el principio requiere la

creación de varios métodos de grano-fino que constituyen un servicio que es consumido por un cliente o por otro servicio. La tecnología de servicios Web provee una forma natural de definir servicios de grano-grueso que acceden a la cantidad apropiada de la lógica de negocio.

- Habilidad de ser síncrono o asíncrono

La sincronización se refiere a la dependencia del cliente a la ejecución del servicio. En operaciones síncronas, el cliente se bloquea y espera a que el servicio complete la operación antes de continuar. Las operaciones asíncronas permiten al cliente invocar un servicio y ejecutar otras funciones. Los clientes asíncronos recuperan el resultado después, mientras que los clientes síncronos reciben su resultado cuando el servicio ha sido completado. La capacidad de ser asíncrono es el factor clave para facilitar sistemas débilmente acoplados.

- Soporte de Llamadas a Procedimientos Remotos (RPC)

Los servicios Web permiten a los clientes invocar procedimientos, funciones y métodos de objetos remotos usando un protocolo basado en XML. Los procedimientos remotos exponen parámetros de entrada y salida que un servicio Web debe soportar. El desarrollo de componentes a través de Enterprise JavaBeans (EJBs) y componentes .NET se han convertido en parte de las arquitecturas y desarrollos empresariales de los 2 últimos años. Ambas tecnologías son distribuidas y son accesibles a través de una variedad de mecanismos RPC. Un servicio web soporta RPC proporcionando servicios de su propiedad, equivalente a un componente tradicional, o traduciendo invocaciones entrantes en una invocación de un componente EJB o .NET.

- Soporte de intercambio de documentos

Una de las ventajas clave de XML es la forma genérica de representar datos y documentos complejos. Estos documentos pueden ser sencillos como la representación de direcciones actuales o pueden ser complejos como la representación de un libro completo. Los servicios Web soportan el intercambio transparente de documentos para facilitar la integración del negocio.

1.2.3. PRINCIPALES TECNOLOGÍAS

Han surgido tres tecnologías principales como los estándares mundiales que constituyen el núcleo actual de la tecnología de servicios Web:

- Simple Object Access Protocol (SOAP)

El Protocolo Simple de Acceso a Objetos suministra un estándar de empaquetado para transporte de documentos XML sobre varias tecnologías estándar de internet, como SMTP, HTTP y FTP. También define los estándares de codificación y binding en XML para el transporte y codificación de llamados XML no RPC. SOAP provee una estructura simple para hacer RPC: intercambio de documentos. Teniendo un mecanismo estándar de transporte, clientes heterogéneos y servidores que repentinamente pueden volverse interoperables. Los clientes .NET pueden invocar EJBs expuestos a través de SOAP, clientes Java pueden invocar componentes .NET a través de SOAP. Ver anexo C.

- Web Service Description Language (WSDL)

El lenguaje de descripción de servicios web es una tecnología XML que describe la interfaz de un servicio web en una forma estándar. WSDL estandariza cómo un servicio web representa los parámetros de entrada y salida de una invocación externa, la estructura de la función, la naturaleza de la invocación (solo entrada, entrada/salida, etc.), y el protocolo binding del servicio. WSDL permite que distintos clientes puedan interactuar con el servicio web. Ver anexo D.

- Universal Description, Discovery, and Integration (UDDI)

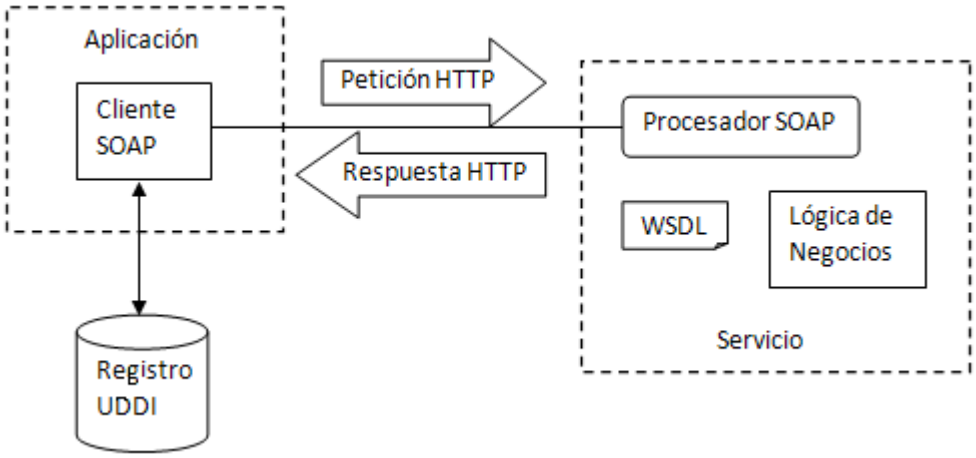
UDDI suministra un registro mundial de servicios web para propósitos de anuncio, descubrimiento e integración. Los analistas de negocios y tecnólogos usan UDDI para descubrir los servicios web disponibles buscando por nombres, identificadores, categorías o las especificaciones implementadas por el servicio web. UDDI provee una estructura para representar empresas, relaciones empresariales, servicios web, especificación de metadatos y puntos de acceso a servicios web.

Una de las grandes promesas de los servicios web es la integración perfecta y automática de negocios: Una pieza de software descubrirá, accederá, integrará e invocará dinámicamente nuevos servicios desde compañías desconocidas sin la necesidad de intervención humana. Una integración dinámica de esta naturaleza requiere la participación combinada de SOAP, WSDL y UDDI para proveer una infraestructura estándar que permita los negocios dinámicos del mañana. Combinadas, estas tecnologías están evolucionando porque son los primeros estándares que ofrecen la promesa de un negocio dinámico.

La relación entre estas partes (SOAP, WSDL, and UDDI) puede ser descrita como sigue: Una aplicación actuando en el rol de un cliente servicio web necesita localizar otra aplicación o una pieza de la lógica de negocios localizada en algún sitio en la red. El cliente consulta un registro UDDI por un servicio por nombre, categoría, identificador o especificación soportada. Una vez localizado, el cliente obtiene información acerca de la localización de un documento WSDL desde el registro UDDI. El documento WSDL contiene información acerca de cómo hacer contacto con el servicio web y el formato de los mensajes de llamado en XML Schema. El

cliente crea un mensaje SOAP de acuerdo con el XML Schema encontrado en el WSDL y envía una petición al host donde se encuentra el servicio, ver Figura 1.

Figura 1. Relación entre UDDI, SOAP Y WSDL



1.2.4. APLICACIONES PRÁCTICAS

Debido a la promesa de interoperabilidad multiplataforma de SOAP y servicios Web, pueden haber soluciones prácticas a problemas que, hasta ahora, han sido solo un sueño de los defensores de la computación distribuida.

Los servicios web pueden tener aplicaciones simples y discretas como el servicio de conversión de moneda de dólares a pesos o el servicio de traducción que convierte de inglés al español.

Este escenario se vuelve más interesante cuando compañías usan servicios web para automatizar y simplificar sus procesos de negocio. Algunos portales web B2C (Business to consumer) como la industria del turismo frecuentemente combinan las ofertas de múltiples productos y servicios de varias compañías y los presenta como una unidad al cliente del portal. Es difícil integrar los sistemas backend de cada negocio para proveer los anuncios del portal de forma confiable y rápida.

Una aerolínea puede ofrecer servicios adicionales como alquilar un auto de una compañía foránea sin salir de la página de la aerolínea, e incluso ofrecer descuentos por comprar paquetes de servicios.

En la industria de la salud, un doctor usando un computador de mano puede acceder a sus registros, historiales de sus pacientes y su farmacia preferida usando un servicio web. El doctor puede escribir una prescripción electrónica y enviarla directamente a su farmacia preferida por otro servicio web. Si todas las farmacias en el mundo estandarizaran la forma de aceptar prescripciones, el doctor podría escribir su prescripción para cualquier farmacia que el paciente seleccione. La farmacia debe cumplir la prescripción inmediatamente y tenerla lista cuando el paciente llegue a casa.

Esto podría extenderse, si las interfaces usadas entre doctores y farmacias están estandarizadas usando servicios web, un portal podría actuar como intermediario entre doctores y farmacias suministrando información direccionada a las peticiones que más se ajusten a las necesidades del paciente. Por ejemplo, un paciente puede registrarse con un intermediario y especificar que quiere usar medicina genérica en lugar de costosas marcas. Un intermediario puede recibir la petición del servicio web farmacéutico y adaptar la petición a la medicina genérica equivalente. El intermediario puede exponer servicios web a los doctores y farmacias y puede manejar asuntos como la seguridad, privacidad y no rechazo.

2. ESTADO DEL ARTE

A continuación se exponen algunas de las herramientas más conocidas en el mercado que solucionan en parte las necesidades que ControlPC satisface.

2.1. GOTOMYPC

Es una solución que utiliza un servidor externo en Internet para conectar a un usuario a su computador desde un navegador y permitirle manejarlo como si estuviera frente a él.

Provee los siguientes servicios:

- Compartir la pantalla: Lanzar un visor desde cualquier navegador para el acceso interactivo a cualquier aplicación de escritorio (incluso las que no están basadas en web).
- Transferencia de archivos: Permite enviar y recibir archivos y carpetas – incluyendo archivos compartidos en red LAN.
- Impresión remota: Imprimir desde su PC en una impresora donde usted esté.

Comparada con ControlPC cuenta además con un modulo de seguridad permitiendo comunicación segura. Encriptación y compresión de los datos a enviar, además está diseñado para ser utilizado desde un computador remoto o desde un PocketPC con sistema operativo Windows Mobile 2003 o superior. No puede ser utilizado desde celulares. No requiere instalación permanente de software cliente, además necesita un navegador con conexión a internet de 56 Kbps o mejor, permite el acceso a cualquier aplicación de escritorio (incluso las que no están basadas en web).

2.2. AVVENU

Avvenu permite acceder a un computador desde cualquier conexión de internet de un computador Windows, Macintosh, celular o SmartPhone. Para utilizarlo es necesario instalar el software Avvenu Connector sobre el PC que se quiere acceder remotamente. Luego crear una cuenta de usuario.

Provee los siguientes servicios:

Las transacciones de archivos se hacen por medio de una encriptación estándar usada también en las transacciones bancarias.

- Es posible compartir con otras personas la información de mi computador de forma total o parcial. Simplemente estableciendo lo correos de las personas que desean ver los archivos a compartir, ellos ingresarán a su cuenta de correo y desde allí podrán ver o descargar dichos archivos.

- Si el usuario se suscribe al servicio Anytime Files, podrá acceder a su computador incluso si este está apagado, pues dicha información quedará almacenada en el servidor de Avvenu, si está información es cambiada, ella misma se sincronizara en el momento en que el computador acceda de nuevo a internet.

Comparada con ControlPC esta herramienta cuenta con un modulo de seguridad muy potente, además permite conexión desde múltiples tipos de clientes, su principal desventaja es que es el servidor centralizado Avvenu quien controla el acceso a los datos de cada computador, por lo tanto él también puede acceder a ellos en cualquier momento, lo cual puede causar algo de desconfianza para cualquier persona.

2.3. LAPLINK

Es una solución que utiliza un servidor externo en Internet para conectar a un usuario a su computador desde un navegador y permitirle manejarlo como si estuviera frente a él.

Provee los siguientes servicios:

- Conexión segura al PC sin problemas de configuración o firewalls corporativos.

- Conectividad desde computadores con diversos sistemas operativos Windows, Linux, Mac

- Integración nativa del protocolo de Microsoft Remote Desktop Protocol para el control de escritorio Remoto sobre internet y sin firewalls.
- Descargar documentos remotos en una maquina local e imprimirlo.
- Hacer búsquedas en el PC usando la herramienta Google Desktop Search.

Esta solución presenta las mismas ventajas y desventajas con respecto a controlPC Avvenu.

3. DESARROLLO DEL PROYECTO

3.1. DESCRIPCIÓN DEL PROYECTO

ControlPC permite al usuario acceder remotamente a su computador desde un celular conectado a internet utilizando servicios Web, para compartir archivos y dar órdenes de consola, en un PC con sistema operativo Windows, aprovechando la implementación de la especificación JSR 172 de algunos celulares.

El sistema está compuesto por una aplicación cliente y una aplicación servidor:

1. La aplicación cliente se encuentra en un celular y le permite a un usuario realizar las siguientes tareas:

    - Registrar la información de varios PCs para permitirle al usuario acceder a los recursos de cada PC en cualquier momento.

      Un registro de un PC consiste de Nombre del PC, Dirección IP. El proceso de registrar un PC en un celular requiere tener la autorización del Superusuario por medio de su login y password, y se requiere también tener instalada la aplicación cliente en el celular. El objetivo de registrar un PC es permitirle a un usuario acceder a la aplicación cliente.

    - Transferir archivos y carpetas desde el Celular hasta el PC y viceversa.
    - Ejecución remota de comandos de consola desde el celular en el PC.
    - Obtener información del PC como la cantidad de memoria RAM instalada, memoria RAM disponible, información del procesador, versión del sistema operativo, etc.
    - Obtener información de red como el nombre del host, dirección IP, máscara de subred, puerta de enlace, número de puerto y dirección MAC.
    - Obtener información de los procesos, lo cual consta de nombre del proceso, cantidad de memoria usada y su respectivo estado. También permite matar procesos.

- Realizar búsquedas de archivos o carpetas en el PC
- Ejecutar operaciones sobre los archivos y carpetas como renombrar, eliminar, ver detalles y cambiar permisos de acceso. También permite crear carpetas. Los detalles son la ruta completa archivo o carpeta, el tamaño y los permisos de almacenamiento, sistema y solo lectura y oculto. Estas opciones están disponibles al navegar por el sistema de archivos del celular y el PC, y al hacer una búsqueda en el PC.
- Permite visualizar algunos tipos de archivos como son las extensiones txt, xml, bat, js, html, htm, java, php, sql. Tambien permite visualizar imágenes con la condición de que estas se encuentren almacenadas en el celular.
- Respecto a las operaciones del sistema de archivos el usuario puede navegar por los diferentes directorios, visualizar su contenido para seleccionar los archivos implicados en la operación seleccionada.
- Ver la ayuda de la aplicación

2. La aplicación servidor se encuentra en el PC y le ofrece al Superusuario las siguientes opciones:
    - Iniciar el servicio para recibir las solicitudes del usuario.
    - Suspender el servicio impidiendo al usuario acceder al PC.
    - Ver un historial de las operaciones que ha realizado un usuario remoto para hacer seguimiento al uso del servicio. Además de permitir la eliminación manual y periódica de los registros del historial.
    - Gestionar usuarios (eliminar y modificar cuentas de usuario). Una cuenta de usuario está compuesta por Login, Password e IMEI del celular.
    - Cambiar el password del Superusuario
    - Ver la ayuda de la aplicación

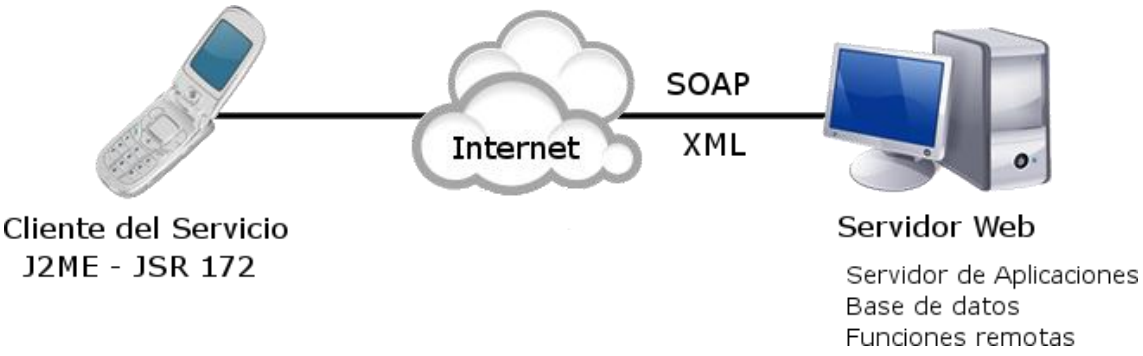
Para poder acceder al PC desde el celular, se deberá dejar el PC encendido, conectado a internet y ejecutando la aplicación servidor.

El tamaño máximo de los archivos que se pueden transferir está determinado por el tamaño máximo de archivo que se pueda almacenar en el celular.

3.2. ARQUITECTURA

La arquitectura empleada es cliente servidor de dos capas donde se separa el procesamiento en dos máquinas, en el cliente (el celular) se procesa la interfaz de usuario y se realizan las peticiones al servidor. El servidor es el proveedor del servicio, almacena la información del catálogo del servicio y de usuarios en la base de datos.

Figura 2. Arquitectura del sistema



La Figura 2 muestra los tres elementos más importantes de la arquitectura:

- Una aplicación que reside en un dispositivo móvil desarrollada usando el api de servicios web. La aplicación incluye un stub que usa el runtime de la JSR 172 para comunicarse con la red.

  El runtime oculta las complejidades como la administración de la conexión y la codificación de los datos. Para independizar los stubs del runtime se utiliza la Service Provider Interface (SPI), permitiendo portabilidad de stubs entre las implementaciones de diferentes vendedores.

- La red inalámbrica, internet, y la comunicación correspondiente y los protocolos de codificación, incluyendo los protocolos binarios, HTTP, y SOAP/XML.

- Un servidor web, actuando como el productor del servicio, típicamente está detrás de uno o más firewalls y un proxy. El servidor web frecuentemente provee acceso a aplicaciones back-end y servidores sobre una red privada.

La primera versión del API de servicios web está dirigida solo al consumo de servicios web. No soporta la creación y el despliegue de servicios; un dispositivo móvil puede ser un consumidor de un servicio, pero no un productor. JSR 172 no especifica un API para el descubrimiento de servicios web usando UDDI.

3.3. ACERCA DE LA INGENIERÍA DE SOFTWARE

La ingeniería de software describe de forma detallada la interacción de cada uno de los actores del proyecto con el sistema, y el proceso interno que realiza el sistema para cumplir los requerimientos planteados en la descripción del proyecto.

Para su realización se utilizaron solo los diagramas UML necesarios para aclarar que hacer y cómo hacerlo y así distribuir el trabajo de implementación entre los autores del proyecto.

El tiempo estimado del proceso de dicha ingeniería fue de aproximadamente 4 meses con sesiones de 4 horas diarias. Ver el Anexo A. Ingeniería de Software

4. COMO REALIZAR UN SERVICIO WEB EN JAVA: EJEMPLO PRÁCTICO

A continuación se describirá detalladamente los pasos a seguir para crear un servicio web utilizando el lenguaje de programación Java por medio del IDE de desarrollo Netbeans 6.1 el cual es a la fecha la versión más actualizada de dicho programa.

Lo primero que debemos hacer es tener los instaladores adecuados, en este caso usaremos los siguientes paquetes de instalación.

- Netbeans 6.1 182 Mb: el cual se descarga en la página web de netbeans.org[1].
- JDK 6.0: El cual es un completo entorno para desarrollo para construir aplicaciones, applets, y componentes usando el lenguaje de programación Java. Incluye herramientas y utilidades que le ayudarán a desarrollar, ejecutar, depurar y documentar los programas escritos en Java. La parte del JDK que permite correr aplicaciones escritas en Java es el Java Runtime Environment JRE[2].

La instalación del JDK es algo muy intuitivo, este requiere que se tenga un mínimo de espacio en el disco duro de 300 Mb, la instalación solo toma de uno a dos minutos.

Luego instale Netbeans, el cual instala automáticamente el servidor de aplicaciones Glashfish donde se montará el servicio web, durante el progreso de la instalación se le solicitará al usuario el nombre de usuario y la clave, la cual por defecto serán:

- Admin UserAdmin: admin
- Admin Password: adminadmin

Esta instalación requiere que se tenga 512Mb de espacio en el disco duro y toma aproximadamente de cuatro a cinco minutos.

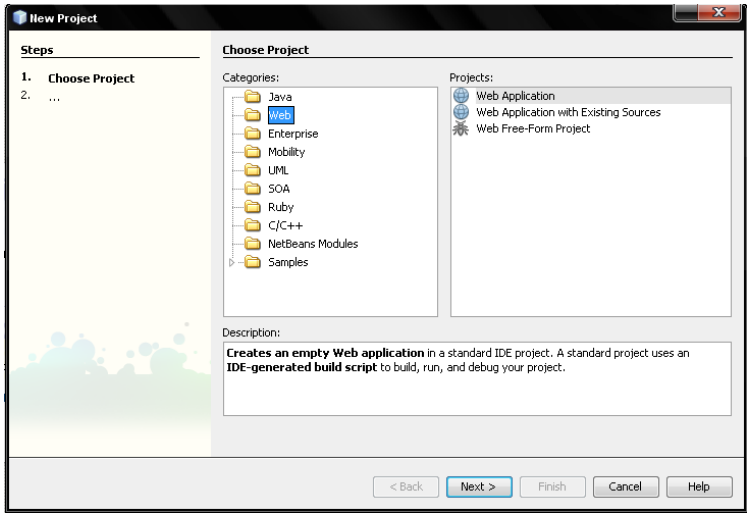
[1] http://bits.netbeans.org/netbeans/6.1/m1/latest/

[2] http://java.sun.com/javase/downloads/index.jsp

4.1. CREACIÓN DEL SERVICIO WEB

Después del proceso de instalación se debe ejecutar Netbeans. Luego vaya al menú y elija File -> New Proyect. Luego se despliega la ventana de la Figura 3 donde se selecciona la categoría Web y en proyectos Web Application.

Figura 3. Pantalla de presentación para la escogencia de un nuevo proyecto en Netbeans



A continuación se le da un nombre al proyecto, ver Figura 4 y se elige que tipo de servidor será el que acepte las peticiones al servicio, en este caso escogemos Glassfish V2 como se muestra en la Figura 5.

Figura 4. Pantalla de presentación para la asignación del nuevo nombre de un proyecto

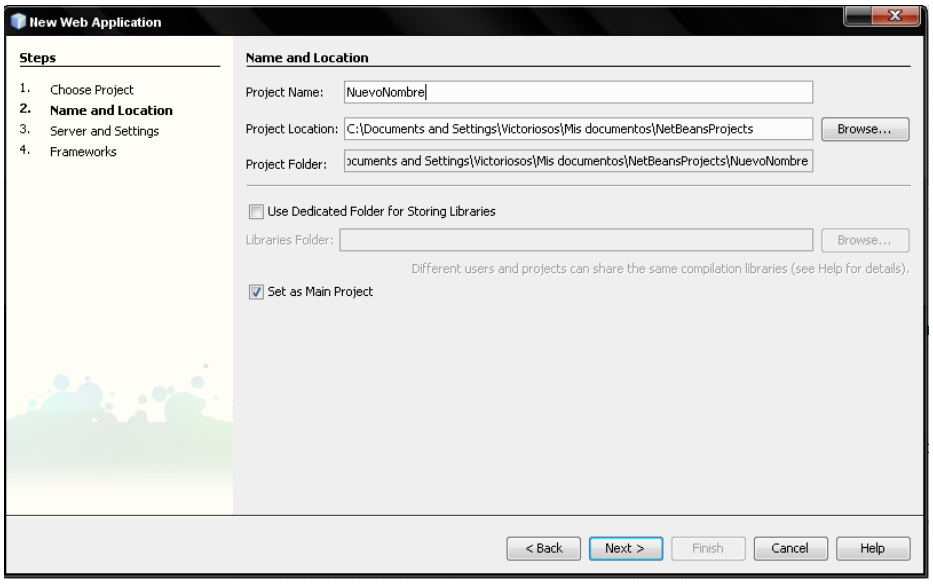
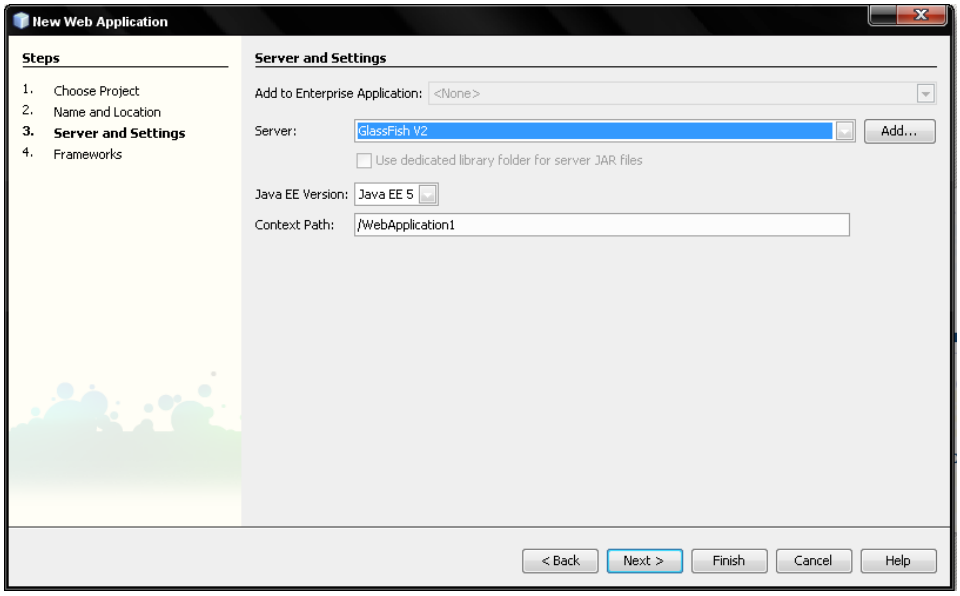


Figura 5. Pantalla de presentación para la elección del tipo de servidor Web



Presione el botón Finalizar.

Netbeans ofrece dos opciones para crear un servicio web

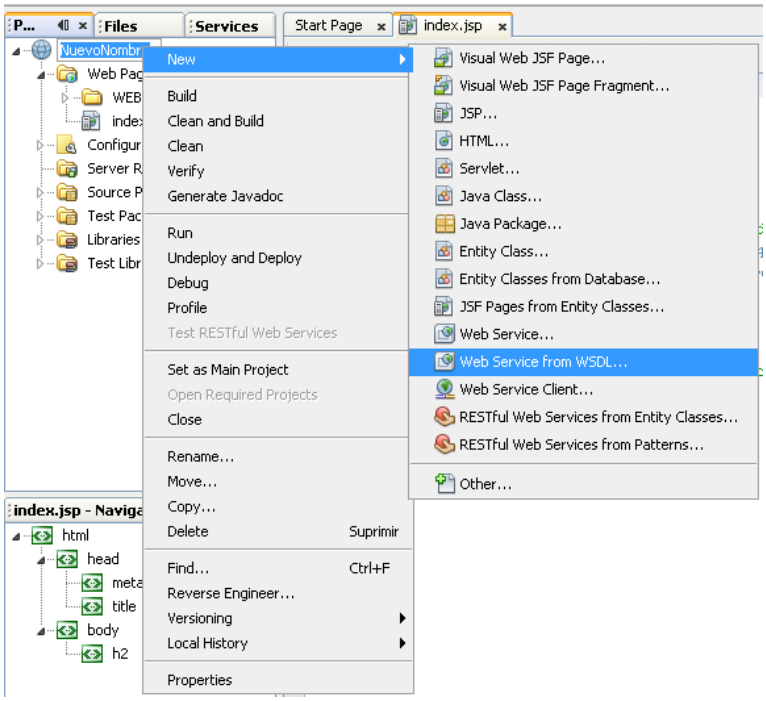
- Web Service from WSDL
- Web Service

4.1.1. *WEB SERVICE FROM WSDL*

Esta opción permite crear un servicio web a partir de un archivo .wsdl creado manualmente u obtenido desde alguna fuente externa.

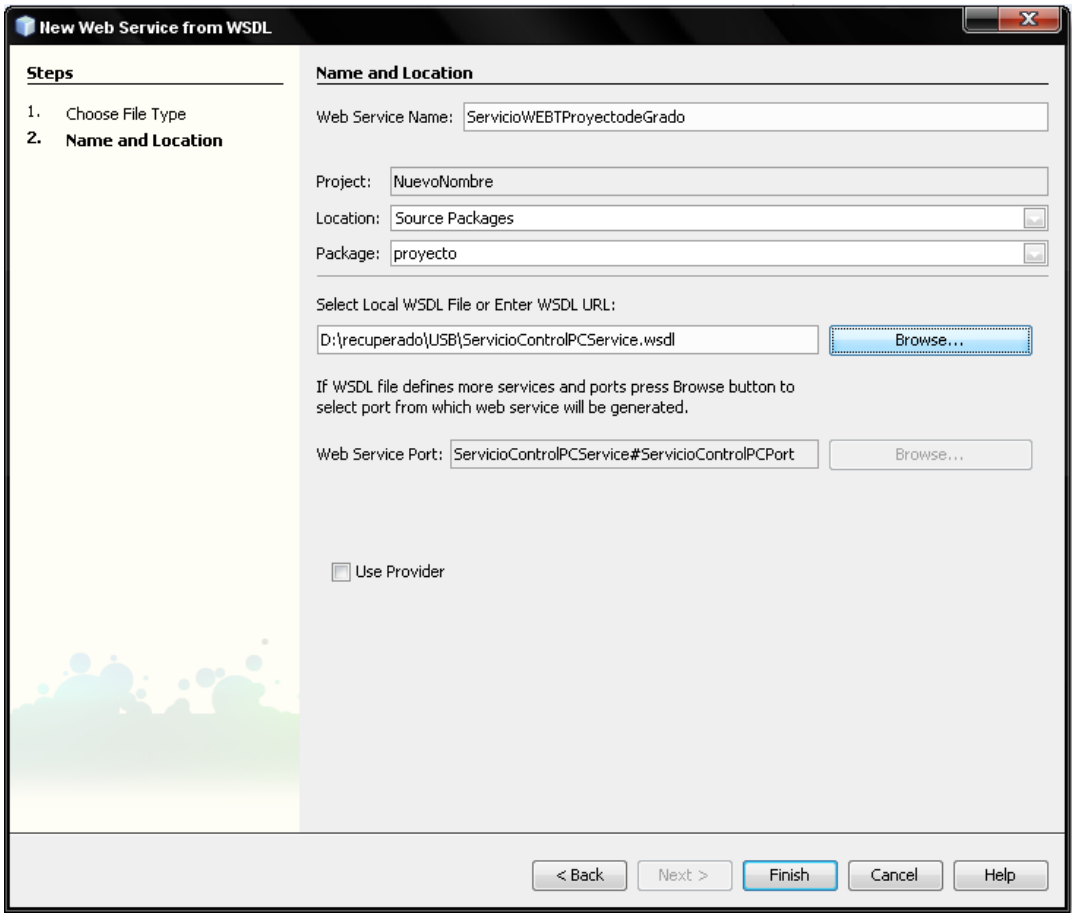
- Click derecho sobre el nombre del proyecto, ver Figura 6.

Figura 6. Despliegue de opciones para la elección de un servicio desde un WSDL existente



- New -> Web Service from WSDL
- Se asigna un nombre para el servicio, ver Figura 7
- Se da un nombre al paquete

Figura 7. Pantalla de presentación para la asignación de un nuevo nombre del servicio.



En este caso el nombre del servicio fue “ServicioWEBTProyectodeGrado” y el paquete se llamo “proyecto”.

- Finalizar

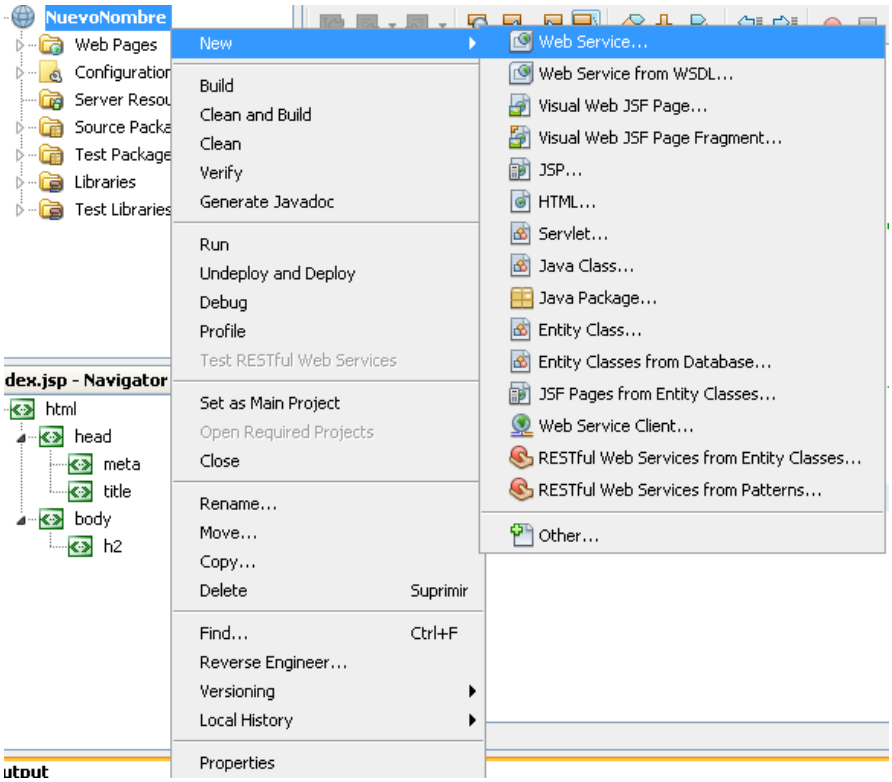
Una vez se finaliza se crean los stubs, los cuales son archivos escritos en lenguaje java que permiten al programador olvidarse de los detalles de la conexión del cliente con el servidor.

4.1.2. *WEB SERVICE*

Esta opción permite crear un servicio web de una forma más intuitiva, pues provee las herramientas necesarias para crear cada método y este a su vez crea el archivo WSDL de una manera adecuada.

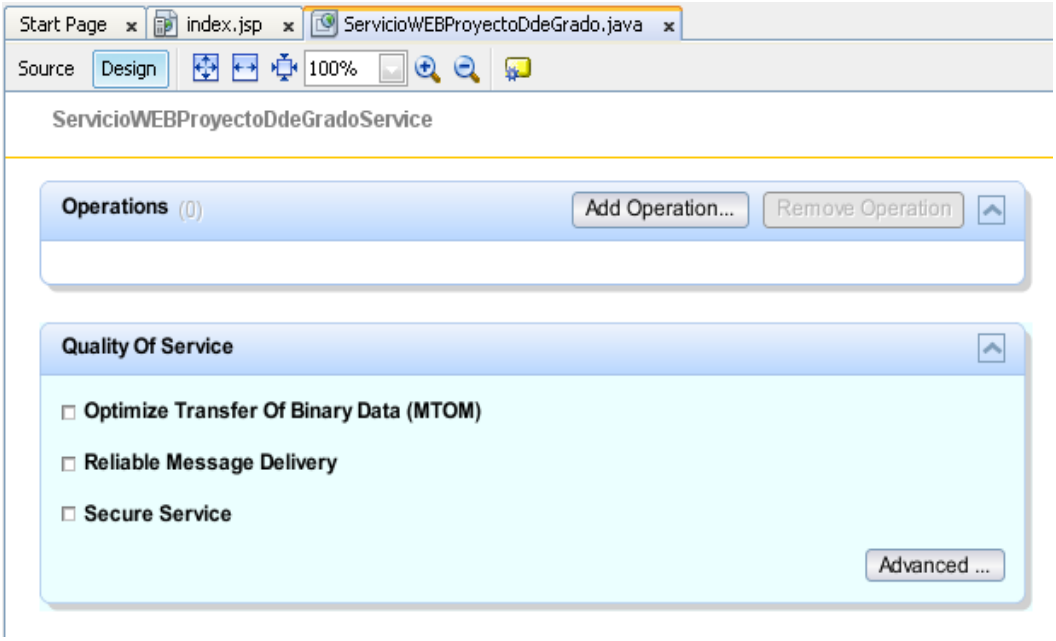
- Click derecho sobre el nombre del proyecto
- New -> WebService, ver Figura 8

Figura 8. Despliegue de opciones para la creación de un servicio Web desde cero



- Después de asignar un nombre al servicio y al paquete se presiona Finalizar.
- Luego aparece una ventana como la que se presenta en la Figura 9.

Figura 9. Pantalla inicial en la cual no se han creado aún operaciones



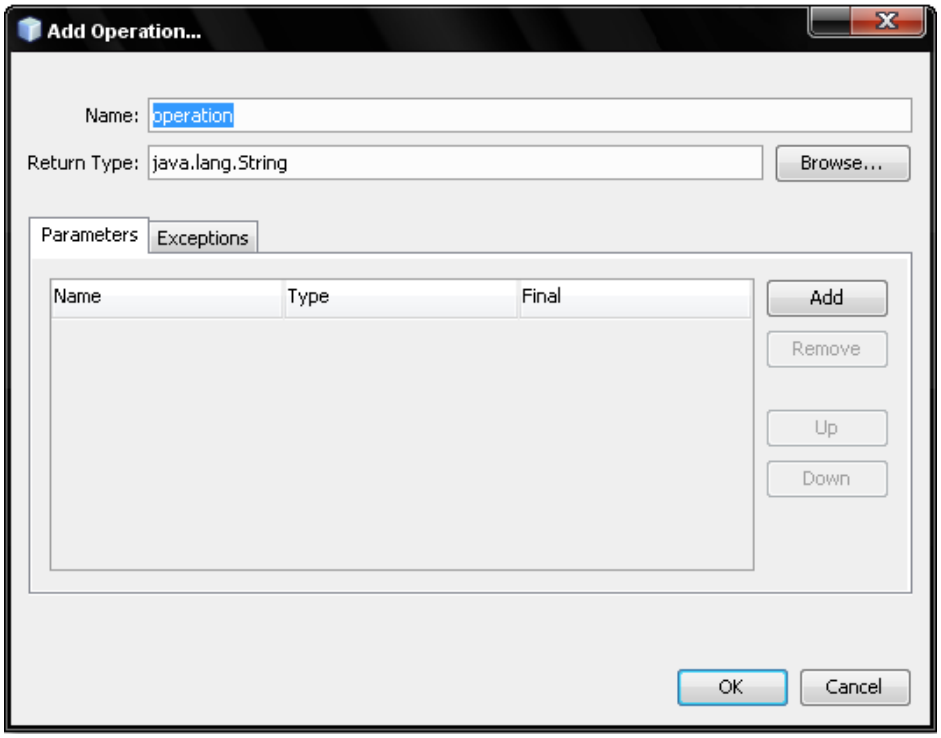
En este caso debido a que se va a crear un servicio desde cero no existen operaciones por ello el cuadro Operations(0) esta vacio, si hubiésemos creado el servicio a partir de un archivo WSDL entonces este espacio debería estar con varios métodos añadidos.

4.2. ADICIONAR OPERACIONES

Una de las razones por las cuales utilizamos la opción “Web Service” es la posibilidad de crear nuestro servicio desde cero de acuerdo a nuestras necesidades. El botón Add Operation permite la agregar nuevos métodos al servicio.

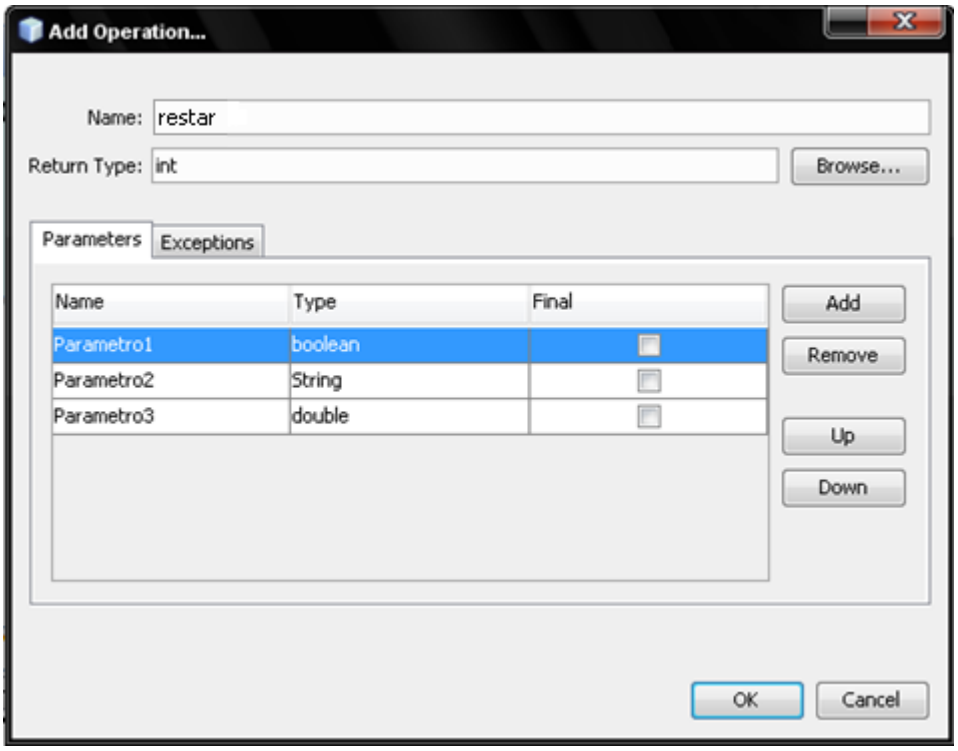
Una vez se presiona este botón aparece el cuadro de dialogo de la Figura 10.

Figura 10. Pantalla para la adición de una nueva operación del servicio



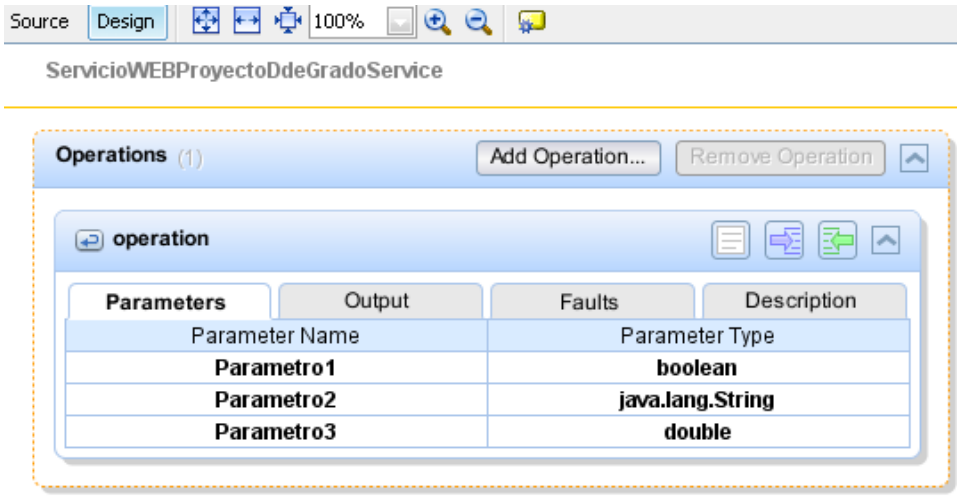
En el cual es posible renombrar nuestro método, decidir qué tipos de parámetros tendrá y además cual será su retorno. En la Figura 11 se expone un ejemplo de un nuevo método llamado restar, el cual tiene los parámetros: parametro1de tipo boolean, parametro2 de tipo String y parametro3 de tipo double. El método retorna un valor de tipo int (entero).

Figura 11. Adición del método restar con tres parámetros de entrada



El resultado final de la adición del método al servicio es el que se muestra en la Figura 12.

Figura 12. Resultado final de la adición del método restar



4.2.1. LA VISTA DISEÑO (DESIGN)

La vista diseño es la que se muestra en la Figura 12. Los resultados de la creación de un nuevo método son desplegados inmediatamente sobre la vista Diseño, la cual se utiliza para adicionar nuevas operaciones y para configurar el servicio Web.

4.2.2. LA VISTA FUENTE (SOURCE)

La vista fuente se utiliza para implementar los métodos creados en la vista diseño, muestra el código fuente generado automáticamente por el IDE incluyendo las anotaciones @WebService para la clase y @WebMethod para los métodos, son obligatorias para la creación de un servicio web. En la Figura 13 se muestra el código generado de la función restar, creada en el apartado 1.8.2.

Figura 13. Vista Fuente

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package proyecto;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

@WebService()
public class ServicioWEBProyectoDdeGrado {
    /**
     * Web service operation
     */
    @WebMethod(operationName = "operation")
    public int restar(@WebParam(name = "Parametro1")
    boolean Parametro1, @WebParam(name = "Parametro2")
    String Parametro2, @WebParam(name = "Parametro3")
    double Parametro3) {
        //TODO write your implementation code here:
        return 0;
    }

}
```

El código que implementará éste método se escribe en el lugar donde dice: "//TODO write your implementation code here:". En la Imagen 18 se muestra la implementación del método restar.

Figura 14. Implementación

```
package proyecto;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
```

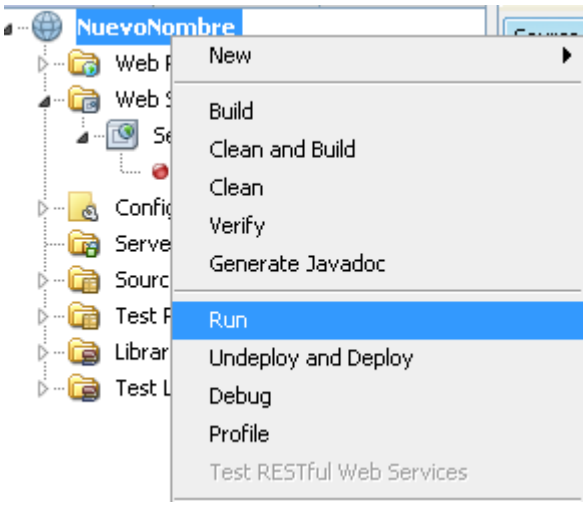
```java
@WebService()
public class ServicioWEBProyectoDdeGrado {
    /**
     * Web service operation
     */

    @WebMethod(operationName = "operation")
    public int restar(@WebParam(name = "Parametro1")
    boolean Parametro1, @WebParam(name = "Parametro2")
    String Parametro2, @WebParam(name = "Parametro3")
    double Parametro3) {

        int retorno;
        if (Parametro1){
            retorno = (int)Parametro3 - 100;
        }else{
            retorno = (int)Parametro3 - 1000;
        }
        System.out.println("Hola soy "+Parametro2);
        return retorno;
    }

}
```

4.3. DESPLEGAR EL SERVICIO

Para desplegar nuestro servicio Web y asegurarnos que está listo para su funcionamiento, se debe escoger uno de los siguientes:

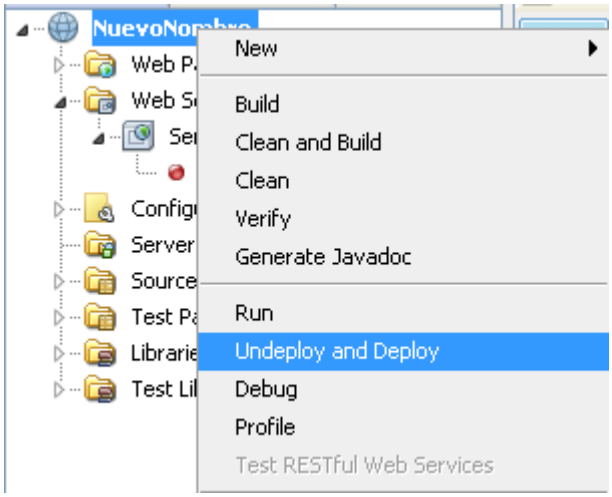
- Run: Inicia el servidor Glassfish si no está iniciado y despliega la aplicación web con los servicios web que contenga. Ver Figura 15.

Figura 15. Despliegue de operaciones para la ejecución del servicio



- Undeploy and deploy: Si ya está desplegada la aplicación web entonces la desmonta, y luego la despliega quedando lista para recibir las peticiones desde una aplicación cliente. Ver Figura 16.
-

Figura 16. Despliegue del servicio



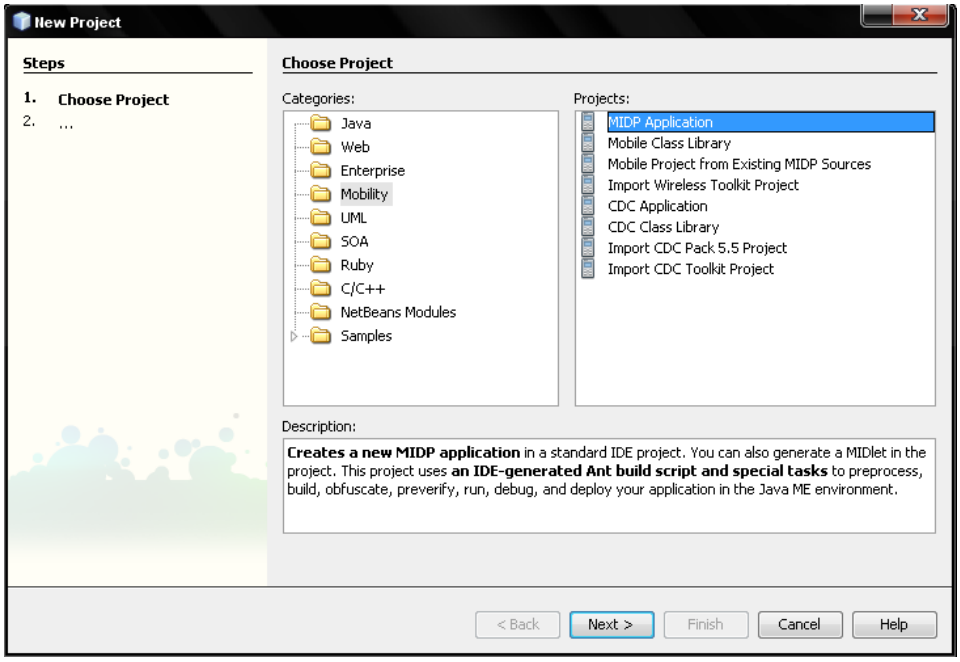
4.4. COMO REALIZAR UN CLIENTE DEL SERVICIO WEB EN J2ME

Una vez se ha creado el servicio Web, es necesario crear un cliente se conecte a él para hacerle peticiones, en este caso se describirá el proceso de crear un cliente móvil utilizando Java 2 Micro Edition (J2ME).

Los pasos para crear un cliente móvil son:

- En el menú elija File -> New Proyect
- Luego seleccione la categoría "Mobility", después el tipo de proyecto "MIDP Application" y presione el botón Next. Ver Figura 17.

Figura 17. Pantalla de presentación para la creación de un proyecto móvil

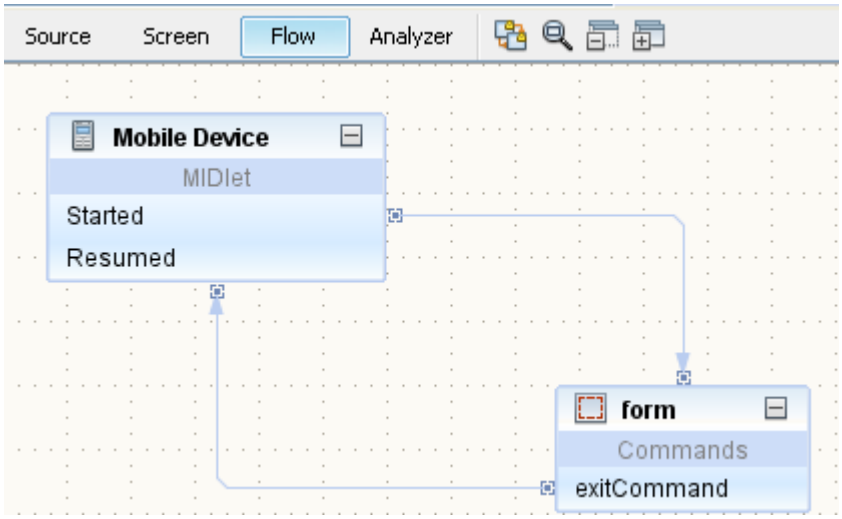


Nombramos el proyecto y habilitamos las opciones "Set as Main proyect" para establecer el proyecto actual como el proyecto principal y "Create Hello Midlet" para generar una aplicación móvil que muestra el popular mensaje "Hello World".

- Finalizar.

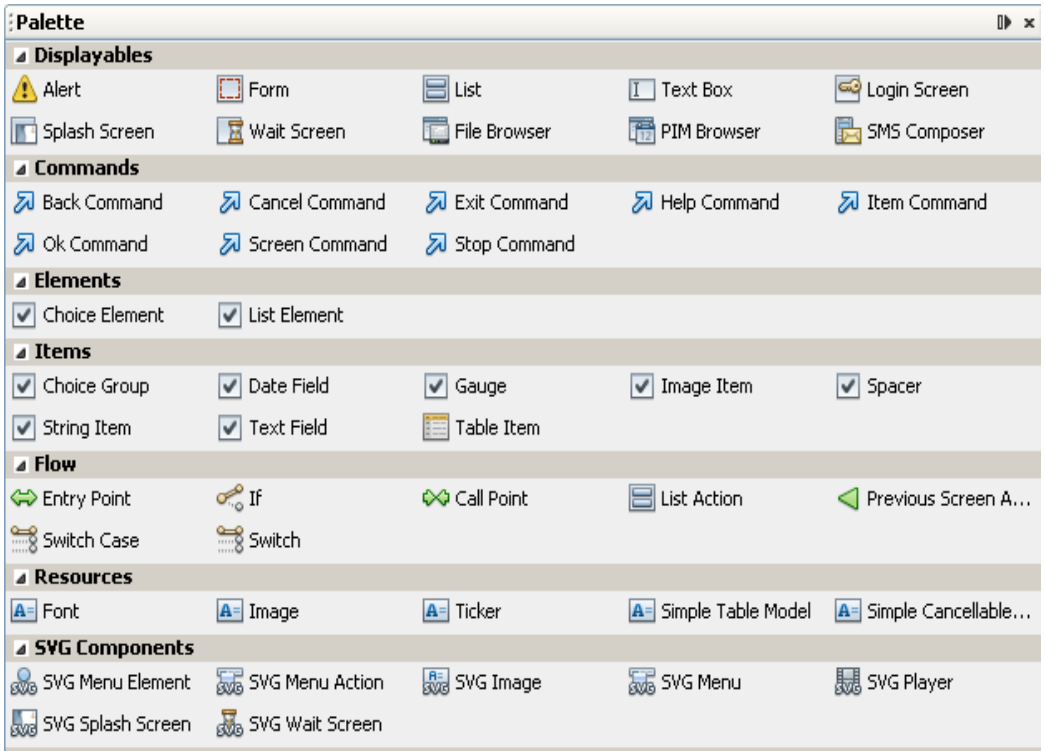
En pantalla aparece la ventana que se muestra en la Figura 18.

Figura 18. Vista Flow



Dicha pantalla tiene a su disposición las siguientes herramientas en la vista Palette

Figura 19. Vista Palette

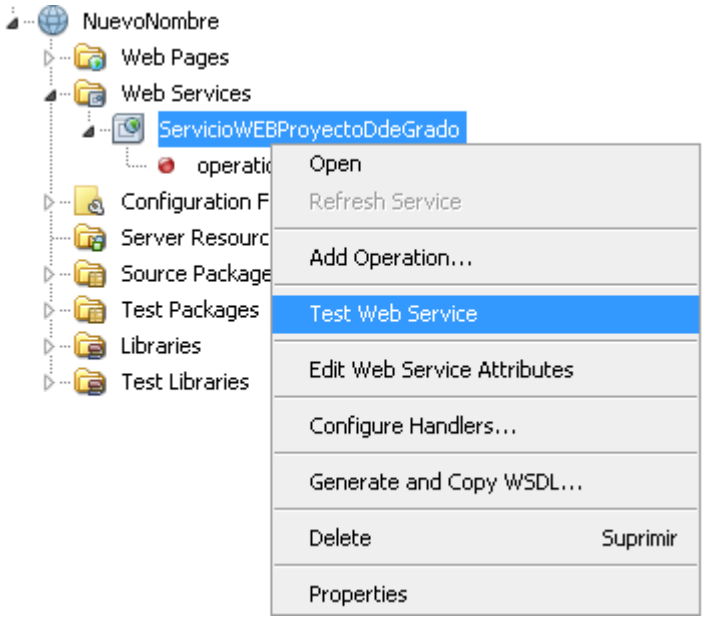


Cada botón agrega un tipo específico de elemento gráfico al proyecto, los elementos más importantes para este tipo de proyecto son los "Displayables" y representan ventanas en el celular. En ellos se pueden adicionar Commands, útiles para recibir las órdenes del usuario. En el Displayable Form es una ventana vacía y se le pueden adicionar los elementos Items de la paleta, los cuales se usan para desplegar información, permitir la escritura del usuario o mostrar imágenes. A los Form también se les puede adicionar los elementos Elements de la paleta, los cuales permiten la selección múltiple o única de opciones.

Ahora es necesario relacionar ambos cliente y servicio, esto se hace obteniendo primero la ubicación del archivo WSDL del servicio.

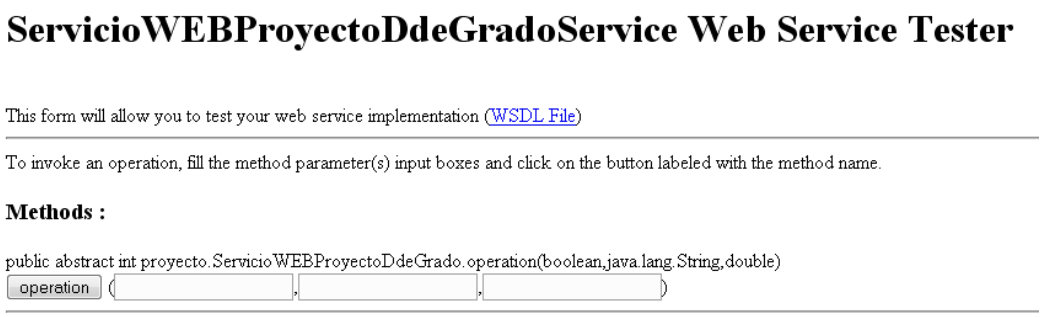
En el nodo Web Services del servicio se da click derecho al servicio y se presiona "Test Web Service" como se muestra en la Figura 20. Nota, esta opción solo funciona si ya se desplegó el servicio.

Figura 20. Despliegue de opciones para la prueba de un servicio Web



Luego se abre el navegador con una página web donde se puede probar el servicio web, ver la Figura 21.

Figura 21. Prueba del servicio web

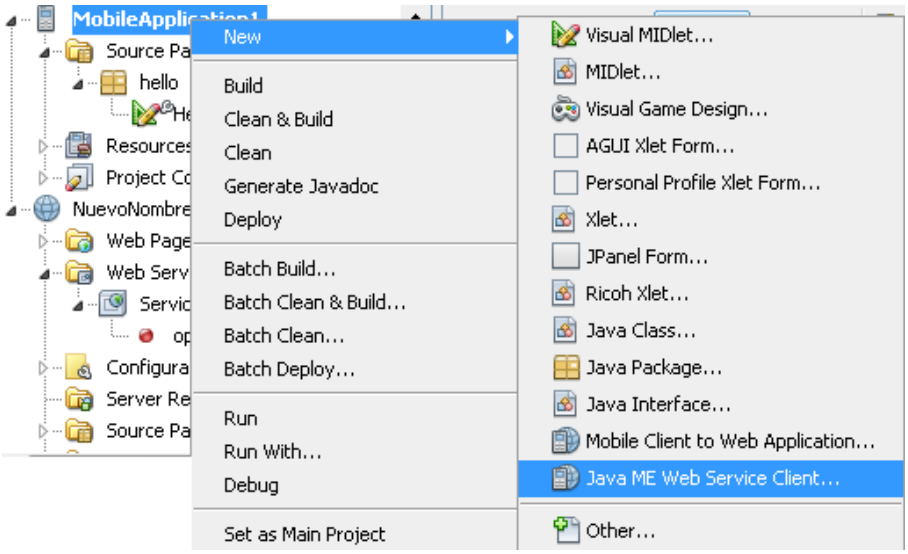


Después copie la ruta del enlace "WSDL File" y si desea ver el documento WSDL haga click en el mismo. En nuestro caso la ruta del documento WSDL es http://localhost:8080/NuevoNombre/ServicioWEBProyectoDdeGradoService?WSDL.

El siguiente paso es generar los archivos de la conexión esenciales para permitir la comunicación entre el cliente y el servidor.

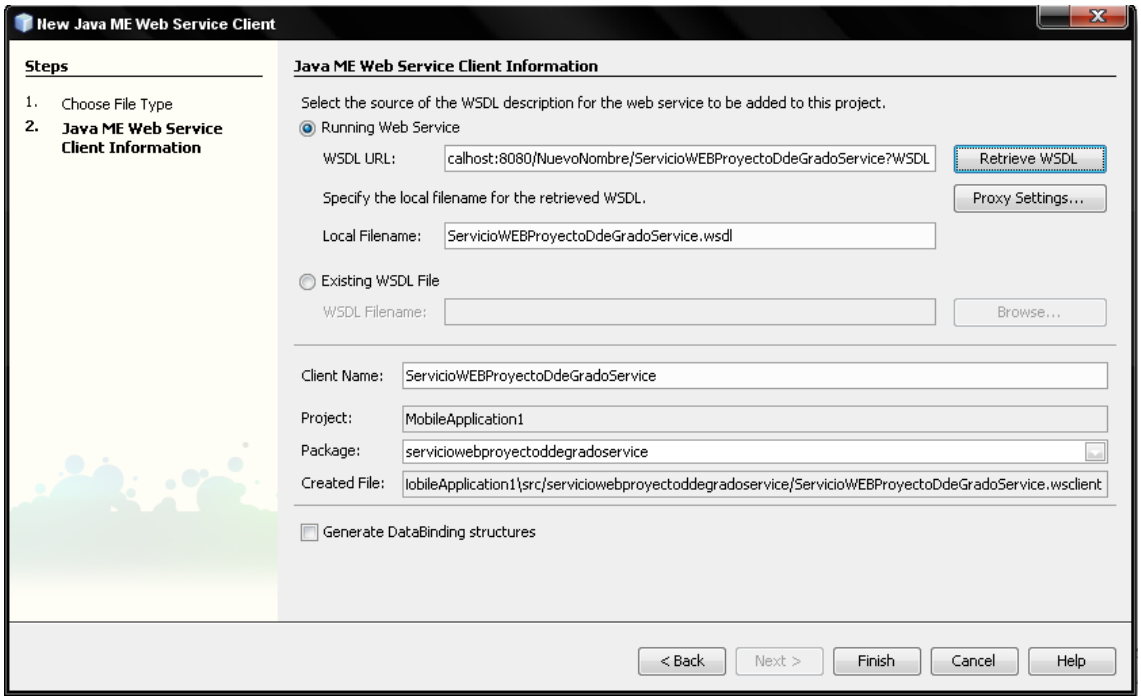
- Click derecho sobre el nombre del proyecto cliente
- Seleccione “New” y “Java ME Web Service Client…” Ver la Figura 22.

Figura 22. Generación del stub



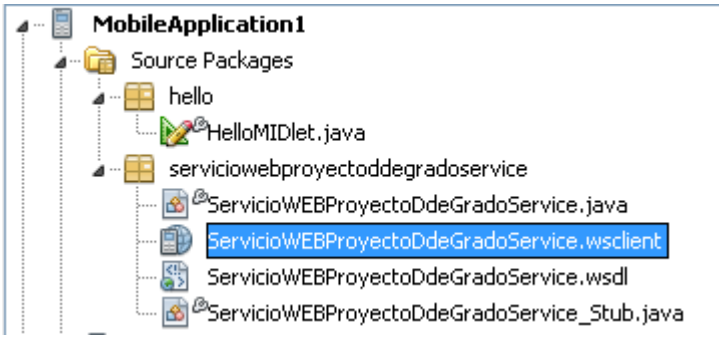
Luego se despliega el cuadro de diálogo “New Java ME Web Service Client”, ver Figura 23. En el campo WSDL URL se debe pegar la ruta de WSDL antes copiada. Después presione el botón “Retrieve WSDL” y luego “Finish” entonces se generan los archivos del stub.

Figura 23. Creación del Cliente



Los archivos del stub permanecen ahora en un paquete a parte listos para ser utilizados.

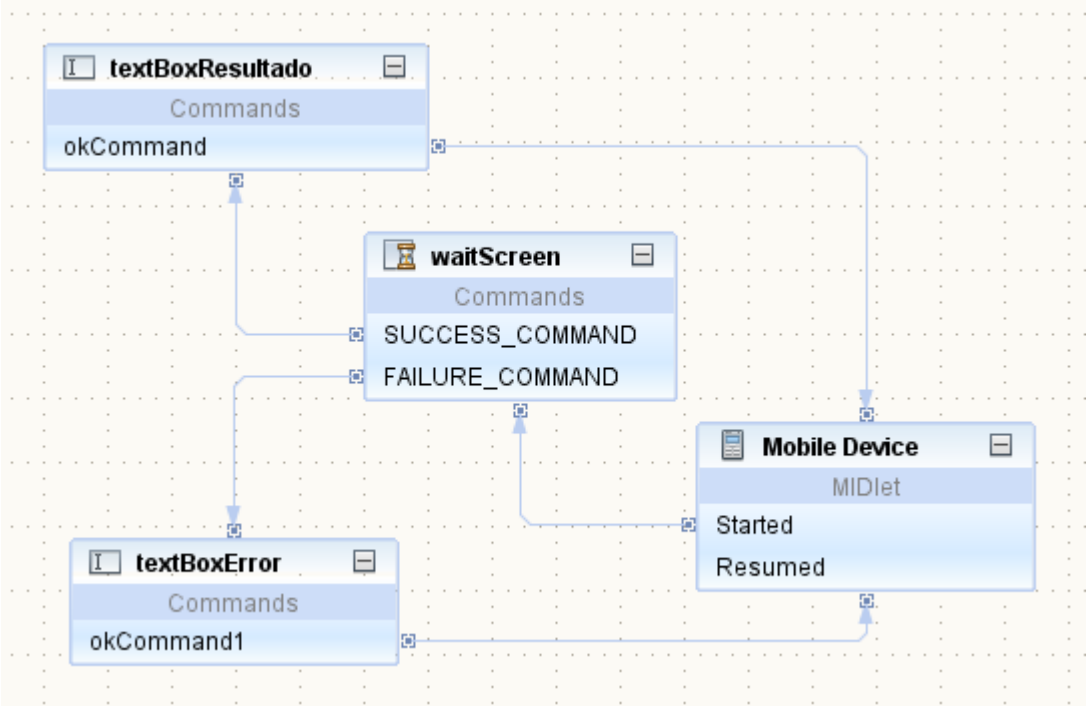
Figura 24. Lista de archivos generados



El llamado a alguno de los métodos del servicio debe hacerse por medio de un hilo pues las llamadas son bloqueadoras (síncronas), por ello se utilizan los Displayables “Wait Screen” sobre los cuales es posible adicionar código que luego será llamado sobre un hilo y además mostrara una pantalla de espera mientras este se ejecuta.

De esta manera la vista de flujo propuesta es la que se muestra en la Figura 25. Una vez iniciada la aplicación va a la ventana waitScreen, si la ejecución del método remoto lanza una excepción, entonces va a mostrar el error en el textBoxError, si la ejecución es exitosa entonces se muestra el resultado del método remoto en el textBoxResultado.

Figura 25. Diagrama de flujo del Cliente



Ahora en la vista fuente (source) se busca el método "SimpleCancellableTask getTask()", ver Figura 26.

Figura 26. Método SimpleCancellableTask

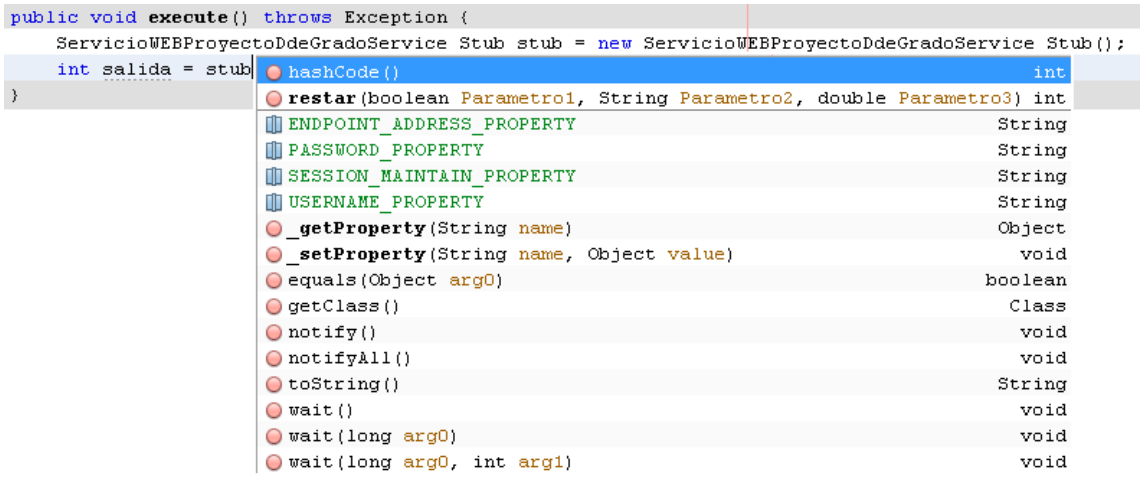
```
//<editor-fold defaultstate="collapsed" desc=" Generated Getter: task ">
/**
 * Returns an initiliazed instance of task component.
 * @return the initialized component instance
 */
public SimpleCancellableTask getTask() {
    if (task == null) {
        // write pre-init user code here
        task = new SimpleCancellableTask();
        task.setExecutable(new org.netbeans.microedition.util.Executable() {
```

```
            public void execute() throws Exception {
                // write task-execution user code here
            }
        });
        // write post-init user code here
    }
    return task;
}
//</editor-fold>
```

El método execute pertenece a una clase anónima que implementa la interface Executable. En este método se hace el llamado al método del servicio. Para esto es necesario crear una instancia del stub.

Una vez se instancia es posible observar por instrospección los metodos que generamos en el servicio, en este caso solo se creo uno restar(). Ver Figura 27.

Figura 27. Llamado a un método remoto



Así que el código propuesto para el método execute es el de la Figura 28.

Figura 28. Código de la invocación del método remoto

```
public void execute() throws Exception {
    ServicioWEBProyectoDdeGradoService_Stub stub = new ServicioWEBProyectoDdeGradoService_Stub();
    int salida = stub.restar(true, "proyecto", 1000);
    getTextBoxResultado().setString(String.valueOf(salida));
}
```

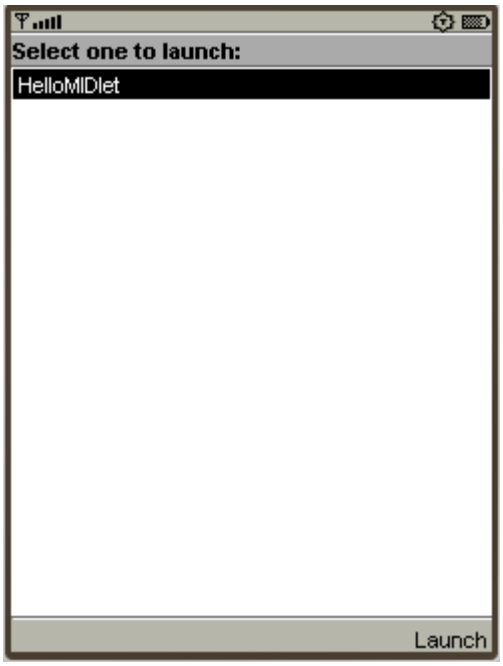
En caso de falla en el llamado a los métodos se mostrará en el textBoxError el mensaje de la excepción, para ello una posible solución es obtener la tarea del Wait Screen y llamar al método getFailureMessage(). Ver la Figura 29.

Figura 29. Obtener el mensaje de falla

```
    public void commandAction(Command command, Displayable displayable) {
        // write pre-action user code here
        if (displayable == textBoxError) {
            if (command == okCommand1) {
                // write pre-action user code here
                exitMIDlet();
                // write post-action user code here
            }
        } else if (displayable == textBoxResultado) {
            if (command == okCommand) {
                // write pre-action user code here
                exitMIDlet();
                // write post-action user code here
            }
        } else if (displayable == waitScreen) {
            if (command == WaitScreen.FAILURE_COMMAND) {
                // write pre-action user code here
                getTextBoxError().setString(getTask().getFailureMessage());
                switchDisplayable(null, getTextBoxError());
                // write post-action user code here
            } else if (command == WaitScreen.SUCCESS_COMMAND) {
                // write pre-action user code here
                switchDisplayable(null, getTextBoxResultado());
                // write post-action user code here
            }
        }
        // write post-action user code here
    }
```

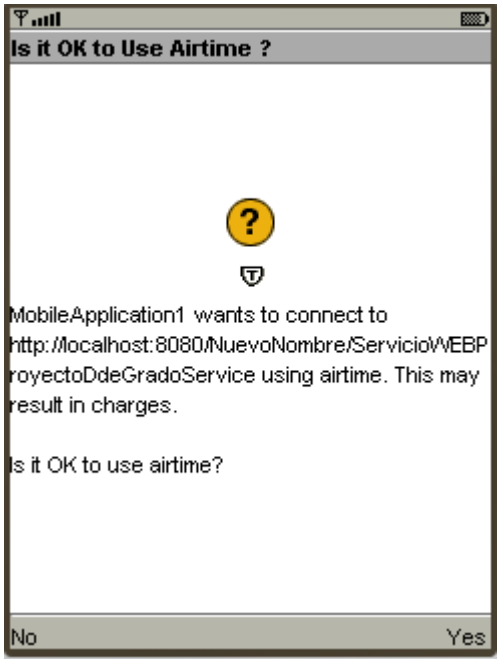
Ahora es tiempo de probar nuestro cliente, vaya al menú Run y luego elija Run Main Project. La primera ventana que aparece se muestra en la Figura 30, después se presiona Launch.

Figura 30. Prueba del Cliente



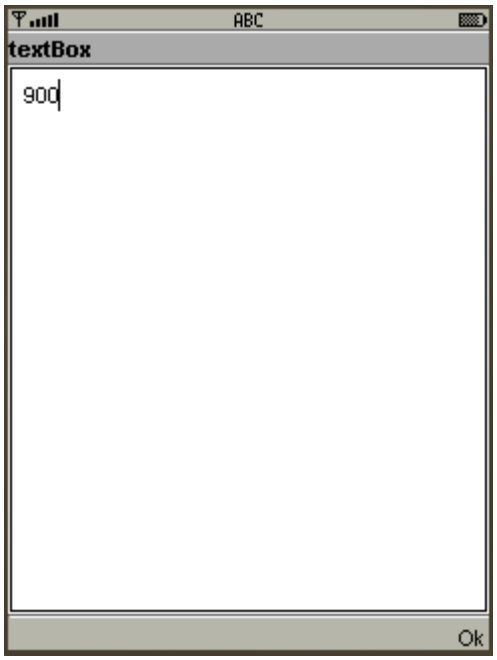
Al igual que en el celular se solicita al usuario su aprobación para enviar un mensaje a través de internet. Ver Figura 31.

Figura 31. Solicitud de Conexión



Después se presiona yes, y si no existe alguna falla aparecerá el resultado de la operación. Ver la Figura 32. Lo cual es lógico debido a que Parametro1 fue true y Parametro3 1000.

Figura 32. Resultado de la Operación



En la consola del servidor Glassfish debe haber aparecido el mensaje que se mandó a imprimir "Hola soy proyecto", Ver Figura 33.

Figura 33. Salida de la consola de Glassfish

5. HERRAMIENTAS UTILIZADAS

5.1. EN LA INGENIERÍA DE SOFTWARE

La herramienta utilizada para el diseño de los diagramas UML fue la suministrada por Netbeans, la cual viene incluida con este a partir de la versión 6.0, anteriormente fue un paquete adicional el cual era necesario instalar desde internet.

La decisión de usarla fue basada en la idea de crear diagramas UML 2.0 en el mismo IDE donde se iba a codificar la aplicación para la generación posterior de código a partir de los diagramas diseñados. Además debido a que tenemos experiencia con el desarrollo de aplicaciones en Netbeans decidimos probar esta herramienta.

Finalmente en el momento de la generación de código, se obtuvo un proyecto con las clases y firmas de los métodos establecidos previamente en la ingeniería realizada.

Los inconvenientes que se presentaron fueron debido a que la opción deshacer no existe y además en ocasiones la funcionalidad de guardar no funciona adecuadamente, y es probable encontrar luego los diagramas incompletos o sin los últimos cambios.

Esperamos que en futuras versiones se mejoren estos problemas porque consideramos que a pesar de ello es una herramienta muy amigable y completa.

5.2. EN LA IMPLEMENTACION

5.2.1. NETBEANS

En la implementación se utilizó Netbeans, uno de los IDEs más populares, permite el desarrollo de aplicaciones J2ME para dispositivos móviles, aplicaciones de escritorio J2SE y aplicaciones Web y empresariales J2EE.

Al momento de instalar posee diversos e interesantes aditamentos para la realización de muchos tipos de proyectos. Entre los cuales se cuenta con:

- Base IDE: Núcleo de componentes de un entorno de desarrollo integrado.
- Java SE: Herramientas esenciales para la programación en Java, incluye un editor, soporte para refactorización y una completa herramienta para diseño de interface de usuario GUI.
- Web & Java EE: Herramientas para la creación de aplicaciones Web con Java y aplicaciones empresariales J2EE. Incluye soporte para Servlets, JSP, Struts, una API para el manejo de la persistencia de los datos, Java Beans empresariales, JAX- WS, y Java Server Faces.
- Mobility: Es una herramienta para la creación de aplicaciones Java Micro Edition, para celulares y dispositivos portátiles. Incluye un diseñador visual, conexión a servicios Web y otras herramientas para la fragmentación de problemas.
- UML: Herramientas de Lenguaje unificado de Modelado, soporte para diseño de diagramas, generación de código, y creación automática de diseños a partir de código existente.
- SOA: Herramientas para la creación de Servicios Orientados a la Arquitectura. Provee comprensivo soporte BPEL. Incluye un editor grafico, un depurador, y soporte para refactorización.
- Ruby: Completo soporte para creación de aplicaciones Ruby on Rails.
- C/C++: Herramientas para desarrollo de aplicaciones C y C++.
- Glashfish: Es una implementación open-source de un servidor para la especificación Java EE 5.
- Apache Tomcat 6.0: Es una implementación open-source de un servidor Java Servlet y especificación JSP.

Es un entorno integrado de desarrollo libre y de código abierto. Además funciona en Windows, Linux, MAC OS X y Solaris.

Entre las características más importantes de este IDE se cuenta con la generación automática de stubs en aplicaciones móviles, generación de archivos WSDL, creación de un servicio web a

partir de un documento WSDL, el programador puede utilizar una interfaz gráfica para ingresar los métodos a publicar, el ide genera las clases necesarias para gestionar la conexión y luego el programador solo se preocupa por implementar los métodos del servicio.

5.2.2. GLASSFISH

Glassfish es un servidor de aplicaciones libre, de código abierto el cual implementa las características más nuevas en la plataforma java EE 5. Los requerimientos mínimos de instalación en Windows son 1 GB de RAM, 500 MB de espacio en disco.

La versión utilizada fue Sun Java System Aplication Server 9.1. GlassFish se encarga de desplegar el servicio Web de manera que un cliente con un stub creado a partir del WSDL del servicio pueda hacerle las peticiones pactadas en el WSDL.

5.2.3. P*OSTGRESQL*

Es un sistema de gestión de bases de datos Objeto-Relacional de código abierto, el cual fue utilizado para almacenar la información de usuarios, e historial de las acciones del usuario.

5.3. EN LAS PRUEBAS

Durante las pruebas se utilizó el Celular Sony Ericsson K510a uno de los celulares más baratos que tenían implementada la JSR 172 actualmente cuesta $230.000 en plan prepago.

6. VENTAJAS DE CONTROLPC

- Ejecución remota de comandos de consola en el PC
- Funciona en celulares, otras aplicaciones en el mercado no funcionan en dispositivos tan limitados en recursos como los celulares.
- Consumo menor de ancho de banda ya que no transmite pantallazos del computador remoto. Lo cual se traduce en reducción de costos para el usuario.
- Maneja el mínimo de información para realizar su tarea lo cual hace que se pueda usar con celulares de menor capacidad.
- La aplicación cliente podría funcionar en PDAs o Smartphone si se le instala una máquina virtual de java que tenga implementada la JSR 172.
- Actualmente hay mayor cantidad de usuarios de celulares que de Smartphone o PDA, por esto consideramos que podría tener mayor cantidad de usuarios potenciales.
- La aplicación cliente aprovecha el uso de la red celular para acceder a internet la cual tiene muy buena cobertura, permitiéndole al usuario acceder a su computador desde otras ciudades, zonas rurales o incluso cuando está viajando.
- Es más rápido que otras soluciones porque la aplicación cliente queda instalada en el celular, además ofrece una interfaz más amigable que la que se puede ofrecer desde una página WAP 2.0 la cual utiliza un subconjunto de XHTML en celulares.
- Como la aplicación cliente está hecha en Java, la interfaz de usuario puede ser más amigable si el sistema operativo del celular ofrece mejores opciones para ingresar texto.

7. PROBLEMAS Y SOLUCIONES

Durante el desarrollo del proyecto surgieron diversos problemas los cuales fueron solucionados satisfactoriamente:

- Envío de archivos grandes: se deseaba transferir un archivo desde el PC hasta el celular pero no había memoria para almacenar todo el archivo temporalmente, por ello se decidió que la aplicación debe solicitar partes de mínimo 1024 bytes de dicho archivo para luego escribirlos en la memoria de almacenamiento permanente.

- Envío de los resultados de búsqueda y exploración de carpetas en el PC: no siempre es posible y puede ser costoso enviar todos los resultados de una búsqueda al celular debido a que su tamaño puede ser demasiado grande y el usuario no siempre va a querer ver toda la lista, se decidió entonces recuperar solo diez datos para llenar la ventana que el usuario ve en cada instante.

- Lentitud de la Visual Mobile Designer: ésta herramienta permite diseñar visualmente la interfaz grafica de las aplicaciones móviles, el problema surge cuando se crean muchos elementos tipo Displayable [3] que están unidos por medio commands [4] a partir de un único archivo lo cual se vuelve demasiado lento y la herramienta se vuelve inmanejable, por ello se decidió utilizar únicamente el código que esta genera para continuar con la realización del diseño previo de la interfaz de la aplicación cliente.

- Demasiadas solicitudes de permisos de usuario para leer y escribir los datos en el celular: Cuando se intenta transferir un archivo desde el PC al celular, debido a que se transmiten partes de un archivo, el sistema operativo del celular le pregunta al usuario si desea permitirle a la aplicación cliente leer y escribir los datos de usuario, cuando el archivo es grande esto puede ser muy incomodo haciéndose necesario tener algún mecanismo que evite esa lluvia de permisos, como alternativa se propone firmar la aplicación cliente con un certificado digital que puede ser comprado a las empresa VeriSign, Ahuate u otras, el costo para un certificado de este tipo en Thawte puede ser de US230 con duración de un

[3] Son ventanas en el celular

[4] Los commands proporcionan al usuario la capacidad de interactuar con el MIDlet seleccionando funcionalidades. Es el método por el que el usuario introduce “ordenes” en el MIDlet.

año con la ventaja de que la firma sirve para todas las copias del mide. Los Mide firmados pueden ser instalados solo en celulares donde el nombre de la empresa aparezca en la lista de certificados confiables del mismo.

- Como capturar las excepciones en el celular lanzadas por un método remoto: La clase stub debe implementar la interface FaultDetailHandler la cual obliga a definir el método handleFault. El método del Stub que captura la excepción siempre instancia un objeto tipo Operation, en su uso normal se utiliza la firma del método estático newInstance(QName arg0, Element arg1, Element arg2), ahora se utiliza una firma que incluye además el argumento FaultDetailHandler newInstance(QName arg0, Element arg1, Element arg2,FaultDetailHandler arg3). Como se menciono anteriormente debido a que stub implementa la interface FaultDetailHandler entonces allí se debe entregar la referencia del stub this. Luego se debe capturar la excepción FaultDetailException la cual contiene el mensaje de la excepción y se obtiene a través del método getFaultDetail en el elemento cero. A continuación se muestra un ejemplo de la utilización de las excepciones en el método crear carpeta. Ver Figura 33.

Figura 34. Método crearCarpeta

```
public String crearCarpeta(String nuevaCarpeta)
        throws java.rmi.RemoteException, controlException {
    Object inputObject[] = new Object[]{
        nuevaCarpeta
    };

    Operation op = Operation.newInstance(
            _qname_operation_crearCarpeta,
            _type_crearCarpeta,
            _type_crearCarpetaResponse, this);
    _prepOperation(op);
    op.setProperty(Operation.SOAPACTION_URI_PROPERTY, "");
    Object resultObj;

    try {
        resultObj = op.invoke(inputObject);
    } catch (JAXRPCException e) {
        Throwable cause = e.getLinkedCause();
        if (cause instanceof java.rmi.RemoteException) {
            throw (java.rmi.RemoteException) cause;
        }
```

```
            if (cause instanceof FaultDetailException) {
                FaultDetailException fde = ((FaultDetailException) cause);
                String mensaje = (String) ((Object[]) fde.getFaultDetail())[0];
                throw new controlException(mensaje);
            }
            throw e;
        }

        return (String) ((Object[]) resultObj)[0];
    }
```

Implementación del método handleFault

```
public Element handleFault(QName arg0) {
    ComplexType c = new ComplexType();
    c.elements = new Element[]{
        new Element(new QName("message"), Type.STRING)};
    return new Element(new QName("message"), c);
}
```

8. RIESGOS DEL PROYECTO

- El mecanismo de seguridad utilizado para acceder al servicio desde el celular es utilizando cuentas de usuario las cuales consisten de Login, password e IMEI, elevando una excepción en caso de intentar usar el servicio si no se ha utilizado inicialmente el método validarUsuario e IniciarSesion satisfactoriamente. El problema consiste en que no se implemento algún otro mecanismo para cifrar los mensajes que se envían entre el cliente y el servidor y no se garantiza la integridad o la privacidad de los mensajes transmitidos.

- EL proyecto solo funciona cuando el Servidor donde se aloja el servicio Web cuenta con una IP estática.

- La eliminación de archivos y carpetas no envía los elementos a la papelera de reciclaje de Windows haciéndose necesario en caso de querer recuperarlos, alguna herramienta adicional especializada en la recuperación de archivos eliminados.

- Si la conexión se cae mientras se está haciendo una operación de transferencia de datos, podrían quedar archivos o carpetas incompletos.

9. RESULTADO FINAL

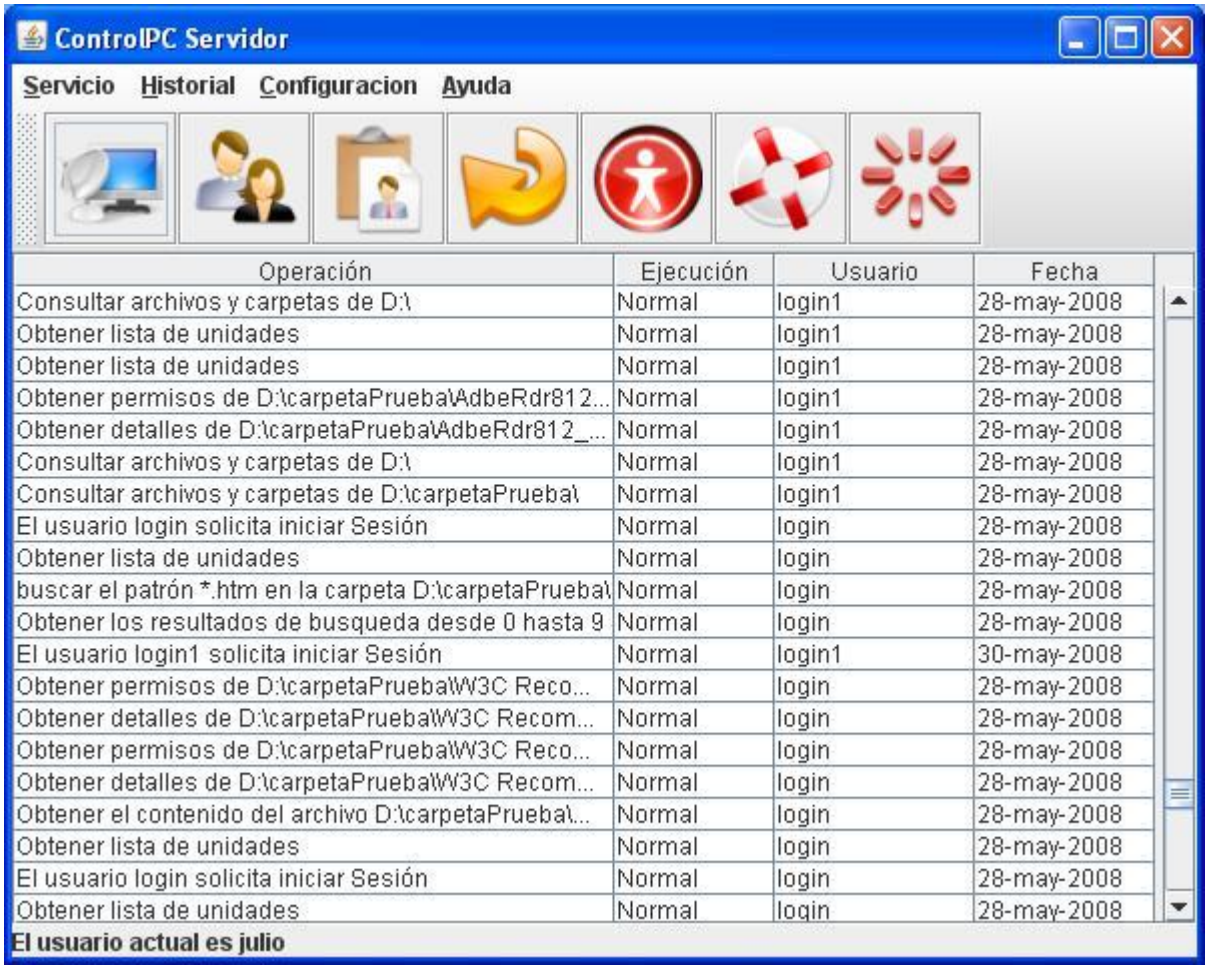
Todo lo planteado en la descripción del proyecto se probó satisfactoriamente y se obtuvieron los siguientes resultados:

- Permite registrar en el celular un servidor por medio de la dirección IP y el nombre del PC servidor.
- Recupera el listado de los procesos en ejecución en el servidor permitiendo finalizar el proceso seleccionado.
- Muestra la descripción del sistema Servidor como su sistema operativo, procesador, cantidad de memoria RAM total y disponible.
- Entrega el nombre del host, la dirección MAC y dirección IP del servidor.
- Se ejecutaron todos los comandos propuestos en los objetivos específicos desde el celular, tal como si se estuviera manejado la consola frente al computador.
- Se pueden realizar búsquedas sobre cualquier unidad de disco retornando aquellas rutas de archivo que coincidan con un patrón que utiliza los comodines de Windows * e ? con las restricciones como búsqueda de solo archivos ocultos, solo directorios o archivos de solo lectura.
- Es posible explorar el sistema de archivos del PC como si fuera propio del celular y realizar las operaciones de renombrar, eliminar y ver detalles de archivos y carpetas. También se puede copiar un archivo o carpeta al celular o a otra carpeta del mismo PC.
- Es posible explorar el sistema de archivos del celular y seleccionar un archivo para copiarlo al PC o en el mismo celular. Además permite mostrar un archivo de texto e imágenes jpg y png, eliminar archivos y carpetas.
- Es posible ver en la aplicación servidor el historial de las operaciones realizadas por un usuario y la gestión de cuentas de usuario.

10. MANUAL DE USUARIO

10.1. SERVIDOR CONTROLPC

Figura 35. Ventana Principal del Servidor



La aplicación servidor ControlPC le permite al Superusuario realizar las siguientes tareas:

- Iniciar el servicio

Para iniciar el servicio Web el Superusuario hace click en el botón "Iniciar Servicio", lo cual inicia el servidor de aplicaciones Glassfish y monta el servicio Web, dejando listo el servidor para recibir peticiones de un cliente. Este botón cambia de apariencia por la del botón "Suspender Servicio" una vez presionado. Ver Figura 35.

Figura 36. Botón Iniciar Servicio

- Suspender el servicio

Para suspender el servicio Web el Superusuario hace click en el botón "Suspender Servicio", lo cual cierra la sesión del cliente que esté conectado si lo hay, desmonta el servicio y para el servidor de aplicaciones. Una vez presionado cambia de apariencia por el botón "Iniciar Servicio" Ver Figura 36.

Figura 37. Botón Suspender

- Actualizar la tabla de historial

Para actualizar la tabla del historial el Superusuario presiona el botón "Actualizar historial", ver Figura 37. La aplicación servidor consulta en la base de datos las operaciones que han realizado los usuarios del servicio y las muestra en la tabla de la ventana principal.

Figura 38. Botón Actualizar

- Eliminar los registros seleccionados del historial

Para eliminar registros del historial el Superusuario debe seleccionar las filas que desea eliminar luego ir al menú Historial -> Eliminar. Ver Figura 38.

Figura 39. Menú Historial

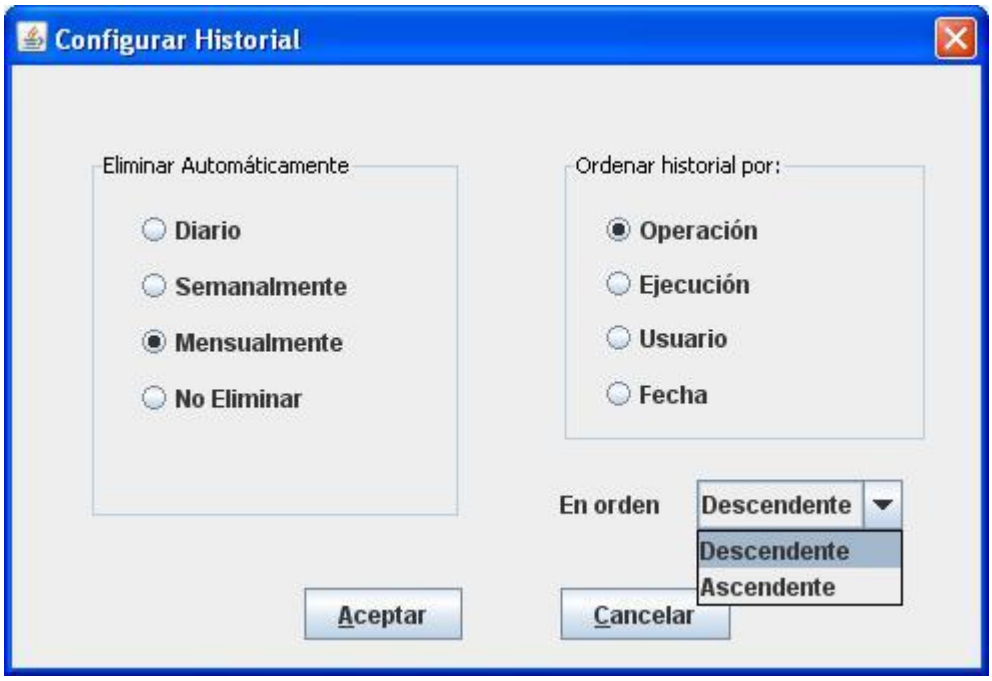
- Configurar el historial

Para configurar la forma en la cual se visualizará el historial se debe ir al menú Configuración -> Configurar Historial Ver Figura 39.

Figura 40. Menú Configuración

Luego de ello aparecerá el cuadro de dialogo "Configurar Historial" de la Figura 40 en la cual es posible seleccionar cada cuanto tiempo se eliminarán las entradas del historial y además la forma en la cual estará ordenado.

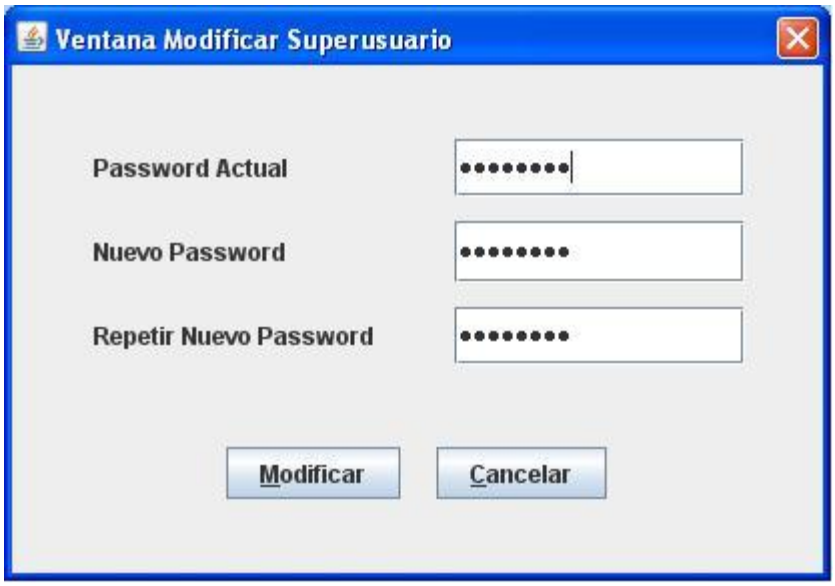
Figura 41. Cuadro de diálogo configurar historial



- Modificar el password del Superusuario

Para configurar la forma en la cual se visualizará el historial se debe ir al menú Configuración -> Modificar Superusuario ver figura 39. y se despliega la ventana de la Figura 41 donde se recibe el nuevo password de Superusuario

Figura 42. Ventana Modificar Superusuario

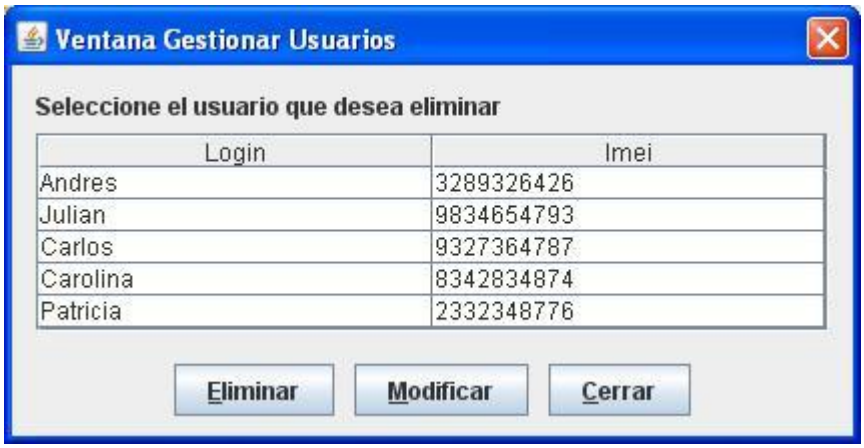


- Gestionar usuarios

Para gestionar las cuentas de usuario que están registradas el Superusuario debe presionar el botón "Gestionar usuarios" de la Figura 42, luego de ello se despliega la venta de la Figura 43 la cual muestra una tabla con todas las cuentas de usuario, al seleccionar una fila de la tabla es posible eliminarla o modificarla.

Figura 43. Botón Gestionar Usuarios

Figura 44. Ventana gestionar usuarios



10.2. CLIENTE CONTROLPC

La aplicación cliente muestra el pantallazo inicial que se muestra en la Figura 44.

Figura 45. Ventana Inicial Cliente

En el pantallazo inicial se ofrecen las siguientes opciones:

- Registrar PC: Esta opción despliega la ventana Registrar PC como se muestra en la Figura 45. Luego el usuario llena la información de registro de un PC y selecciona la opción registrar. A continuación se despliega la ventana Validar Superusuario donde el Superusuario debe ingresar el login y el password. El registro sólo es posible si el Superusuario es válido. Ver Figura 46

- Iniciar Sesión: Esta opción despliega una lista de servidores registrados anteriormente y a los cuales es posible conectarse por medio del comando "Iniciar", eliminarlos con el comando "eliminar" o establecer uno de los PCs como el predeterminado. Ver Figura 47.

- Iniciar Sesión por defecto: Esta opción utiliza aquel PC que se estableció como predeterminado en "Iniciar Sesión" para proveer un acceso directo al PC más utilizado por el usuario.

Figura 46. Registrar PC

Figura 47. Validar Superusuario

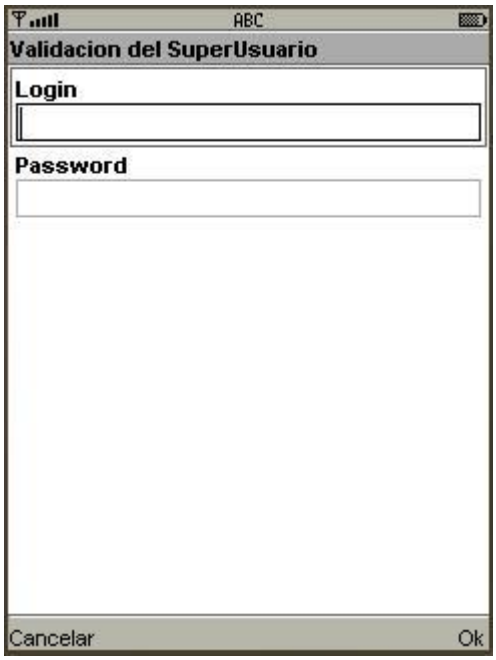


Figura 48. Seleccionar PC

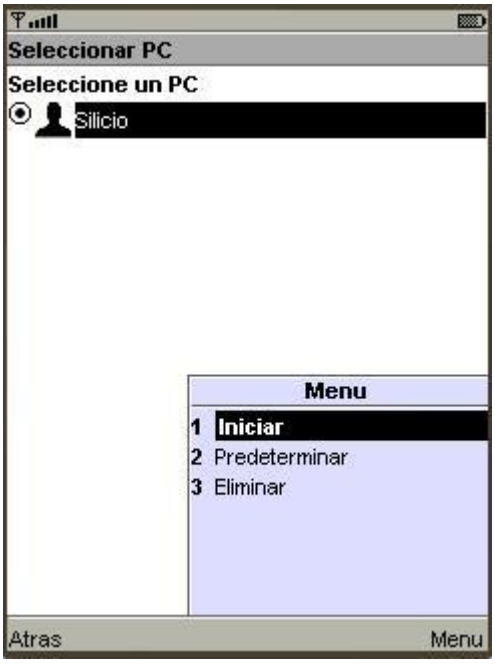
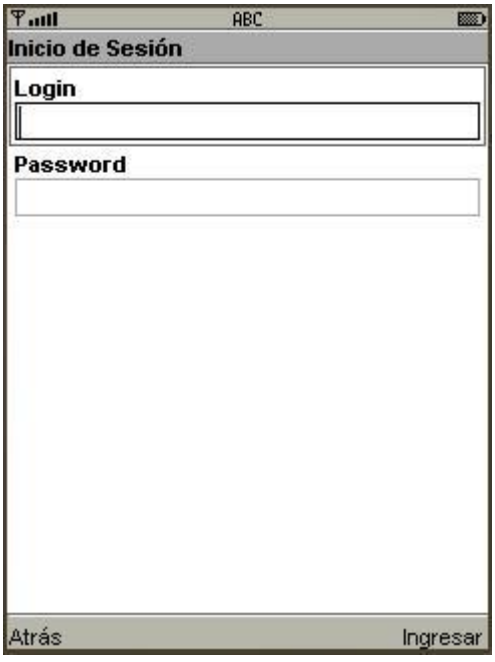


Figura 49. Iniciar Sesión



Una vez se ha iniciado una sesión se muestra la ventana de la Figura 49, en la cual se muestran las diferentes operaciones permitidas sobre el PC servidor.

Figura 50. Ventana de operaciones

El menú de operaciones permite elegir entre las siguientes opciones:

- Buscar: Por medio de un patrón de entrada, la elección del tipo de archivos a buscar entre los cuales se encuentran directorios, ocultos o de solo lectura, y el orden en el cual mostrar la lista de encontrados, ésta es una de las utilidades más interesantes de ControlPC móvil Ver Figura 50. Una vez se presiona el botón Buscar se despliega la ventana de unidades del equipo como se muestra en la Figura 51. Luego de ello se debe elegir uno de los directorios y escoger una carpeta donde comenzar la búsqueda por medio del comando "buscar", los resultados de la búsqueda se visualizan en una nueva pantalla como en la Figura 52, cuyos resultados coinciden con el patrón de la búsqueda j* por lo tanto todos empiezan por "j" y para ayudar al reconocimiento de cada archivo los archivos de extensiones más conocidos son mostrados con iconos alusivos. En esta pantalla es posible utilizar los mismos comandos nombrados en la pantalla "explorar PC", entrar, copiar, detalles, crear carpeta, o cambiar permisos.

Figura 51. Buscar

Figura 52. Lista de unidades

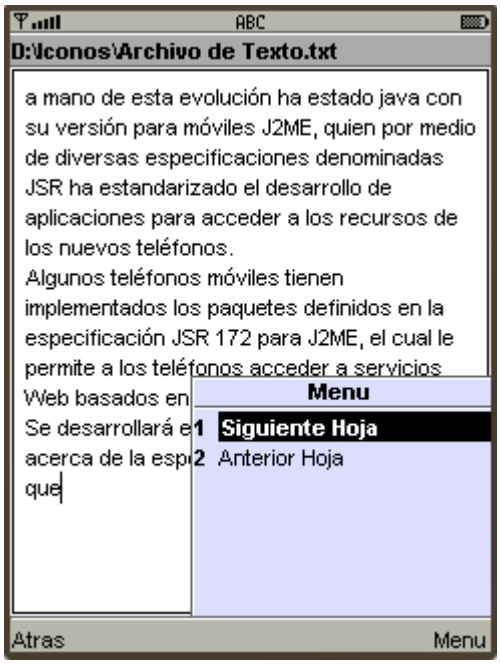
Figura 53. Lista de archivos encontrados

- Explorar PC: Despliega una ventana que permite navegar por el sistema de archivos del PC comenzando por las unidades e inspeccionar las carpetas, visualizar los archivos de texto y hasta visualizar imágenes, la Figura 53 muestra la exploración de la carpeta iconos con los correspondientes posibles comandos a ejecutar para cada archivo o carpeta.

Figura 54. Explorar PC

- Entrar: actúa dependiendo al contexto, si existe una carpeta seleccionada ingresa a ella mostrando sus archivos y carpetas internos. En caso de ser un archivo de texto con extensión txt, java, html, htm, php, sql. etc se muestra una pantalla donde leer el texto de dichos archivos ver Figura 54. Al leer un archivo de texto este aparecerá separado por hojas cortas las cuales es posible ir avanzado por medio del comando “siguiente” o retrocediendo por medio de “anterior”.

Figura 55. Visualizando un archivo de texto

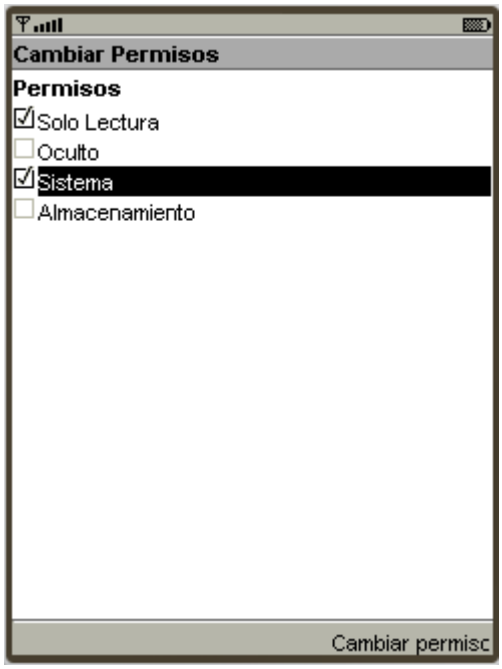


- Ver detalles: Muestra una pantalla con la información de la ruta completa de un archivo, su tamaño completo en bytes y los permisos actuales como en la Figura 55, la cual expone los detalles del archivo D:\postresql.PNG.

Figura 56 Detalles del archivo

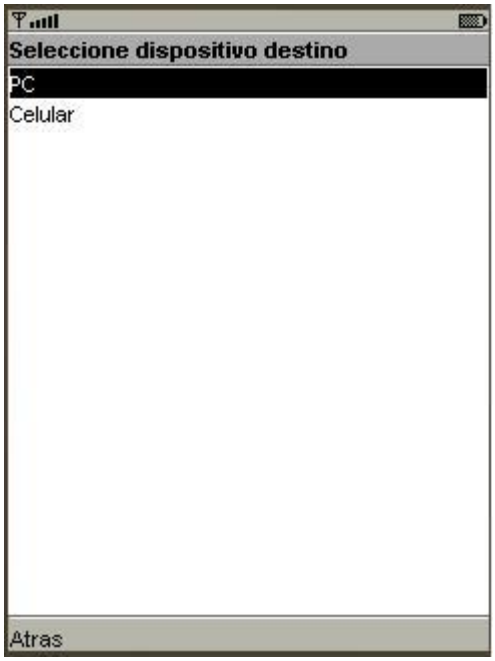
- Cambiar permisos: Muestra una pantalla en la cual es posible modificar los actuales permisos de un archivo o carpeta. Ver Figura 56.

Figura 57. Cambiar Permisos



- Copiar archivo: Una vez se selecciona esta opción aparece una ventana que solicita indicar a que dispositivo se desea copiar el archivo seleccionado el PC o el celular. Como se muestra en la Figura 57.

Figura 58. Seleccionar dispositivo destino



Dependiendo de la selección se muestra el sistema de archivo del dispositivo seleccionado, por medio de lo cual es necesario presionar el comando "Seleccionar destino", si la operación es un éxito se despliega un mensaje de "Operación exitosa".

- Eliminar: Esta opción permite eliminar un archivo seleccionado.
- Crear carpeta: Por medio de esta opción es posible crear una carpeta dentro de la carpeta selecciona.
- Renombrar: Esta opción permite renombrar un archivo o carpeta.

- Explorar celular: Ofrece las opciones copiar, eliminar, mostrar y entrar presentadas al explorar PC solo que ahora las consultas se realizan sobre el sistema de archivos de celular.

- Consola Remota: Esta opción muestra una pantalla que permite la edición de comandos de consola para ser ejecutados en el PC servidor. Ver Figura 58.

Figura 59. Consola Remota

Presenta la facultad de almacenar los últimos comandos ejecutados para luego ser elegidos cuando sean necesarios. Además guarda una copia temporal de la ejecución del último comando para el caso en que el usuario desee revisarla de nuevo. A continuación se muestra un ejemplo con la ejecución del comando Dir. sobre la ruta c:\.

Figura 60. Resultado de la ejecución de un comando Dir.

- Obtener Información del PC: Presenta tres opciones para elegir. Ver Figura 61.

Figura 61. Obtener Información del PC

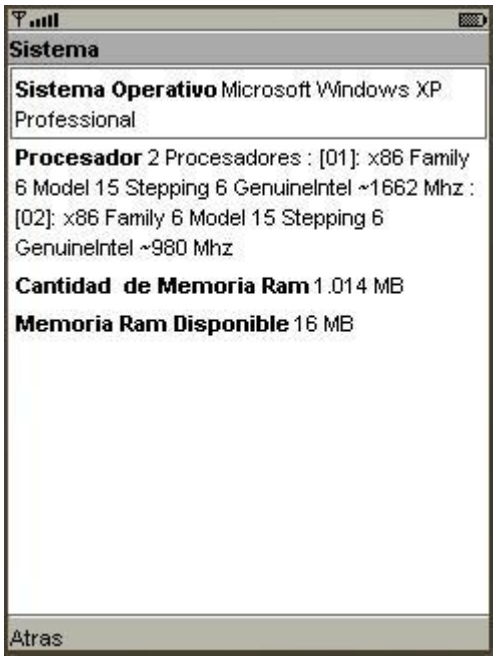
- Información de Red: Muestra en pantalla el nombre del equipo o Hostname, la dirección Ip, la dirección MAC y la puerta de enlace.

- Información de los procesos: Muestra en pantalla la información de los procesos en ejecución del servidor, permitiendo además la terminación de los procesos elegidos por medio del comando "Matar". Ver Figura 61. Cada fila de la ventana representa un proceso, con su respectivo nombre, PID, la memoria consumida, y el estado en el cual se encuentra actualmente.

- Información del sistema: Esta opción despliega en pantalla la información básica del Servidor, como el nombre de su sistema operativo, tipo de procesador, cantidad de memoria RAM, memoria RAM disponible. Ver Figura 62.

Figura 62. Listado de procesos

Figura 63. Información del sistema



- Modificar Password: Esta opción permite la modificación del password de ingreso al sistema solicitando para ello el actual y el nuevo password además la repetición del nuevo password para evitar problemas de digitación. Ver Figura 63.

Figura 64. Modificar password

11. CONCLUSIONES

- Se cumplieron todos los objetivos mencionados en el anteproyecto
- La plataforma J2ME utiliza un subconjunto de la J2SE lo cual hace que la programación de aplicaciones para celulares sea más limitada, en cuanto a interfaces, collections y librerías en general.
- Las aplicaciones para celulares deben considerar las limitaciones de memoria y procesamiento de los dispositivos móviles, para evitar problemas imprevistos.
- El proyecto fue probado satisfactoriamente con los comandos establecidos en los objetivos específicos, incluso puede funcionar con otros comandos con la condición que no bloqueen la consola o ingresen a una nueva línea de comandos como NETSH y WMIC.
- ControlPC permite transferir archivos y carpetas desde el PC al celular y viceversa, con una interfaz agradable de acuerdo a las capacidades de los celulares. El sistema operativo del celular siempre solicita el permiso del usuario para leer o escribir datos en la memoria del celular siendo incómodo para el usuario, la solución que se propone para esto es firmar digitalmente la aplicación del celular para que sea considerada confiable.
- La utilización de Netbeans para el desarrollo de un proyecto de servicios web es recomendable debido a que genera las interfaces necesarias para que el desarrollador solo se tenga que preocupar por la implementación de las reglas del negocio.
- Netbeans no es recomendable para el modelado UML del proyecto porque pueden haber irregularidades al guardar un proyecto tal como no poder recuperar de nuevo el proyecto en el mismo estado en el que se guardó.
- Actualmente consideramos que este proyecto como producto no es viable porque solo es funcional cuando la dirección IP es estática y falta implementar un mecanismo que garantice la seguridad en la comunicación con el PC.
- Gracias al generador de stubs de Netbeans a partir de un documento WSDL el programador se libera de los detalles de la conexión y la creación de los mensajes a enviar al servidor. Solo es necesario una instancia de dicho stub para acceder a todos los

métodos remotos como si estuvieran en la misma aplicación.

- Las aplicaciones J2ME que accedan a servicios web solo pueden funcionar en celulares que tengan implementada la JSR 172.
- Un proyecto que incluya las tecnologías J2ME y J2EE para servicios web no es tan complicado como imaginábamos en un principio y se cuenta con mucha documentación de libre acceso y sin costo alguno.
- Las herramientas Netbeans, Java y Glassfish utilizadas para desarrollar ControlPC son software libre, lo cual permitió desarrollar el proyecto a bajo costo.

GLOSARIO

CLDC: Connected Limited Device Configuration, una configuración provee el conjunto más básico de librerías y características de la máquina virtual que deben estar presentes en cada implementación de un entorno J2ME.

CONSOLA: interfaz de línea de comandos, es un mecanismo para interactuar con el sistema operativo escribiendo comandos para ejecutar taréas específicas.

DEPLOY: montar un servicio web en el servidor de aplicaciones Glassfish.

DOM: Document Object Model, es una interface de plataforma neutral que le permite a los programas y scripts acceder dinámicamente y actualizar el contenido, estructura y el estilo de documentos XML.

DTD: Document Type Definition, es una definición de un documento XML y especifica las restricciones en cuanto a estructura y sintaxis del mismo.

ESPACIO DE NOMBRES: provée un método simple para calificar nombres de elementos y atributos usados en documentos XML al asociarlos con referencias URI.

GLASSFISH: es un servidor de aplicaciones para la plataforma J2EE

HTTP: protocolo de transferencia de hipertexto, define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores, proxies) para comunicarse en la World Wide Web.

J2EE: Java Platform Enterprise Edition se construye sobre la J2SE y es el estándar de la industria para implementar aplicaciones empresariales de arquitectura orientada a servicios y aplicaciones web.

J2ME: Java platform Micro Edition, provée un entorno robusto y flexible para aplicaciones portables sobre muchos dispositivos, aprovechando las capacidades nativas del mismo. Las aplicaciones J2ME pueden funcionar sobre celulares, PDAs, dispositivos embebidos, etc.

J2SE: Java Platform Standard Edition, permite desarrollar y desplegar aplicaciones Java sobre PCs y servidores, como también entornos embebidos y de tiempo real. Java SE incluye clases que soportan el desarrollo de servicios web y provee los fundamentos para la Java EE.

JAVA: lenguaje de programación desarrollado por Sun Microsystems para el desarrollo de aplicaciones independientes de la plataforma.

JAXP: Java Api for XML Processing, permite a las aplicaciones interpretar, transformar, validar y consultar documentos XML usando un API que es independiente de un procesador XML particular.

JAX-RPC: Java API for XML based RPC, permite la creación de aplicaciones que usen XML para hacer llamadas a procedimientos remotos.

JAX-WS: Java Api for Web Services, es una tecnología para construir servicios web y clientes que se comunican usando un protocolo basado en XML como SOAP.

JDK: kit de desarrollo de aplicaciones en lenguaje Java, el cual incluye la máquina virtual, el compilador y herramientas de depuración, entre otras.

JRE: Java Runtime Environment, permite la ejecución de programas hechos en Java. El JRE está constituido por una máquina virtual (JVM) que es el programa que interpreta el código Java y tiene además las librerías de clases estándar que implementan el API de java.

JSR: Java Specification Requests, son las descripciones de especificaciones para la plataforma Java.

JSR 172: especificación que extiende la J2ME para soportar servicios web. Tiene dos paquetes opcionales que estandarizan dos áreas de funcionalidad cruciales para los clientes de servicios web: invocación de servicios remotos e interpretación de documentos XML.

LATENCIA: es la suma de los retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de una red.

MAPEO EXTENSIBLE DE TIPOS: es la correspondencia extendida de tipos XML y tipos Java. Está implementado en el paquete javax.xml.rpc.encoding.

MIDLET: son aplicaciones creadas usando la especificación MIDP y CLDC. Los MIDlets son diseñados para ser ejecutados en dispositivos móviles, los cuales se caracterizan por su poca capacidad de procesamiento y memoria.

MIDP: Mobile Information Device Profile, es el elemento clave de la J2ME. Cuando se combina con CLDC, MIDP suministra un entorno de ejecución estándar para dispositivos móviles como teléfonos celulares y asistentes digitales PDAs. Define una plataforma para desplegar aplicaciones gráficas, optimizadas, conectadas a la red de forma dinámica y segura.

NETBEANS: entorno integrado de desarrollo de código abierto, que permite el desarrollo de aplicaciones de escritorio, empresariales, web y móviles con el lenguaje en Java.

NOMBRE CALIFICADO: consiste en un nombre de elemento o atributo XML que pertenece a un espacio de nombres calificado.

PROXY DINÁMICO: una clase de proxy dinámico implementa una lista de interfaces especificadas en tiempo de ejecución, tal que la invocación de un método a través de una o más interfaces sobre una instancia de la clase será codificada y despachada a otro objeto a través de una interface uniforme.

QNAME: Qualified Name, ver nombre calificado.

RMI: Java Remote Method Invocation, permite al programador crear aplicaciones distribuidas basadas en la tecnología Java, en la cual los métodos de objetos Java remotos pueden ser invocados desde otras máquinas virtuales, posiblemente sobre diferentes hosts.

RPC: Remote Procedure Call, es una tecnología que permite a un programa de computador ejecutar procedimientos en otro espacio de direcciones (comúnmente sobre otro computador en otra red compartida) sin que el programador tenga que codificar explícitamente los detalles para su interacción remota, debe escribir el mismo código como si el procedimiento estuviera en la misma máquina.

RUNTIME JAX-RPC: forma el núcleo de una implementación JAX-RPC, es una librería del lado del cliente que provee un conjunto de servicios requeridos para los mecanismos de ejecución del subconjunto JAX-RPC. El runtime oculta las complejidades asociadas con la administración de la conexión, SOAP y la codificación de datos.

SAX: Simple Api for XML, api para interpretar secuencialmente documentos XML. Fue implementado originalmente por Java y ahora es soportado por la mayoría de lenguajes de programación.

SCHEMA: permite definir la estructura, el contenido y la semántica de documentos XML.

SERVICIO WEB: cconjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

SOAP: Simple Object Access Protocol, es un protocolo para el intercambio de mensajes basados en XML en redes de computadores, usando normalmente HTTP/HTTPS.

STUB: es una clase Java generada a partir de un documento WSDL y permite acceder a los métodos remotos del servicio web descrito por el WSDL. La aplicación cliente de un servicio web interactúa con el runtime a través de un stub.

UML: Lenguaje Unificado de Modelado, es la especificación de la OMG más usada para modelar la estructura, comportamiento y arquitectura de aplicaciones, también se usa para modelar los procesos de negocio y la estructura de datos.

UTF-8: Unicode Transformation Format-8, es la codificación por defecto para XML y codifica cada carácter Unicode con un número variable de octetos usando entre 1 y 4 octetos.

W3C: el Consorcio World Wide Web (W3C) es un consorcio internacional donde las organizaciones miembro, personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web. La misión del W3C es guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.

WAP: Wireless Application Protocol o WAP (protocolo de aplicaciones inalámbricas) es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, p.ej. acceso a servicios de Internet desde un teléfono móvil.

Se trata de la especificación de un entorno de aplicación y de un conjunto de protocolos de comunicaciones para normalizar el modo en que los dispositivos inalámbricos, se pueden utilizar para acceder a correo electrónico, grupo de noticias y otros.

WIRELESS: la comunicación inalámbrica es la transferencia de información a distancia sin el uso de conductores eléctricos.

WSDL: Web Services Description Language, es un formato XML que se utiliza para describir Servicios Web.

WS-I BP: Web Services Interoperability Organization Basic Profile, consiste de un conjunto de especificaciones de servicios web para promover la interoperabilidad independiente de la tecnología.

XML: Extensible Markup Language, es un formato de texto simple y muy flexible derivado del SGML. El lenguaje de etiquetado extensible es un conjunto de reglas para definir etiquetas semánticas que dividen un documento en partes e identifica las diferentes partes del documento. Es un metalenguaje de etiquetado que define una sintaxis en el cual se pueden escribir otros lenguajes de etiquetado.

BIBLIOGRAFIA

CHAPPELL David y JEWELL Tyler. Java Web Services. O'Reilly, 276 p.

Dan Pilone y Neil Pitman. UML 2.0 in a Nutshell. O'Reilly,Junio del 2005. 234 p.

ELLIS Jon y YOUNG Mark. J2ME Web Services JSR 172. http://jcp.org/en/jsr/detail?id=172. 74 p.

ENGLANDER, Robert. Java and SOAP. O'Reilly, Mayo 2002. 276 p.

JDK 6 Documentation. http://java.sun.com/javase/6/download.jsp#docs. Sun Microsystems. 2006.

Kim Hamilton y Russell Miles. Learning UML 2.0. O'Reilly, Abril del 2006. 286 p.

RUSTY, Elliotte Harold. XML 1.1 Bible, 3rd Edition. Wiley Publishing, Inc. Indianapolis, Indiana 2004. 1057 p.

Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/. World Wide Web Consortium (W3C). 08 de Mayo del 2000.

The Java Tutorial. http://java.sun.com/docs/books/tutorial. Sun Microsystems. Agosto 1 del 2007.

WEITZENFELD, Alfredo. Ingeniería de Software Orientada a Objetos con UML, Java e Internet. Internacional Thomson Editores, México 2005. 678 p.

XML Schema Tutorial. http://www.w3schools.com/Schema/default.asp.

ANEXO A. INGENIERIA DE SOFTWARE

1. ACTORES DEL PROYECTO

- **Superusuario**

  Es el encargado de administrar la aplicación servidor y puede realizar tareas como agregar, modificar, eliminar y consultar usuarios. Además puede cerrar conexiones, parar el servicio, y registrar PCs en el celular. El Superusuario interactúa con los siguientes casos de uso:

  - Gestionar Usuarios
  - Gestionar Superusuario
  - Agregar Usuario
  - Gestionar Servidor
  - Cerrar Sesión
  - Iniciar Servicio

- **Usuario**

  Es el actor principal y representa a la persona usuaria del celular en el cual se ha instalado la aplicación cliente. El Usuario está involucrado en los siguientes casos de uso:

  - Agregar Usuario
  - Cerrar Sesión
  - Modificar Password
  - Iniciar Sesión
  - Ejecutar Operación

- Obtener Información del PC
- Ejecutar Comando Shell
- Consultar Archivos y Directorios
- Buscar

- **Registro PCs**

  Es un actor secundario encargado de las siguientes actividades

  - Adicionar registros de nuevos PCs
  - Eliminar registros de PCs
  - Modificar registros de PCs

  Este actor participa en los siguientes casos de uso:
  - Agregar Usuario
  - Iniciar Sesión

- **Registro Superusuario**

  Es un actor secundario y se encarga de las siguientes actividades:

  - Adicionar Superusuarios
  - Eliminar Superusuarios
  - Modificar Superusuarios

  Este actor participa en el caso de uso gestionar superusuario.

- **Registro usuarios**

  Es un actor secundario encargado de las siguientes actividades

  - Adicionar registros de nuevos Usuarios
  - Eliminar registros de Usuarios
  - Modificar registros de Usuarios

  Los registros están almacenados en cada PC Servidor. Este actor participa en los casos de uso:

  - Gestionar Usuarios
  - Agregar Usuario

- Modificar Password

- **Recursos Celular**

  Es un actor secundario y está encargado de leer y escribir archivos en el celular. Está involucrado en los siguientes casos de uso:

  - Obtener Información del PC
  - Ejecutar Comando Shell
  - Consultar Archivos y Directorios
  - Buscar

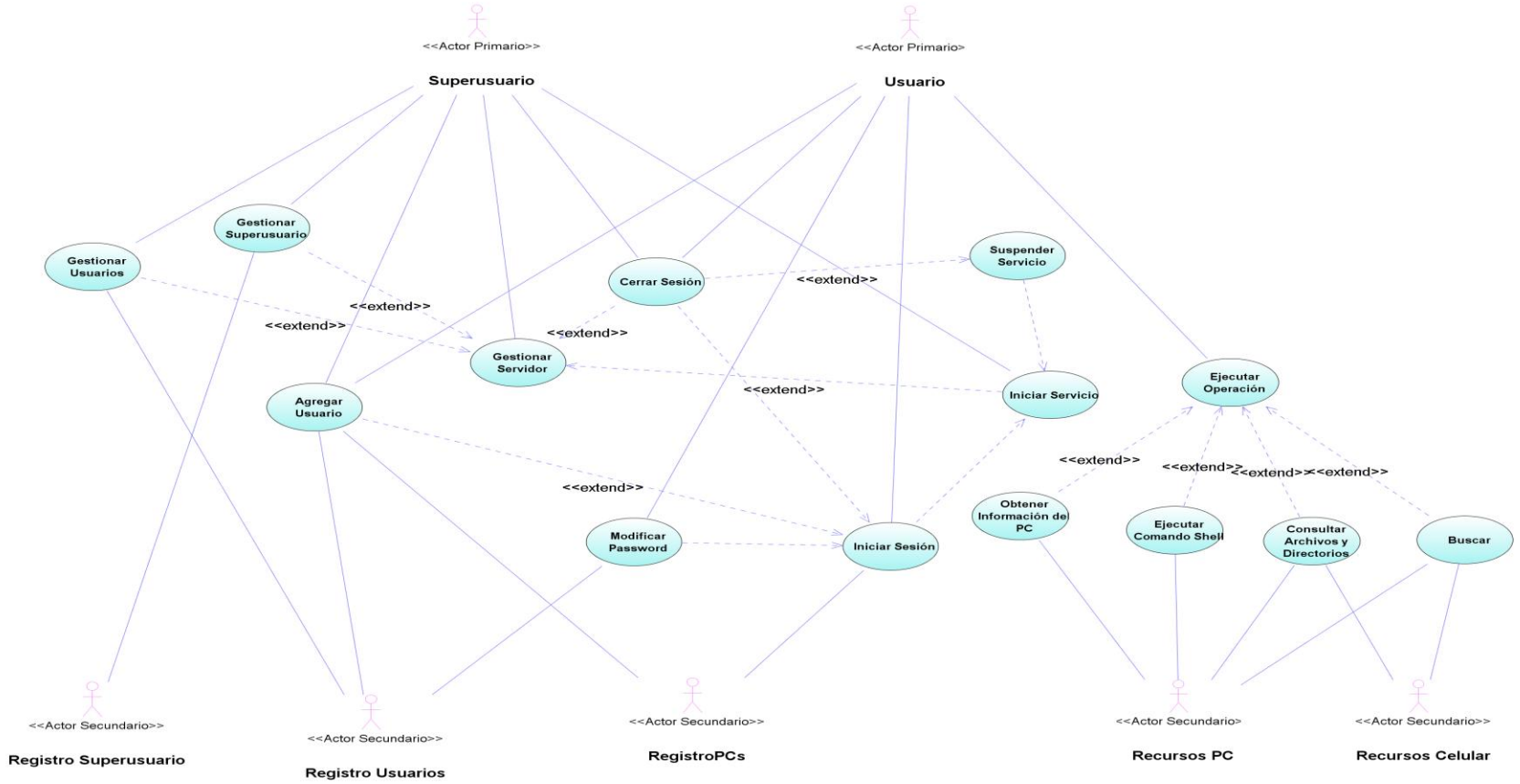
- **Recursos PC**

  Es un actor secundario que interactúa con el sistema cuando se le solicita la lectura y escritura de archivos en el PC, y la ejecución de comandos de consola. Está involucrado en los siguientes casos de uso:

  - Consultar Archivos y Directorios
  - Buscar

2. DIAGRAMA COMPLETO DE LOS CASOS DE USO

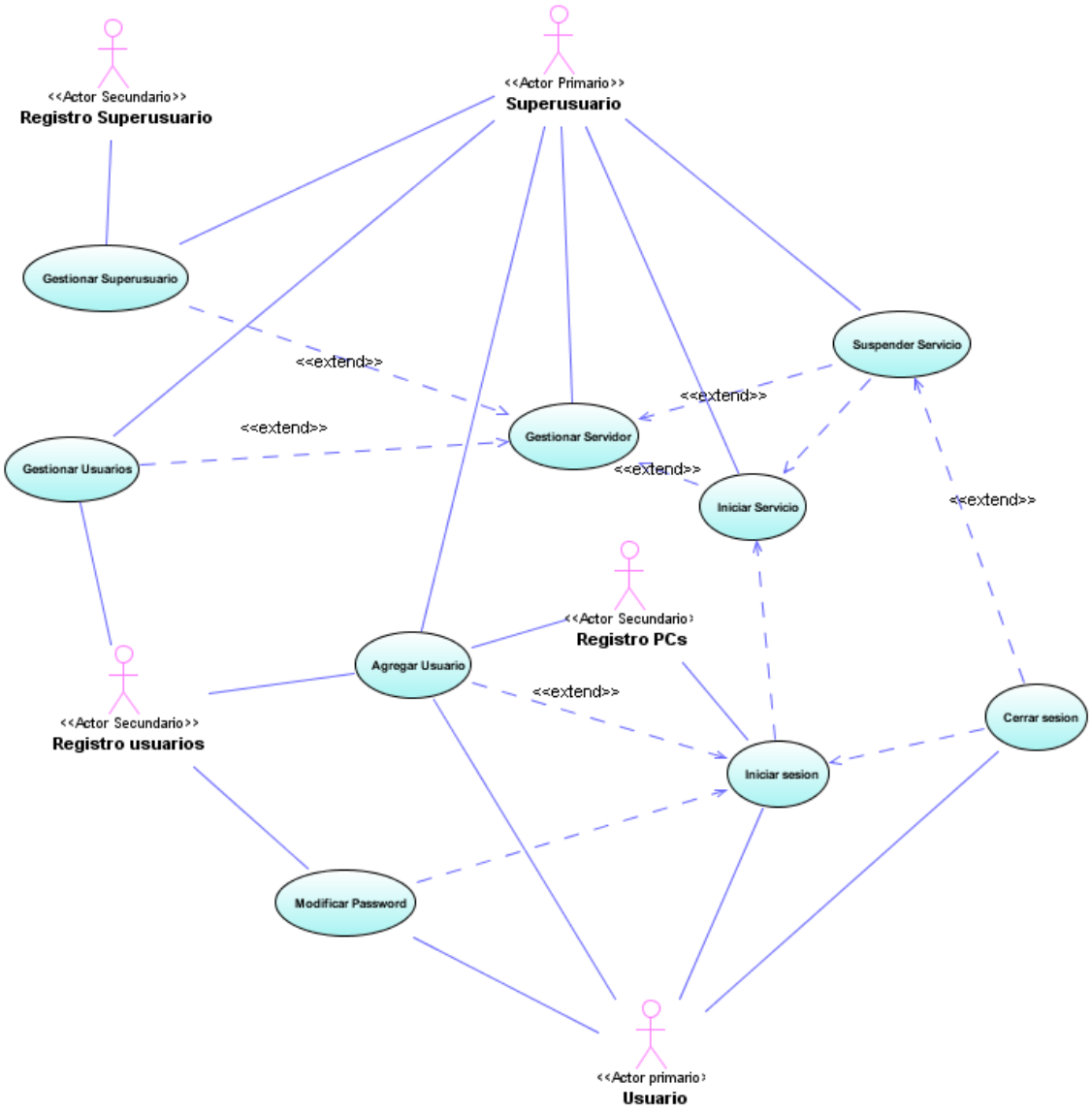
Figura 65. Diagrama completo de los casos de uso



2.1. *DIAGRAMA DE CASOS DE CONFIGURACIÓN*

El siguiente diagrama de casos de uso muestra la interacción entre actores y casos de uso relacionados con la configuración del sistema.

Figura 66. Diagrama de casos de Configuración



- ✓ **Iniciar sesión**

Propósito

Iniciar una nueva sesión con el servidor

Resumen

El usuario inicia este caso de uso. Ofrece la posibilidad de empezar a dar órdenes remotas a través del celular.

Precondiciones

Se requiere cargar la aplicación cliente. Haberse ejecutado el caso de uso "Iniciar servicio" E-4

Flujo Principal

Se despliega la ventana “M0_VentanaPrincipal” la cual presenta las opciones de “iniciar sesión por defecto”, “Registrar PC” e “Iniciar sesión”

Si el usuario selecciona la opción “iniciar sesión por defecto” se continúa con el subflujo S-2 usando el PC por defecto.

Si el usuario selecciona la opción “Registrar PC” se ejecuta el caso de uso "Agregar Usuario"

Si el usuario selecciona la opción “Iniciar sesión” E-1 se continúa con el subflujo S-1

Subflujos

S-1 Se despliega la ventana “M0_2_SeleccionarPC”, la cual contiene una lista con los PCs registrados en “Registro PCs” donde el usuario puede seleccionar un PC y debe elegir una de las siguientes opciones:

Iniciar: se continúa con el subflujo S-2

Eliminar: Se continúa con el subflujo S-3

Predeterminar: El PC seleccionado será el PC por defecto usado al “iniciar sesión por defecto”

S-2 Se despliega la ventana “M1_PantallaDeInicioSesión”, donde se recibe el login y el password del usuario actual, a continuación el usuario debe presionar “Ingresar” para verificar si tiene permiso para usar el sistema E-2 E-3 E-5.

S-3 Se despliega la ventana "M1_PantallaDeInicioSesión" para validar el usuario y luego el usuario debe presionar el botón “Eliminar” para eliminar el PC de “Registro PCs”

Excepciones

E-1 No hay PCs registrados, se despliega un mensaje informativo con el mensaje “No hay PCs registrados”.

E-2 Si E-3 ocurre 3 veces entonces se despliega el mensaje “Número de intentos de inicio superado” y se cierra la aplicación cliente

E-3 Usuario inválido, se despliega una ventana de error con el mensaje "El Usuario no existe" y se le dan dos oportunidades más para ingresar la información de validación correctamente.

E-4 No se ha iniciado el servicio, se despliega una ventana de error con el mensaje "No se ha iniciado el servicio"

E-5 El usuario no se encuentra registrado en el servidor

- ✓ **Cerrar sesión**

Propósito

Cerrar la sesión del usuario actual.

Precondiciones

Se requiere haber ejecutado el caso de uso "Iniciar Sesión" y el caso de uso "Iniciar Servicio".

Flujo Principal

Una vez iniciado este caso de uso, si existe una orden remota en ejecución se espera a su correcta finalización, luego se liberan los recursos correspondientes a la sesión

Este caso de uso se puede iniciar debido a las siguientes causas:

El Superusuario presionó el botón “Suspender Servicio” de la “W0_Ventana Principal”, se despliega el mensaje “El Superusuario ha suspendido el servicio” en el celular.

El Superusuario presionó el botón “Cerrar Sesión” de la “W0_Ventana Principal”, se despliega el mensaje “El Superusuario ha cerrado la conexión con el servidor” en el celular.

El Usuario presionó el botón “Salir” E-1 de la “M0_VentanaPrincipal" no se despliega algún mensaje.

El Superusuario presionó el botón “Salir” de la “W0_Ventana Principal”, se despliega el mensaje “El Superusuario ha cerrado la conexión con el servidor” en el celular.

Excepciones

E-1 Problemas de conexión, se despliega un mensaje informando el problema al usuario actual.

✓ Modificar Password

Propósito

Permitirle al usuario actual modificar su password de inicio de sesión en Registro Usuarios.

Precondiciones

El usuario inicia este caso de uso. Se requiere haber ejecutado el caso de uso "Iniciar Sesión".

Flujo Principal

El usuario presiona el botón “Modificar password” en la ventana”M2_Operaciones” y se despliega la ventana “M2_4_ModificarPassword” donde el usuario debe llenar los campos “Password” y “Repetir Password” y presionar el botón “OK” E-1 para efectuar la modificación del password en “Registro Usuarios” E-2.

Excepciones

E-1 Los campos “Password” y “Repetir Password” no coinciden, se muestra un mensaje de error.

E-2 Problemas de conexión, se despliega un mensaje informando el problema al usuario actual.

✓ Iniciar Servicio

Propósito

Permitirle al servidor recibir peticiones de un cliente.

Precondiciones

Se requiere haber ejecutado el caso de uso "Gestionar Servidor", El Superusuario debe haber presionando el botón Iniciar Servicio de la “W0_Ventana Principal”

Flujo Principal

El Superusuario presiona el botón "Iniciar servicio" de la “W0_Ventana Principal”. Luego este botón cambia su etiqueta a "Suspender servicio" y el servidor inicia el servicio web.

- ✓ **Suspender Servicio**

Propósito

Impedirle al servidor recibir peticiones de un cliente.

Precondiciones

Se requiere haber ejecutado el caso de uso "Gestionar Servidor" y el caso de uso "Iniciar Servicio", El Superusuario debe haber presionando el botón “Suspender Servicio” de la “W0_Ventana Principal”

Flujo Principal

El Superusuario presiona el botón "Suspender servicio" de la “W0_Ventana Principal”. Luego este botón cambia su etiqueta a "Iniciar servicio", se ejecuta el caso de uso cerrar sesión, y se detiene el servicio web.

- ✓ **Gestionar Usuarios**

Propósito

Permitir Gestionar la información de todos los usuarios, almacenada en "Registro Usuarios" en la aplicación servidor.

Resumen

El Superusuario inicia este caso de uso. Ofrece la funcionalidad de modificar y eliminar usuarios.

Precondiciones

Se requiere haber ejecutado el caso de uso "Validar Superusuario", el Superusuario debe haber presionado el botón "Gestionar Usuarios" de la “W0_Ventana Principal”.

Flujo Principal

Se despliega “W6_Ventana Gestionar Usuarios”, la cual contiene los botones “Eliminar” y “Modificar” y además una tabla con la información de todos los usuarios.

Si el Superusuario presiona el botón "Eliminar" se continúa con el subflujo eliminar usuarios S1.

Si el Superusuario presiona el botón "Modificar" se continúa con el subflujo modificar usuarios S2.

Si el Superusuario presiona el botón "Cerrar" se cierra " W6_Ventana Gestionar Usuarios”.

Subflujos

S1. Eliminar Usuarios: El Superusuario selecciona una fila de la tabla y presiona el botón eliminar E-5

Se despliega el siguiente mensaje "Desea eliminar los # usuarios seleccionados"; donde # es la cantidad de filas seleccionadas; con dos botones "Si" y "No". Si se presiona "Si", todas las filas que se hayan seleccionado en la tabla serán eliminadas de "Registro Usuarios" y se actualizará la tabla de la “W6_Ventana Gestionar Usuarios”.

S2. Modificar Usuarios: El Superusuario selecciona una fila de la tabla y presiona el botón modificar E-5 E-6, entonces se despliega la ventana "W2_Modificar Usuarios", donde el Superusuario podrá seleccionar los Checkbox para modificar ese campo del registro y se habilita el cuadro de texto correspondiente para digitar los nuevos datos. Se puede elegir modificar el Login, el Password y el IMEI del celular. Si se presiona el botón modificar E-1 E-2 E-3 E-4 se actualizará el registro correspondiente en "Registro Usuarios" y se actualizará la tabla de la ventana W6.

Excepciones

E-1 IMEI incorrecto, el IMEI debe estar conformado por un número de 15 dígitos, se despliega un mensaje de error "IMEI incorrecto".

E-2 Password Inseguro, el password tiene una longitud menor a 8 caracteres, se despliega un mensaje de error "Password inseguro, el password debe tener una longitud mayor o igual a 8 caracteres".

E-3 No coinciden Password y repetir Password, se despliega un mensaje de error "No coinciden password y repetir password"

E-4 Login repetido, se despliega un mensaje de error "Login Repetido"

E-5 No hay filas seleccionadas, se despliega una ventana con el mensaje "Debe seleccionar por lo menos una fila para efectuar la operación"

E-6 Hay más de una fila seleccionada, se despliega una ventana con el mensaje "Debe seleccionar una y solo una fila para efectuar la operación modificar"

- ✓ **Gestionar Servidor**

Propósito

Verificar si un Superusuario tiene permiso para entrar al sistema y permitirle realizar una operación de la “W0_Ventana Principal”.

Resumen

El Superusuario inicia este caso de uso. Ofrece la funcionalidad de recibir la información de validación de Superusuario, comprobar si está registrado en "Registro Superusuario" y permitirle acceder a las opciones del Superusuario

Precondiciones

Se requiere cargar la aplicación servidor.

Flujo Principal
Busca en "Registro Superusuario" si existe algún Superusuario registrado, en caso de no haberlo se continúa con el subflujo S-1.

Se despliega la ventana "W5_Validar Superusuario", encargada de recibir la información de validación, en este caso el login y el password del Superusuario, luego el Superusuario debe presionar entrar y se compara la información recibida con la información almacenada en Registro Superusuario.

Si la comparación E-1, E-2 retorna que ambas son iguales entonces se despliega la “W0_Ventana Principal”, en la cual se presentan las opciones: “Gestionar Usuarios”, “Modificar Superusuario”, “Cerrar Sesión”, “Iniciar Servicio”, “ayuda” y “Salir”.

Si el Superusuario presiona “Modificar Superusuario” se ejecuta el caso de uso “Gestionar Superusuario”.

Si el Superusuario presiona “Gestionar Usuarios” se ejecuta el caso de uso “Gestionar Usuarios”

Si el Superusuario presiona “Cerrar Sesión” se ejecuta el caso de uso “Cerrar Sesión”

Si el Superusuario presiona “Suspender Servicio” se ejecuta el caso de uso “Suspender Servicio”

Si el Superusuario presiona “Iniciar servicio” se ejecuta el caso de uso “Iniciar Servicio”

Subflujos

S-1 Se despliega la ventana "W3_Registrar Superusuario" que contiene información del registro del Superusuario, lo cual incluye login, password y la entrada adicional repetir password para asegurarse de la correcta escritura del password. Esta información se utilizará para validar al Superusuario cada vez que intente cargar la aplicación servidor.

Si el usuario selecciona "Registrar", el sistema guarda la información digitada en "Registro Superusuario" E-3, E-4

Si el usuario selecciona "Cancelar", la información digitada se perderá y se cerrará la ventana "W3_Registrar Superusuario".

Excepciones

E-1 Si E-2 ocurre 3 veces entonces se despliega el mensaje “Número de intentos de inicio superado” y se cierra la aplicación servidor.

E-2 Superusuario inválido, se despliega una ventana de error con el mensaje "El Superusuario no existe" y se le dan dos oportunidades más para ingresar la información de validación correctamente.

E-3 Password Inseguro, el password tiene una longitud menor a 8 caracteres

E-4 No coinciden Password y repetir Password

- ✓ **Modificar Superusuario**

Propósito
Permitir modificar el registro del Superusuario en la aplicación servidor.

Resumen
El Superusuario inicia este caso de uso. Ofrece la funcionalidad de recibir el password actual, el nuevo login y el nuevo password para luego modificar el "Registro Superusuario".
Precondiciones
Se requiere haber ejecutado anteriormente los casos de uso "Gestionar Servidor" y haber presionado el botón "Modificar Superusuario" de la “W0_Ventana Principal”.
Flujo Principal

Se despliega la ventana "W4_Modificar Superusuario" de la cual se obtienen los siguientes datos: nuevo login, nuevo password y la entrada adicional repetir nuevo password para asegurarse de la correcta escritura del password. Esta información se utilizará para validar al Superusuario cada vez que intente cargar la aplicación servidor.

Si el usuario selecciona "Modificar", el sistema actualiza el "Registro Superusuario"

Si el usuario selecciona "Cancelar", la información digitada se perderá y se cerrará la ventana "W4_ModificarSuperusuario".
Excepciones
E-1 Password Inseguro, el password tiene una longitud menor a 8 caracteres
E-2 No coinciden Password y repetir Password.

- ✓ **Agregar Usuario**

Propósito

Permitirle al Superusuario registrar un PC en el celular y agregar un nuevo usuario en el PC servidor

Precondiciones

El Superusuario y el usuario inician este caso de uso

Flujo Principal

En el celular, se despliega la ventana “M0_1_2_1_RegistrarPC” en la cual debe ingresar el “Nombre del PC”, “Dirección IP”, el “Puerto”, “Login Usuario”, “Password Usuario” y “Repetir Password”, los tres primeros campos son los datos del PC servidor y los tres últimos campos se refieren a la cuenta de usuario en el PC servidor, luego se debe presionar el botón “Registrar” E-1 E-2 E-4 y se despliega la ventana “M0_1_2_2_ValidarSuperusuario”, donde se

debe ingresar el “Login” y el “Password” del Superusuario E-3. Se crea un registro en “Registro PCs” con el nombre del PC, el puerto y la dirección IP digitadas y se crea un registro en “Registro Usuarios” con el login y el password.

Excepciones

E-1 Nombre de PC repetido, se despliega un mensaje solicitando ingresar otro nombre

E-2 Ip inválida o Puerto inválido se despliega un mensaje solicitando ingresar estos datos correctamente.

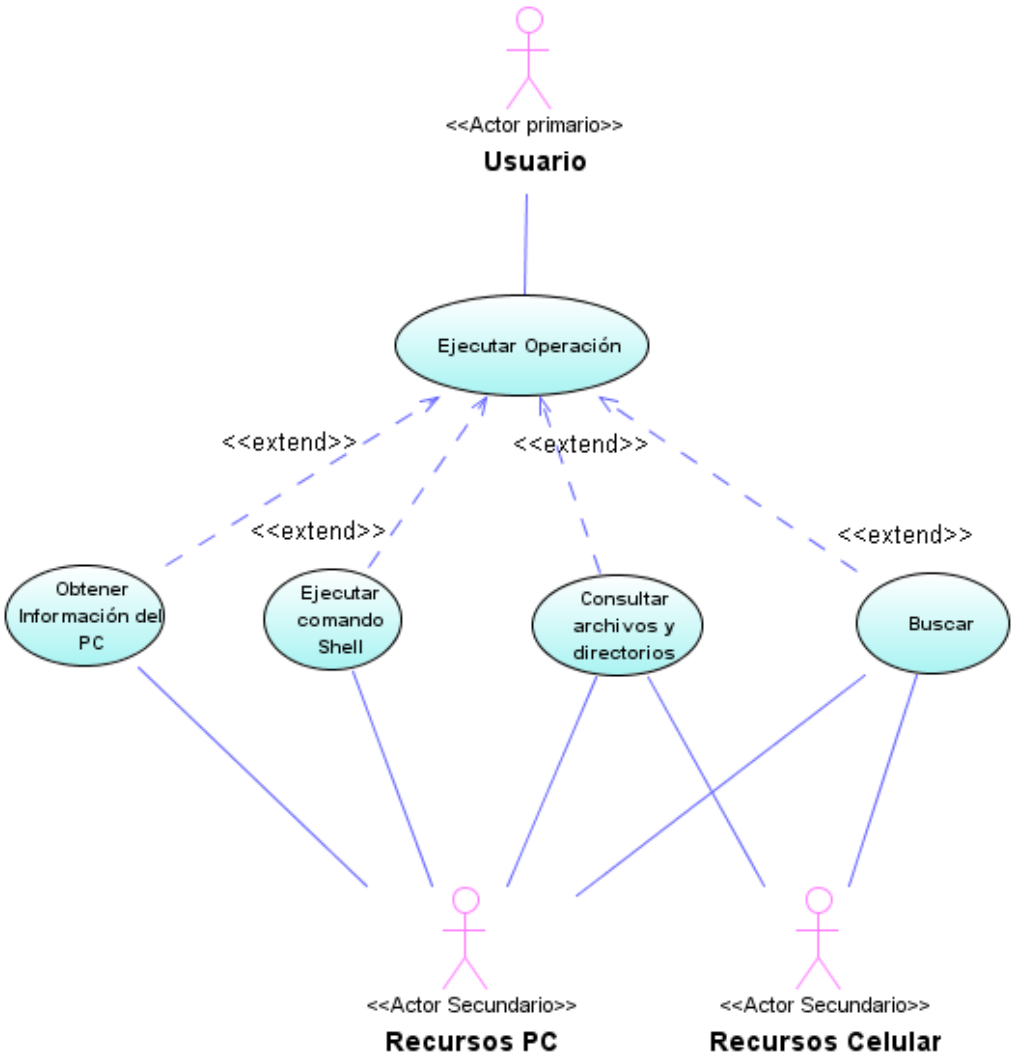
E-3 Información del Superusuario incorrecta. El Login y Password no están registrados en el PC. Se despliega un mensaje informativo.

E-4 Password y Repetir password no coinciden

2.2. DIAGRAMA DE CASOS DE USO OPERACIONES

El siguiente diagrama de casos de uso muestra la interacción entre el actor usuario y los casos de uso relacionados con las operaciones que este posible hacer. Debido a que los casos de uso “Obtener Información del PC”, “Ejecutar comando Shell”, “Consultar archivos y directorios” y “Buscar” son operaciones estas extienden del caso de uso “Ejecutar Operación”, el cual agrupa las funcionalidades comunes del resto.

Figura 67. Diagrama de casos de uso operaciones



- ✓ **Ejecutar Operación**

Propósito

Permitirle al usuario la ejecución de una operación remota a través del celular.

Resumen

El usuario inicia este caso de uso cuando selecciona la opción “ingresar” de la ventana “M1_PantallaDeInicioSesion” en el caso de uso “Iniciar Sesión”. Le ofrece acceso al servicio web.

Precondiciones

Se debió haber ejecutado anteriormente el caso de uso “Iniciar Sesión”

Flujo Principal

Se despliega la ventana “M2_Operaciones” donde el usuario puede elegir realizar las siguientes operaciones: “Buscar Archivo”, “Explorar PC”, “Consola Remota” y “Obtener Información PC” y adicionalmente “Cerrar Sesión”

Si el usuario elige “Buscar Archivo”, se ejecuta el caso de uso “Buscar Archivo”.

Si el usuario elige “Explorar PC”, se ejecuta el caso de uso “Consultar Archivos y Directorios”.

Si el usuario elige “Consola Remota”, se ejecuta el caso de uso “Ejecutar Comando Shell”.

Si el usuario elige “Obtener Información PC”, se ejecuta el caso de uso “Obtener Información PC”.

Si el usuario elige “Cerrar Sesión”, se ejecuta el caso de uso “Cerrar Sesión”.

- ✓ **Ejecutar comando Shell**

Propósito

Permitirle al usuario ejecutar comandos remotos.

Resumen

El usuario inicia este caso de uso seleccionando la opción “Consola Remota” en el caso de uso “Ejecutar Operación”

Precondiciones

Se debe haber ejecutado previamente el caso de uso “Ejecutar Operación”

Flujo Principal

Se despliega la ventana “M2_3_ConsolaRemota” en la cual el usuario puede ingresar el comando Shell, el cual desea ejecutar en el PC servidor. Después de ingresar el comando el usuario debe elegir “Ejecutar” E-1, el comando se ejecuta en el servidor haciendo uso en algunos casos de “Recursos PC” y luego se despliega la ventana “M2_3_1_ResultadosComando” con los resultados de la ejecución del comando.

El usuario también puede elegir ver los resultados de la ejecución del último comando ingresado por medio de la opción denominada “Últimos Resultados”.

Excepciones

E-1 No es posible establecer conexión con el servidor, se despliega un mensaje informando el error ocurrido.

✓ Consultar archivos y directorios del Celular

Propósito

Permitirle al usuario explorar el sistema de archivos del celular y realizar operaciones sobre algún archivo o carpeta seleccionada.

Resumen

El usuario inicia este caso de uso seleccionando la opción “Explorar Celular” en el caso de uso “Ejecutar Operación”

Precondiciones

Se debe haber ejecutado previamente el caso de uso “Ejecutar Operación”

Flujo Principal

Se despliega la ventana “M2_5_ExplorarCelular”, donde el usuario puede navegar por el sistema de archivos del Celular y seleccionar una carpeta o seleccionar un archivo, para esto se hace uso de “Recursos Celular”.

El usuario puede elegir una de las siguientes opciones:

- Mover: Se despliega la ventana “M3_3_SeleccionarDispositivoDestino” donde el usuario debe elegir el dispositivo destino (PC o Celular) donde se alojará la carpeta seleccionada. Si el usuario seleccionó PC, se despliega la ventana “M3_3_1_SeleccionarCarpetaDestinoPC”. Si el usuario seleccionó Celular, se despliega

la ventana “M3_3_2_SeleccionarCarpetaDestinoCelular” y se hace uso de Recursos Celular.

Luego el usuario puede navegar por el sistema de archivos del dispositivo destino y elegir la carpeta destino. Se muestra la ventana “M4_IngresarNombreDestino” para ingresar el nombre de la carpeta seleccionada en la carpeta destino. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Copiar: Se despliega la ventana “M3_3_SeleccionarDispositivoDestino” donde el usuario debe elegir el dispositivo destino (PC o Celular) donde se alojará la carpeta seleccionada. Si el usuario seleccionó PC, se despliega la ventana “M3_3_1_SeleccionarCarpetaDestinoPC”. Si el usuario seleccionó Celular, se hace uso de Recursos Celular, se despliega la ventana “M3_3_2_SeleccionarCarpetaDestinoCelular”. Luego el usuario puede navegar por el sistema de archivos del dispositivo destino y elegir la carpeta destino. Se muestra la ventana “M4_IngresarNombreDestino” para ingresar el nombre de la carpeta seleccionada en la carpeta destino. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

✓ Consultar archivos y directorios PC

Propósito

Permitirle al usuario explorar el sistema de archivos del PC y realizar operaciones sobre algún archivo o carpeta seleccionada.

Resumen

El usuario inicia este caso de uso seleccionando la opción “Explorar PC” en el caso de uso “Ejecutar Operación”

Precondiciones

Se debe haber ejecutado previamente el caso de uso “Ejecutar Operación”

Flujo Principal

Se despliega la ventana “M2_2_ExplorarPC”, donde el usuario puede navegar por el sistema de archivos del PC y seleccionar una carpeta o seleccionar un archivo, para esto se hace uso de “Recursos PC”.

Si el usuario selecciona una carpeta, se continúa con el subflujo S-1

Si el usuario selecciona un archivo, se continúa con el subflujo S-2

Subflujos

S-1 Se despliega la ventana “M2_2_2_OperacionesCarpeta”, la cual le permite elegir al usuario una las siguientes opciones:

- Renombrar: Se despliega la ventana “M3_1_Renombrar” en donde el usuario debe ingresar el nuevo nombre de la carpeta y luego seleccionar la opción “Renombrar” para cambiarle el nombre a la carpeta seleccionada, luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Crear: Se despliega la ventana “M3_2_CrearCarpeta” donde el usuario puede crear una subcarpeta de la carpeta seleccionada. Para esto debe ingresar el nombre de la nueva carpeta y seleccionar la opción “Crear Carpeta”. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Eliminar: El usuario puede eliminar la carpeta seleccionada con todo su contenido, aparece una ventana de confirmación de eliminación, si el usuario selecciona “OK”, se ejecuta la eliminación. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Mover: Se despliega la ventana “M3_3_SeleccionarDispositivoDestino” donde el usuario debe elegir el dispositivo destino (PC o Celular) donde se alojará la carpeta seleccionada. Si el usuario seleccionó PC, se despliega la ventana “M3_3_1_SeleccionarCarpetaDestinoPC”. Si el usuario seleccionó Celular, se despliega la ventana “M3_3_2_SeleccionarCarpetaDestinoCelular” y se hace uso de Recursos Celular.

  Luego el usuario puede navegar por el sistema de archivos del dispositivo destino y elegir la carpeta destino. Se muestra la ventana “M4_IngresarNombreDestino” para ingresar el nombre de la carpeta seleccionada en la carpeta destino. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Cambiar Permisos: Se despliega la ventana “M3_4_CambiarPermisos” donde el usuario puede elegir los nuevos permisos de acceso a la carpeta seleccionada. Dichos permisos son: “Solo lectura”, “Sistema”, “Oculto” y “Almacenamiento”. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Copiar: Se despliega la ventana “M3_3_SeleccionarDispositivoDestino” donde el usuario debe elegir el dispositivo destino (PC o Celular) donde se alojará la carpeta seleccionada. Si el usuario seleccionó PC, se despliega la ventana “M3_3_1_SeleccionarCarpetaDestinoPC”. Si el usuario seleccionó Celular”, se hace uso de Recursos Celular, se despliega la ventana

"M3_3_2_SeleccionarCarpetaDestinoCelular". Luego el usuario puede navegar por el sistema de archivos del dispositivo destino y elegir la carpeta destino. Se muestra la ventana "M4_IngresarNombreDestino" para ingresar el nombre de la carpeta seleccionada en la carpeta destino. Luego se despliega la ventana "M3_0_ResultadoOperacion".

- Copiar Ruta: Guarda temporalmente la ruta de la carpeta seleccionada, con el fin de usarla luego en la consola remota.

S-2 Se despliega la ventana "M2_2_1_Operacionesdearchivo", la cual le ofrece al usuario las mismas opciones de "M2_2_2_OperacionesCarpeta" a excepción de "Crear", todas ellas con el mismo funcionamiento pero aplicado a un archivo.

Excepciones

E-1 No es posible establecer conexión con el servidor, se despliega un mensaje informando el error ocurrido.

✓ Obtener Información del PC

Propósito

Proporcionarle al usuario la posibilidad de ver la información del sistema, la información de red y la información de los procesos actualmente en ejecución, en el PC Servidor

Resumen

El usuario inicia este caso de uso seleccionando la opción "Obtener Información PC" en el caso de uso "Ejecutar Operación"

Precondiciones

Se debe haber ejecutado previamente el caso de uso "Ejecutar Operación"

Flujo Principal

Se despliega la ventana “M2_4_ObtenerInformacionPC”, la cual ofrece las siguientes opciones: “Información de Red”, “Información de los Procesos”, “Información del sistema”, para realizar la consulta de la información requerida se hace uso de “Recursos PC”.

Si el usuario elige “Información de Red”, se continúa con el subflujo S-1

Si el usuario elige “Información de los Procesos”, se continúa con el subflujo S-2

Si el usuario elige “Información del Sistema”, se continúa con el subflujo S-3

Subflujos

S-1 Se despliega la ventana “M2_4_1_InformaciondeRed” con la siguiente información del PC servidor actual:

- Hostname: es el nombre del equipo en la red
- Dirección Ip
- Máscara de Subred
- Puerta de Enlace
- Número de puerto
- Dirección Mac

S-2 Se despliega la ventana “M2_4_2_InformacionProcesos” con una tabla de los procesos actualmente en ejecución, la cantidad de memoria usada por cada proceso y su estado “Activo” si está funcionando correctamente, o “No responde” si está bloqueado.

El usuario puede elegir la opción “Matar” el proceso seleccionado. Luego se despliega una ventana informando el éxito de la operación.

S-3 Se despliega la ventana “M2_4_3_InformacionSistema” con información del sistema como: cantidad de memoria RAM, RAM disponible, Información del Procesador, sistema operativo, etc.

Excepciones

E-1 No es posible establecer conexión con el servidor, se despliega un mensaje informando el error ocurrido.

- ✓ **Buscar**

Propósito

Permitirle al usuario buscar un archivo en el PC y realizar operaciones sobre algún archivo seleccionado.

Resumen

El usuario inicia este caso de uso seleccionando la opción “Buscar” en el caso de uso “Ejecutar Operación”

Precondiciones

Se debe haber ejecutado previamente el caso de uso “Ejecutar Operación”

Flujo Principal

Se despliega la ventana “M2_1_Buscar”, donde se debe digitar el nombre del archivo a buscar, opcionalmente se pueden usar los comodines usados en la consola de Windows, el * y la ?, donde el * significa cualquier conjunto de caracteres, y la ? significa cualquier caracter, por ejemplo:

· a*.*, busca todos los archivos que comiencen con la letra ‘a’ y con cualquier extensión

· *.j?g busca los archivos que tienen una extensión que inicia con j, seguido de cualquier carácter y termina con g.

A continuación, el usuario puede restringir la búsqueda seleccionando una o más opciones de tipo Directorio, Oculto y Solo Lectura.

Por ejemplo, si el usuario seleccionara Directorio y Oculto, se buscarían sólo directorios ocultos que coincidan con el patrón entrado anteriormente.

Se despliega la ventana “M2_1_0_1_SeleccionarCarpetaOrigenPC” donde el usuario navegará por el sistema de archivos buscando el directorio donde se efectuará la búsqueda y a continuación debe presionar “seleccionarCarpeta”, luego el sistema realiza la búsqueda y muestra la ventana “M2_1_1_ResultadosBusquedaPC”.

El usuario puede decidir entre las siguientes opciones:

- Renombrar: Se despliega la ventana “M3_1_Renombrar” en donde el usuario debe ingresar el nuevo nombre de la carpeta y luego seleccionar la opción “Renombrar” para cambiarle el nombre a la carpeta seleccionada, luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Eliminar: El usuario puede eliminar la carpeta seleccionada con todo su contenido, aparece una ventana de confirmación de eliminación, si el usuario selecciona “OK”, se ejecuta la eliminación. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Mover: Se despliega la ventana “M3_3_SeleccionarDispositivoDestino” donde el usuario debe elegir el dispositivo destino (PC o Celular) donde se alojará la carpeta seleccionada. Si el usuario seleccionó PC, se despliega la ventana “M3_3_1_SeleccionarCarpetaDestinoPC”. Si el usuario seleccionó Celular, se despliega la ventana “M3_3_2_SeleccionarCarpetaDestinoCelular”. Luego el usuario puede navegar por el sistema de archivos del dispositivo destino y elegir la carpeta destino. Se muestra la ventana “M4_IngresarNombreDestino” para ingresar el nombre de la carpeta seleccionada en la carpeta destino. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Cambiar Permisos: Se despliega la ventana “M3_4_CambiarPermisos” donde el usuario puede elegir los nuevos permisos de acceso a la carpeta seleccionada. Dichos permisos son: “Solo lectura”, “Sistema”, “Oculto” y “Almacenamiento”. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Copiar: Se despliega la ventana “M3_3_SeleccionarDispositivoDestino” donde el usuario debe elegir el dispositivo destino (PC o Celular) donde se alojará la carpeta seleccionada. Si el usuario seleccionó PC, se despliega la ventana “M3_3_1_SeleccionarCarpetaDestinoPC”. Si el usuario seleccionó Celular, se despliega la ventana “M3_3_2_SeleccionarCarpetaDestinoCelular”. Luego el usuario puede navegar por el sistema de archivos del dispositivo destino y elegir la carpeta destino. Se muestra la ventana “M4_IngresarNombreDestino” para ingresar el nombre de la carpeta seleccionada en la carpeta destino. Luego se despliega la ventana “M3_0_ResultadoOperacion”.

- Copiar Ruta: Guarda temporalmente la ruta de la carpeta seleccionada, con el fin de usarla luego en la consola remota.

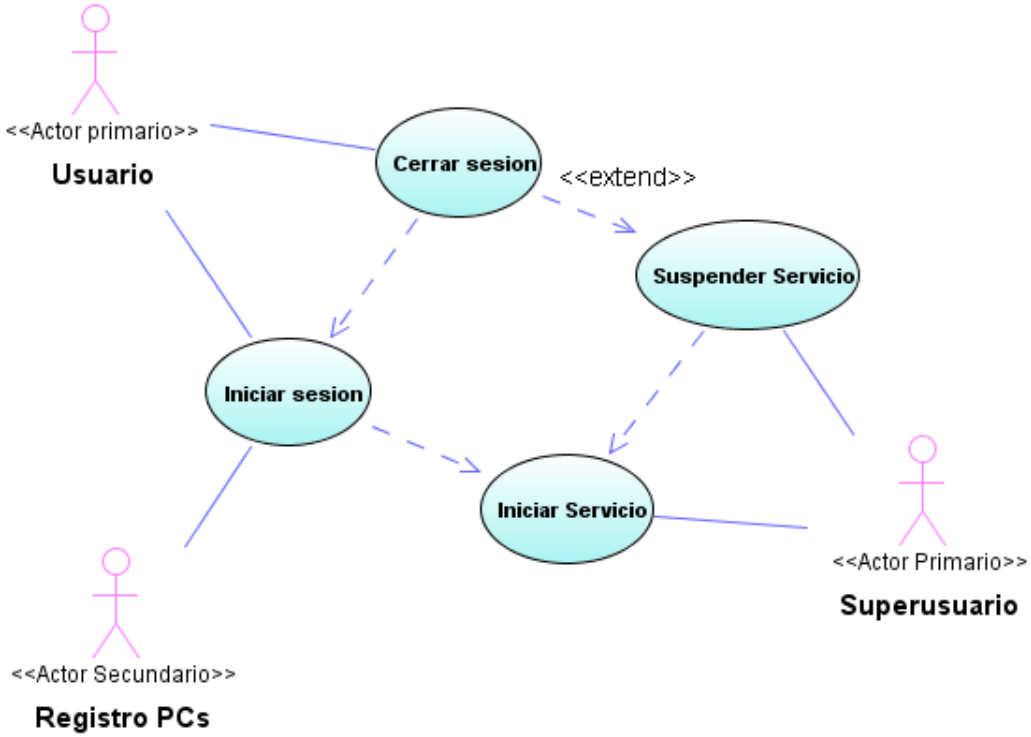
Excepciones

E-1 No es posible establecer conexión con el servidor, se despliega un mensaje informando el error ocurrido.

2.3. *DIAGRAMA DE CONECTIVIDAD*

En el diagrama de casos de uso de conectividad se relacionan las opciones que es posible realizar cada actor humano Usuario y Superusuario con respecto a la conexión entre el cliente y el Servidor, siendo el Superusuario quien inicia o detiene el servicio y el usuario quien lo utiliza cuando este está iniciado.

Figura 68. Diagrama de casos de uso de Conectividad



3. INTERFAZ GRAFICA DE USUARIO

A continuación se muestran las ventanas que se realizaron en la etapa de análisis para modelar la interacción de usuario con el sistema para ambos aplicaciones Cliente y Servidor.

3.1. *VENTANAS APLICACIÓN SERVIDOR*

Figura 69. W0_Ventana Principal

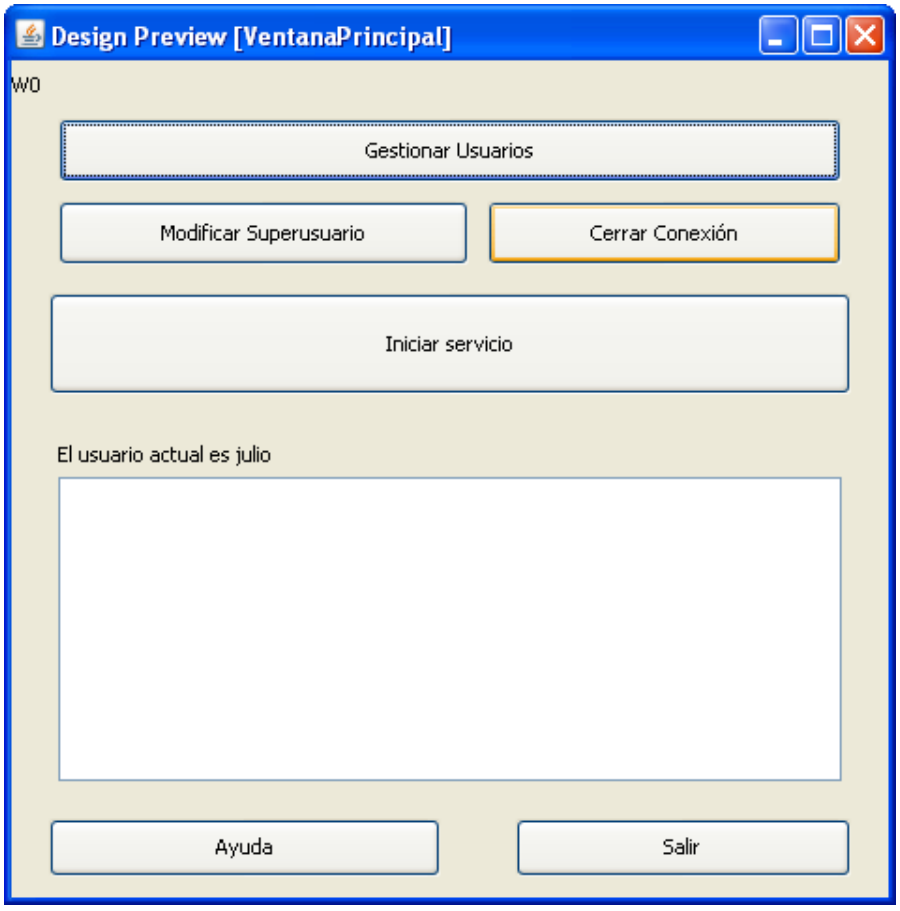


Figura 70. W6_Ventana Gestionar Usuarios

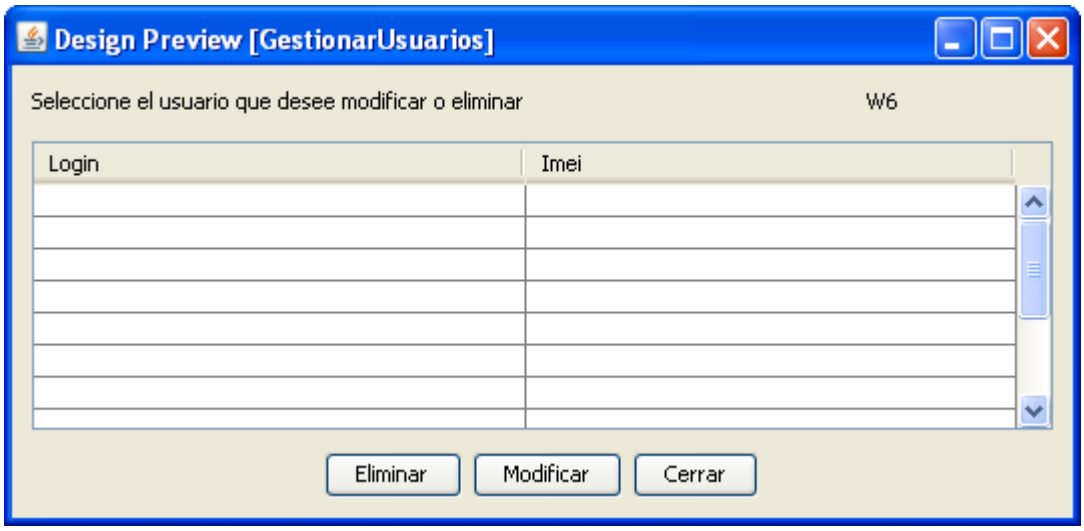


Figura 71. W4_Modificar Superusuario

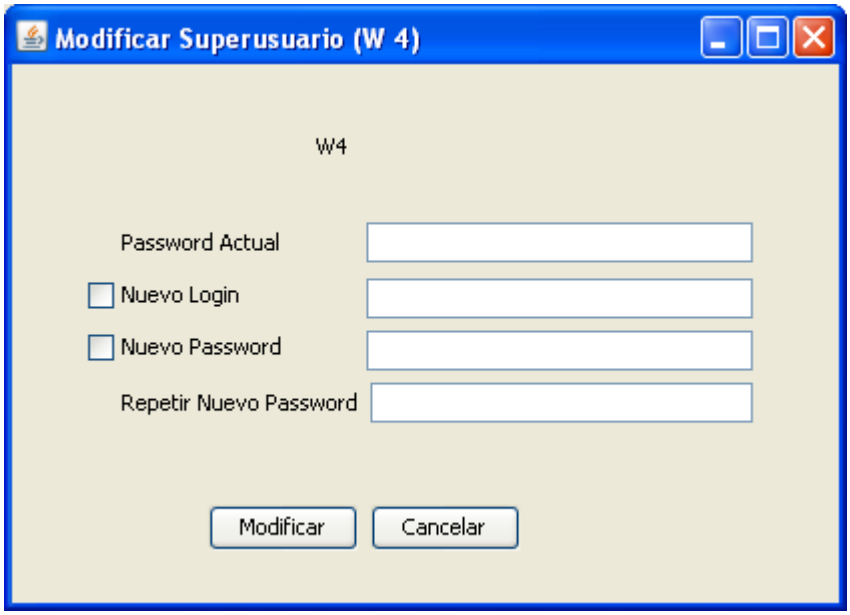


Figura 72. W2_Modificar Usuarios

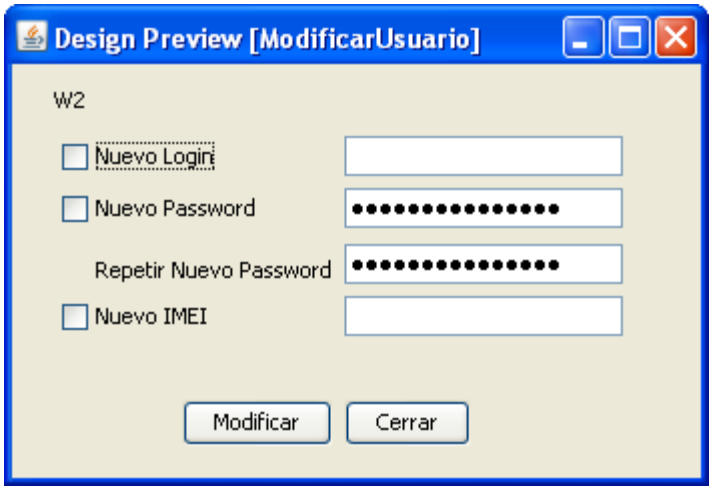


Figura 73. W5_Validar Superusuario

3.2. *VENTANAS APLICACIÓN CLIENTE*

Figura 74. Ventana M0_VentanaPrincipal

Figura 75. Ventana M0_1_2_1_RegistrarPC

Figura 76. Ventana M0_2_2_ValidarSuperusuario

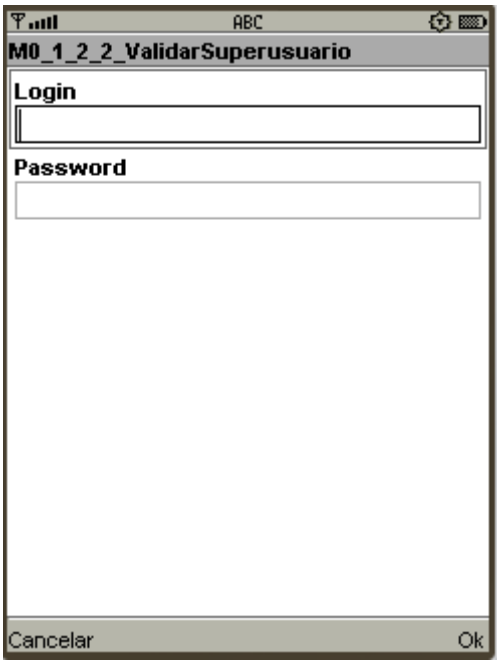


Figura 77. Ventana M0_2_SeleccionarPC

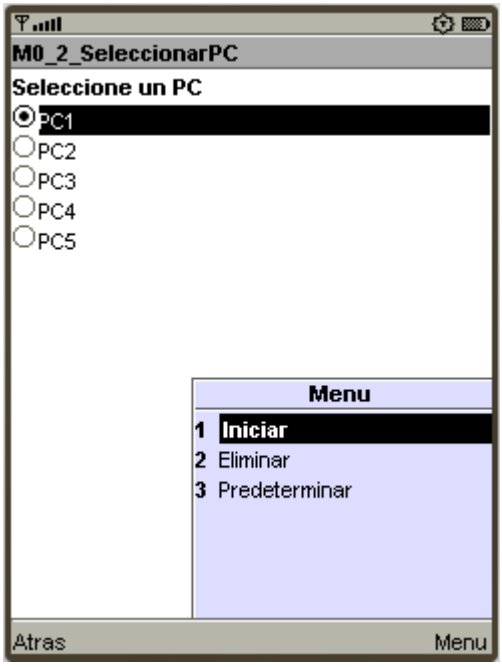


Figura 78. Ventana M1_PantallaDeInicioSesion

Figura 79. Ventana M1_PantallaDeInicioSesion

Figura 80. Ventana M1_0_ConfirmarEliminarPC

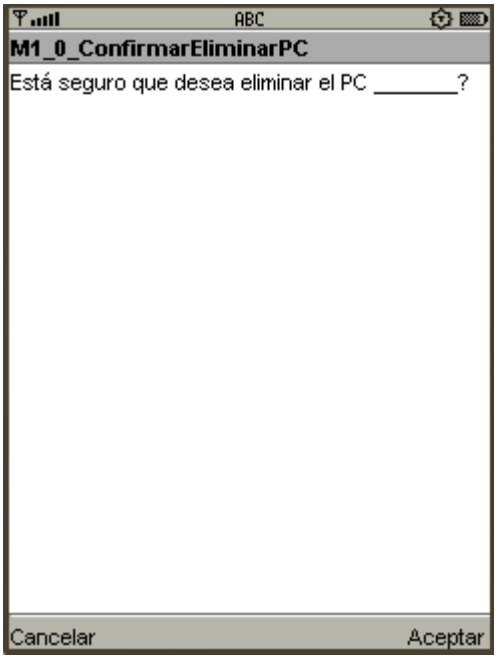


Figura 81. Ventana M2_Operaciones

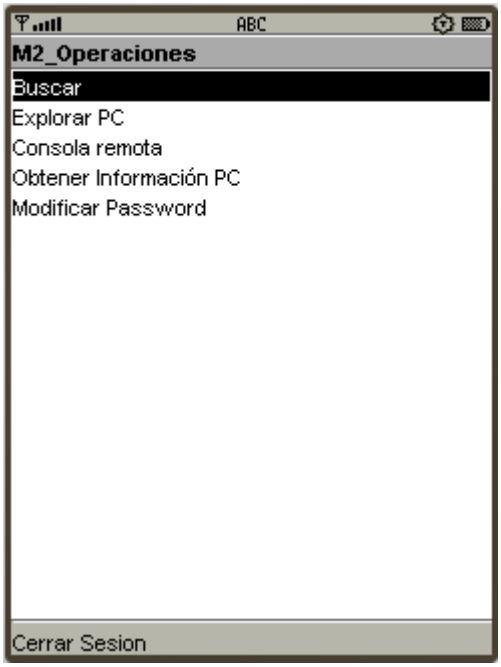


Figura 82. Ventana M2_1_Buscar

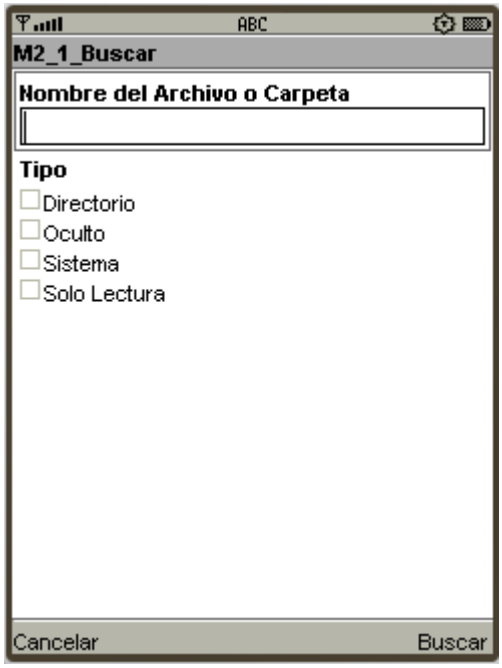


Figura 83. Ventana M2_1_0_SeleccionarOrigen

Figura 84. Ventana M2_1_0_1_SeleccionarCarpetaOrigenPC

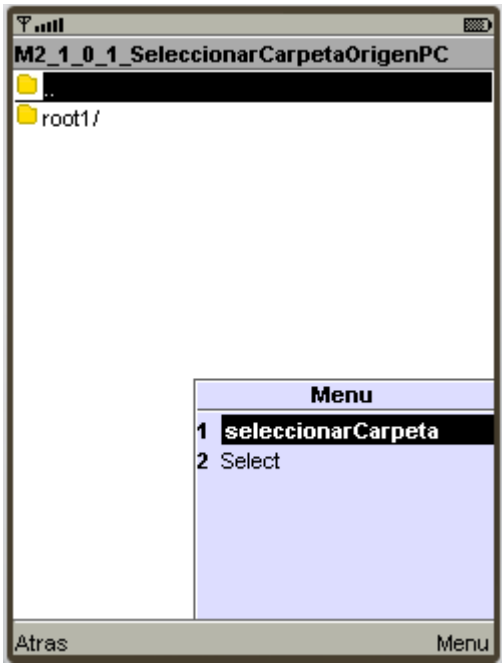


Figura 85. Ventana M2_1_0_2_SeleccionarCarpetaOrigenCelular

Figura 86. Ventana
M2_1_1_ResultadoBusquedaPC

Figura 87. Ventana
M2_1_2_ResultadoBusquedaCelular

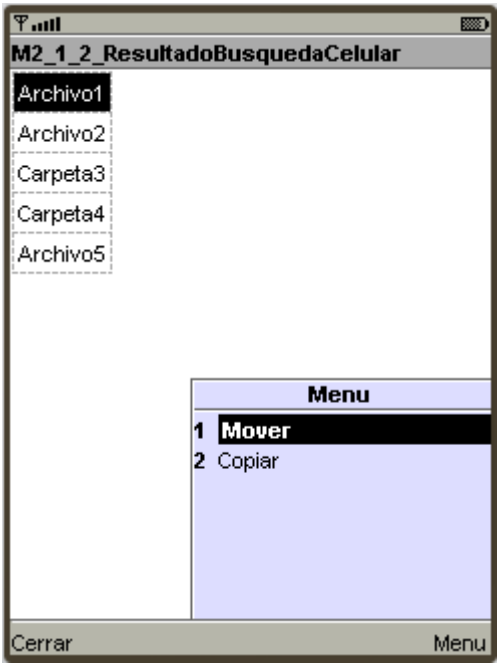


Figura 88. Ventana
M13_0_SeleccionarDispositivoDestino

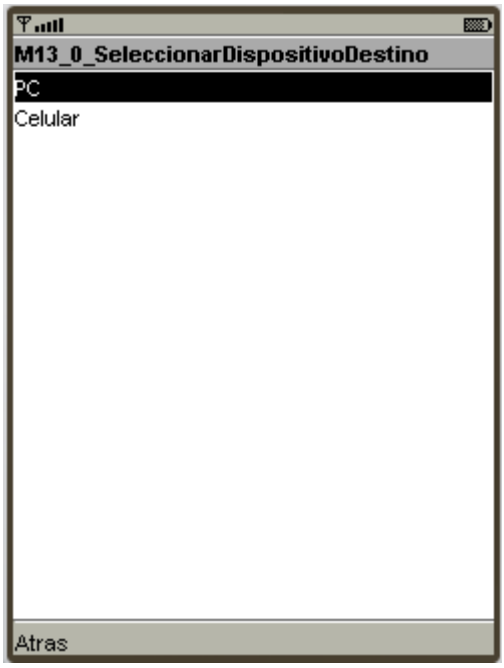


Figura 89. Ventana
M13_1_SeleccionarCarpetaDestinoPC

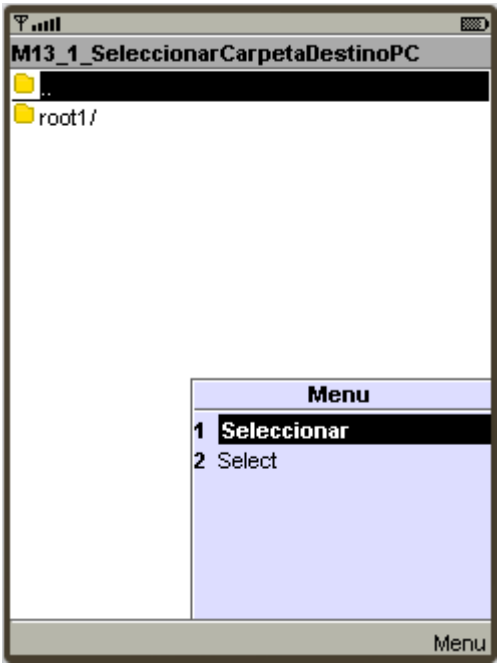


Figura 90. Ventana M13_2_SeleccionarCarpetaDestinoCelular

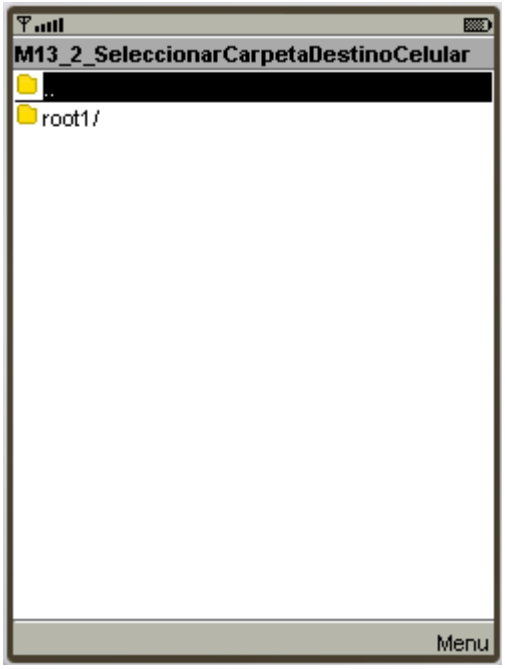


Figura 91. M13_3_IngresarNombreDestino

Figura 92. Ventana M3_0_ResultadoOperacion

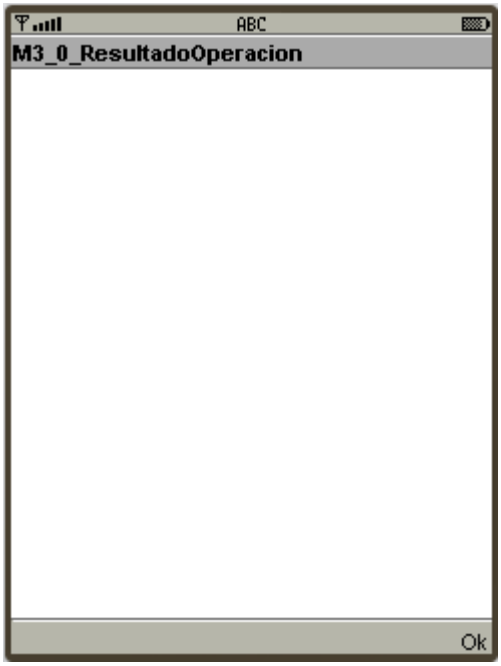


Figura 93. Ventana M3_1_Renombrar

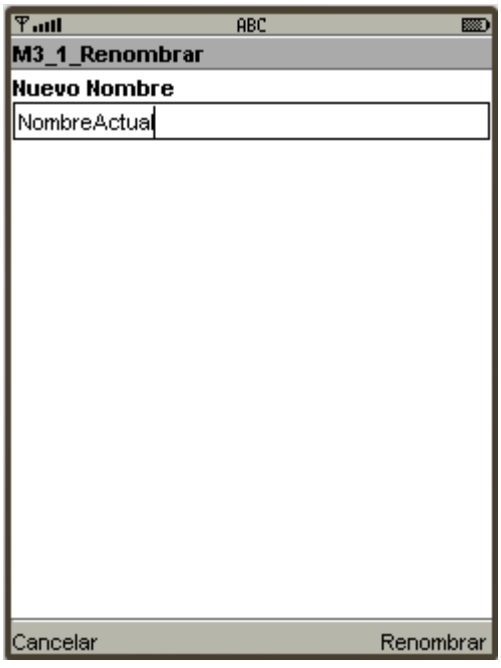


Figura 94. VentanaM3_4_CambiarPermisos

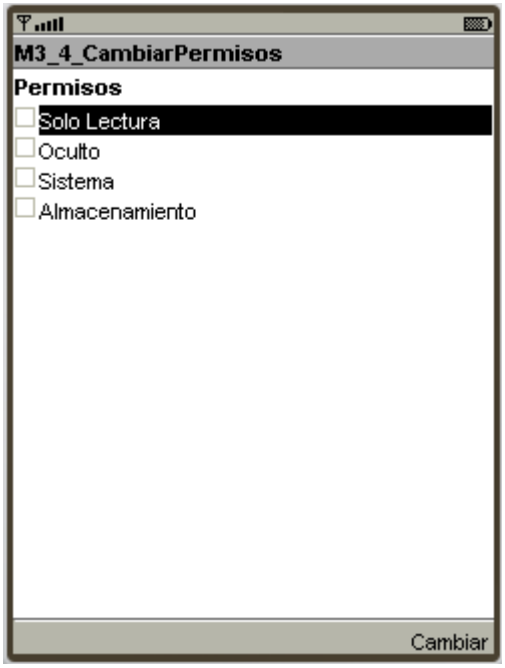


Figura 95. Ventana M2_2_ExplorarPC

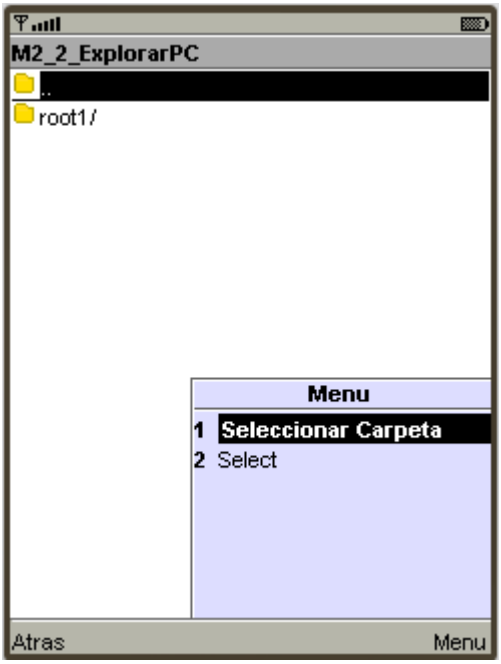


Figura 96. Ventana M2_2_2_OperacionesCarpeta

Figura 97. Ventana M3_2_CrearCarpeta

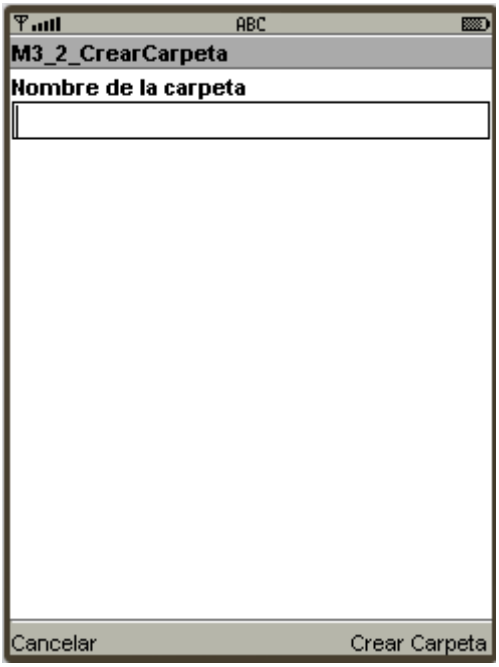


Figura 98. Ventana M2_3_ConsolaRemota

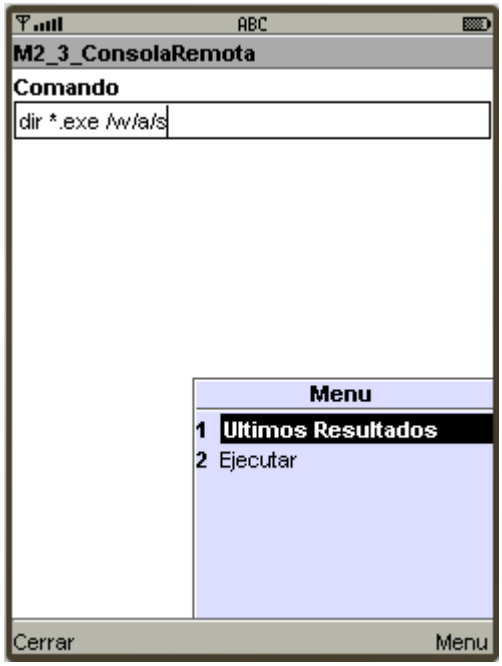


Figura 99. Ventana M2_4_ObtenerInformacionPC

Figura 100. Ventana M2_4_1_InformaciondeRed

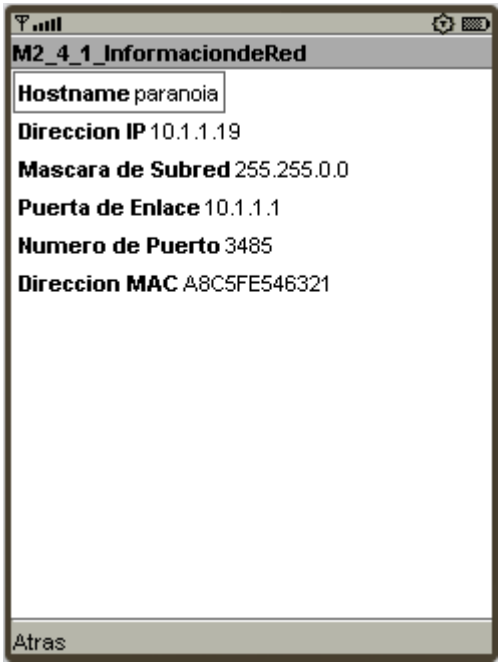


Figura 101. Ventana M2_4_2_InformacionProcesos

Figura 102. Ventana M2_4_2_1_ResultadoMatarProceso

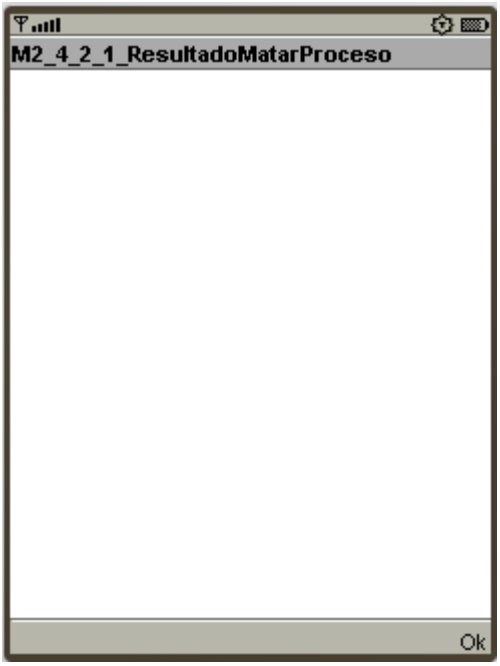


Figura 103. Ventana M2_4_3_InformacionSistema

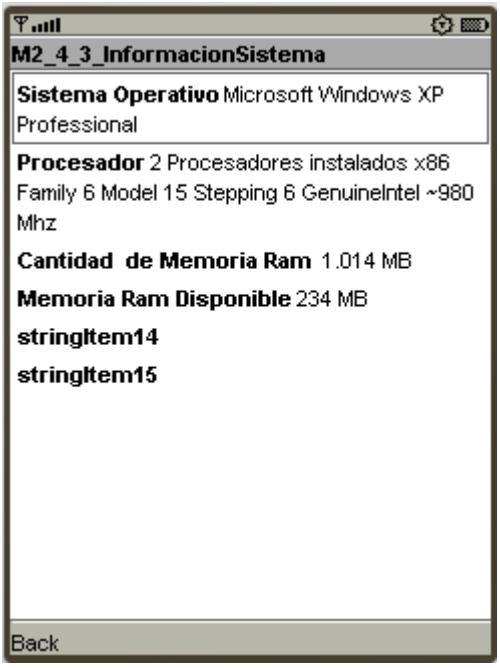


Figura 104. M2_4_ModificarPassword

4. MODELO DE DOMINIO

4.1. DESCRIPCIÓN DEL SISTEMA:

El sistema permite compartir archivos y dar órdenes de consola desde un celular para ser ejecutadas en un PC con sistema operativo Windows utilizando servicios Web, aprovechando la implementación de la especificación JSR 172 de algunos celulares.

El sistema está compuesto por una aplicación cliente y una aplicación servidor.

La aplicación servidor es administrada por un Superusuario encargado de gestionar cuentas de usuario, habilitar o denegar el acceso al PC y registrar un PC. Una cuenta de usuario está compuesta por Login, Password e IMEI. Un registro de un PC consiste de Nombre del PC, Dirección IP, Puerto; el celular puede tener registros de varios PCs. El proceso de registrar un PC en un celular requiere tener la autorización del Superusuario por medio de su login y password, y se requiere también tener instalada la aplicación cliente en el celular. El objetivo de registrar un PC es permitirle el acceso a los archivos y carpetas del PC y la ejecución comandos.

La aplicación cliente se encuentra en un celular y le permite a un usuario realizar las siguientes tareas:

Copiar y mover archivos entre el Celular y un PC,

Ejecución remota de comandos de consola desde el celular.

Obtener información del sistema como la cantidad de memoria RAM instalada, memoria RAM disponible, información del procesador, versión del sistema operativo, etc.

Obtener información de red como el nombre del host, dirección ip, máscara de subred, puerta de enlace, numero de puerto y dirección Mac

Obtener información de los procesos, lo cual consta de nombre del proceso, cantidad de memoria usada y su respectivo estado. También permitirá matar procesos.

Búsqueda de archivos o carpetas en el celular o en el PC

Ejecutar operaciones sobre los archivos y carpetas como renombrar, eliminar y cambiar permisos de acceso. También permitirá crear carpetas.

Registrar la información de varios PCs para permitirle al usuario acceder a los recursos de cada PC en cualquier momento

4.2. *SELECCIÓN DE CLASES CANDIDATAS*

Tabla 1. Proceso de la selección de clases candidatas

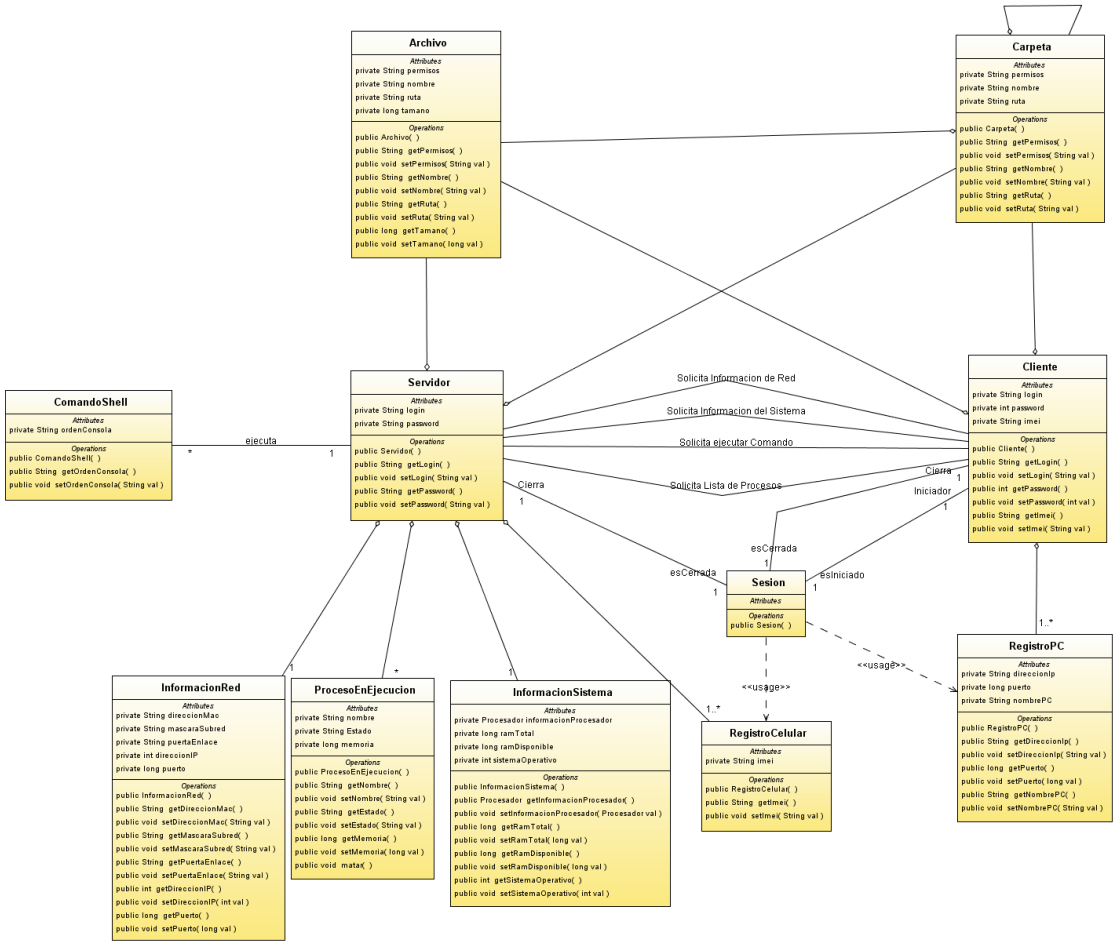
Clases Candidatas	**Modificación**
aplicación cliente	Renombrada : **Cliente**
aplicación servidor	Renombrada : **Servidor**
Archivo	
autorización del Superusuario	Eliminada (operación)
Carpeta	
celular	Eliminada (duplicada con Cliente)
comandos de consola	Renombrado : **ComandoShell**
cuenta de usuario	Eliminada (duplicada con Usuario)
dirección ip	Eliminada (atributo)
dirección mac	Eliminada (atributo)
ejecución comandos	Eliminada (operación)
Estado	Eliminada (imprecisa)
IMEI	Eliminada (atributo)
implementación de la especificación JSR 172	Eliminada (irrelevante)
información de los procesos	Renombrada : **ProcesoEnEjecucion**
información de red	Renombrada : **InformacionRed**
información del procesador	Eliminada (atributo)
Información del sistema	
Login	Eliminada (atributo)

máscara de subred	Eliminada (atributo)
memoria RAM	Eliminada (atributo)
memoria usada	Eliminada (atributo)
nombre del host	Eliminada (atributo)
nombre del proceso	Eliminada (atributo)
numero de puerto	Eliminada (atributo)
órdenes de consola	Eliminada (atributo)
Password	Eliminada (atributo)
PC	Eliminada (duplicada con Servidor)
permisos de acceso	Eliminada (atributo)
proceso	Eliminada (duplicada con Proceso)
puerta de enlace	Eliminada (atributo)
recursos de cada PC	Eliminada (duplicada con Archivo y Carpeta)
registro de un PC	Renombrada : **RegistroPCs**
servicios Web	Eliminada (irrelevante)
sistema operativo	Eliminada (atributo)
Superusuario	Eliminada (Actor)
Usuario	Eliminada (Actor)
versión del sistema operativo	Eliminada (atributo)
Windows	Eliminada (atributo)

4.3. *DIAGRAMA DEL DOMINIO*

El siguiente corresponde al modelo del dominio el cual pretende mostrar de una forma generalizada cada uno de los elementos del proyecto para aterrizar los requerimientos a algo más comprensible por nosotros en forma de clases.

Figura 105. Diagrama de clases del dominio



5. DICCIONARIO DE CLASES

- ✓ **Archivo**

Puede estar en un celular o en un PC, puede tener diferentes extensiones, su atributo principal es el nombre.

- ✓ **Carpeta**

Es un contenedor de archivos, su atributo principal es su ruta que la identifica de forma única en el sistema.

- ✓ **Cliente**

Realiza peticiones al servidor, ofrece un mecanismo de autenticación de usuarios a través de login, password e IMEI.

- ✓ **ComandoShell**

Es una orden del sistema operativo que permite realizar diferentes acciones, entre ellas: copiar, mover, renombrar y eliminar archivos y carpetas.

- ✓ **InformacionRed**

Está compuesto de dirección IP, puerto, dirección MAC, máscara de subred y puerta de enlace. Esta información se obtiene de la configuración actual de red del PC.

- ✓ **InformacionSistema**

Está compuesta de la información de hardware como el procesador y la memoria RAM, e información del sistema operativo instalado actualmente.

- ✓ **ProcesoEnEjecucion**

Representa un proceso actualmente en ejecución en el PC, contiene el nombre del proceso, el consumo de memoria y el estado actual, además da la opción de terminar el proceso.

- ✓ **Servidor**

Es la clase principal, se encarga de responder a las solicitudes del Cliente, para esto accede a información detallada del sistema y ejecuta comandos Shell remotos. Ofrece un mecanismo de autenticación del Superusuario a través de los datos login y password.

Sesión

Representa la conexión necesaria para el intercambio de mensajes entre el cliente y el servidor. Utiliza el registro del celular y el registro PC su instanciación.

- ✓ **RegistroCelular**

Está compuesta por el IMEI del celular

- ✓ **RegistroPC**

Está compuesto del nombre, la dirección IP y puerto del PC

6. DIAGRAMAS DE SECUENCIA

Los diagramas de secuencia intentan representar el flujo de eventos entre objetos, nos cuenta la historia de cómo sucedió un caso de uso. Estos modelan un determinado escenario. El diagrama modela el tiempo de arriba a abajo y los objetos están en la primera fila.

Figura 106. Diagrama de Secuencia Agregar Usuario

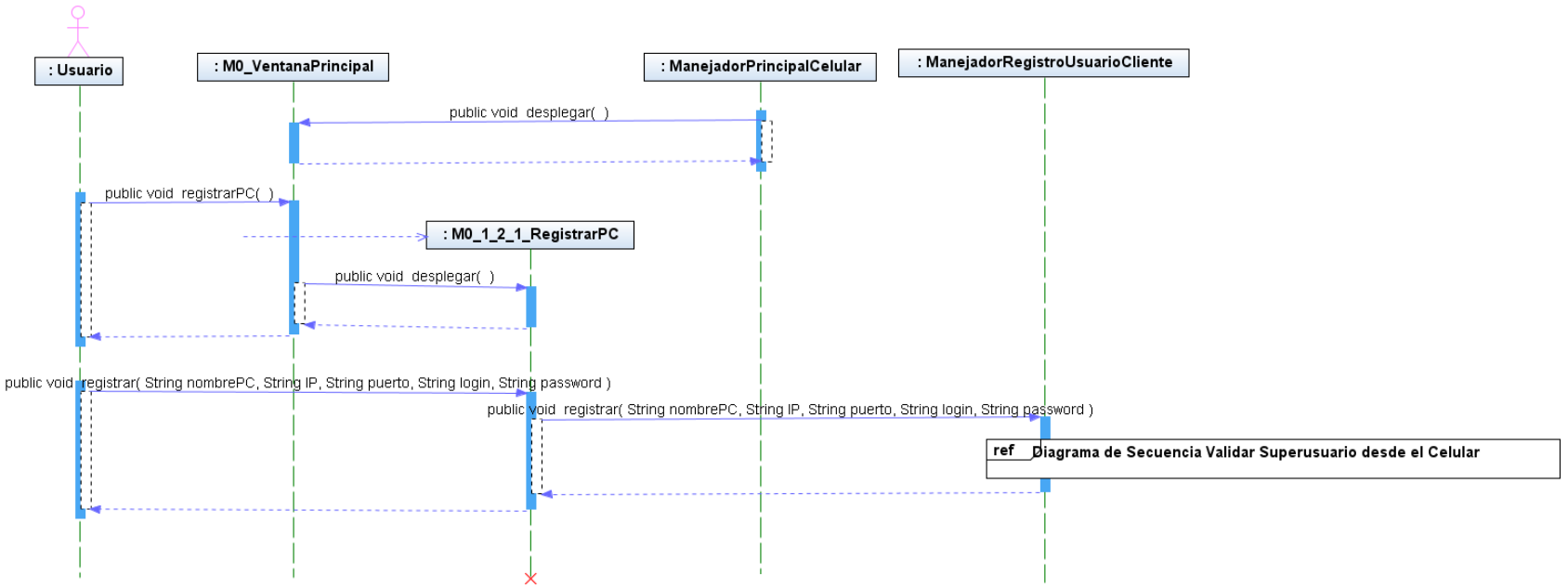


Figura 107. Diagrama de Secuencia Buscar Archivos o Carpetas

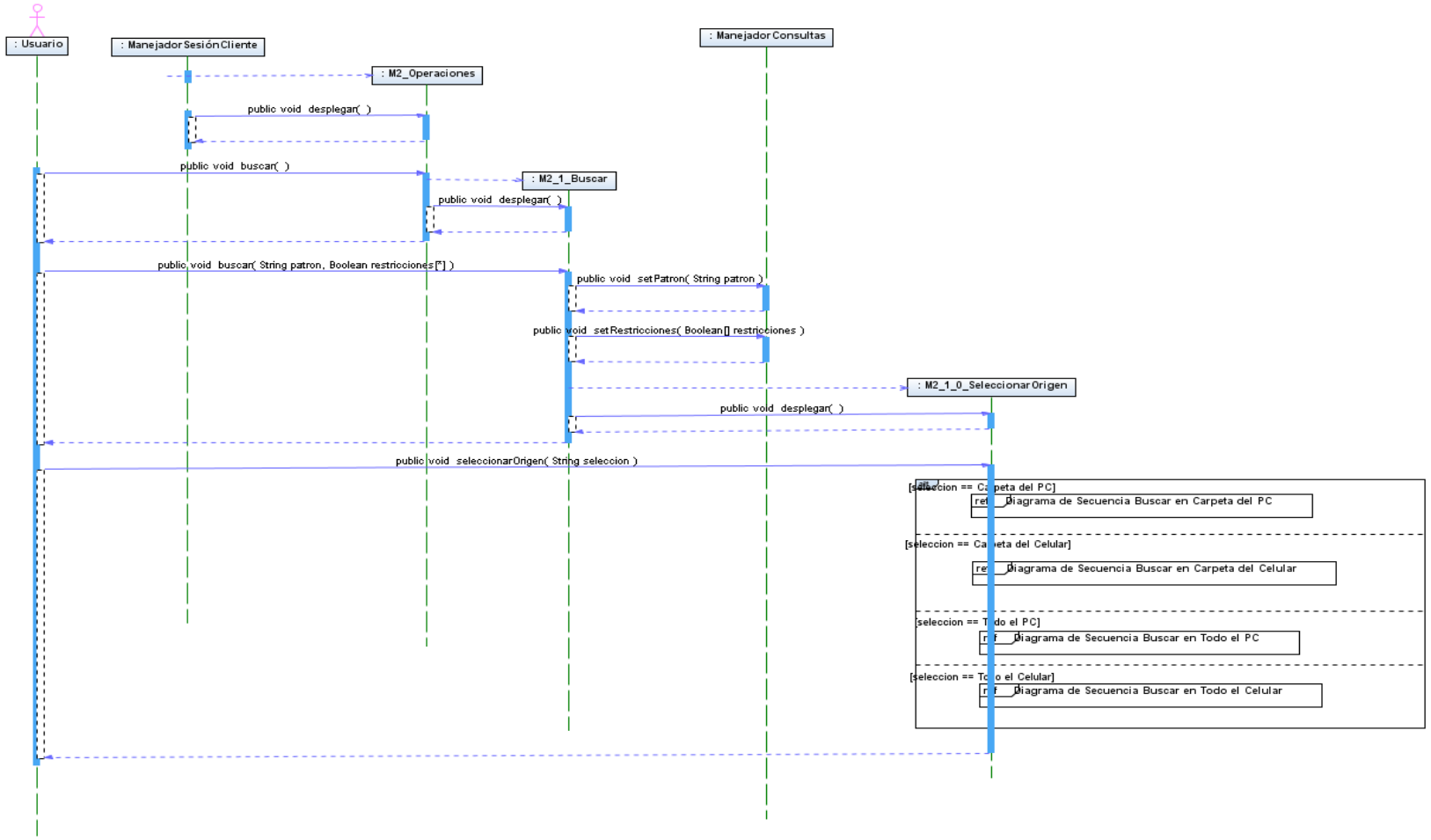


Figura 108. Diagrama de Secuencia Buscar en Carpeta del Celular

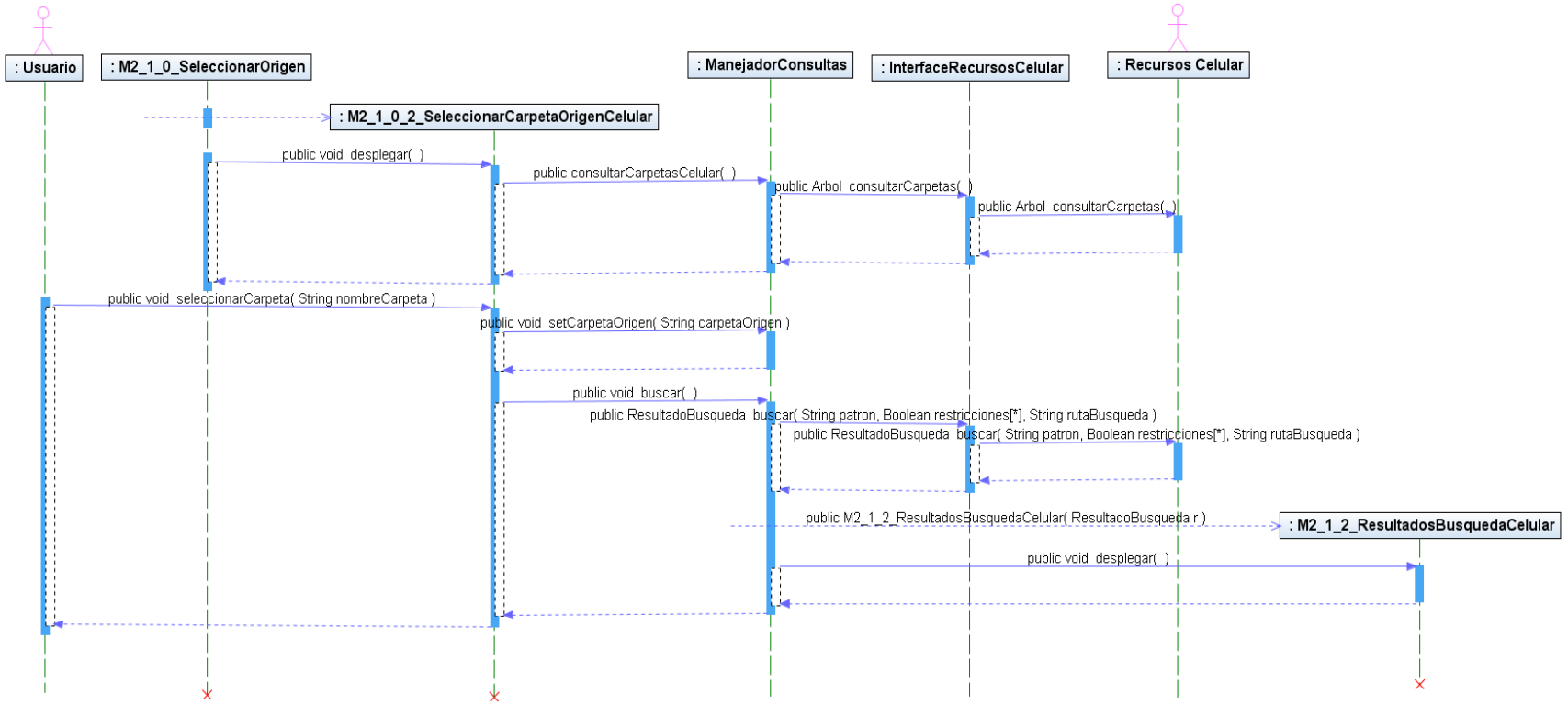


Figura 109. Diagrama de Secuencia Buscar en Carpeta del PC

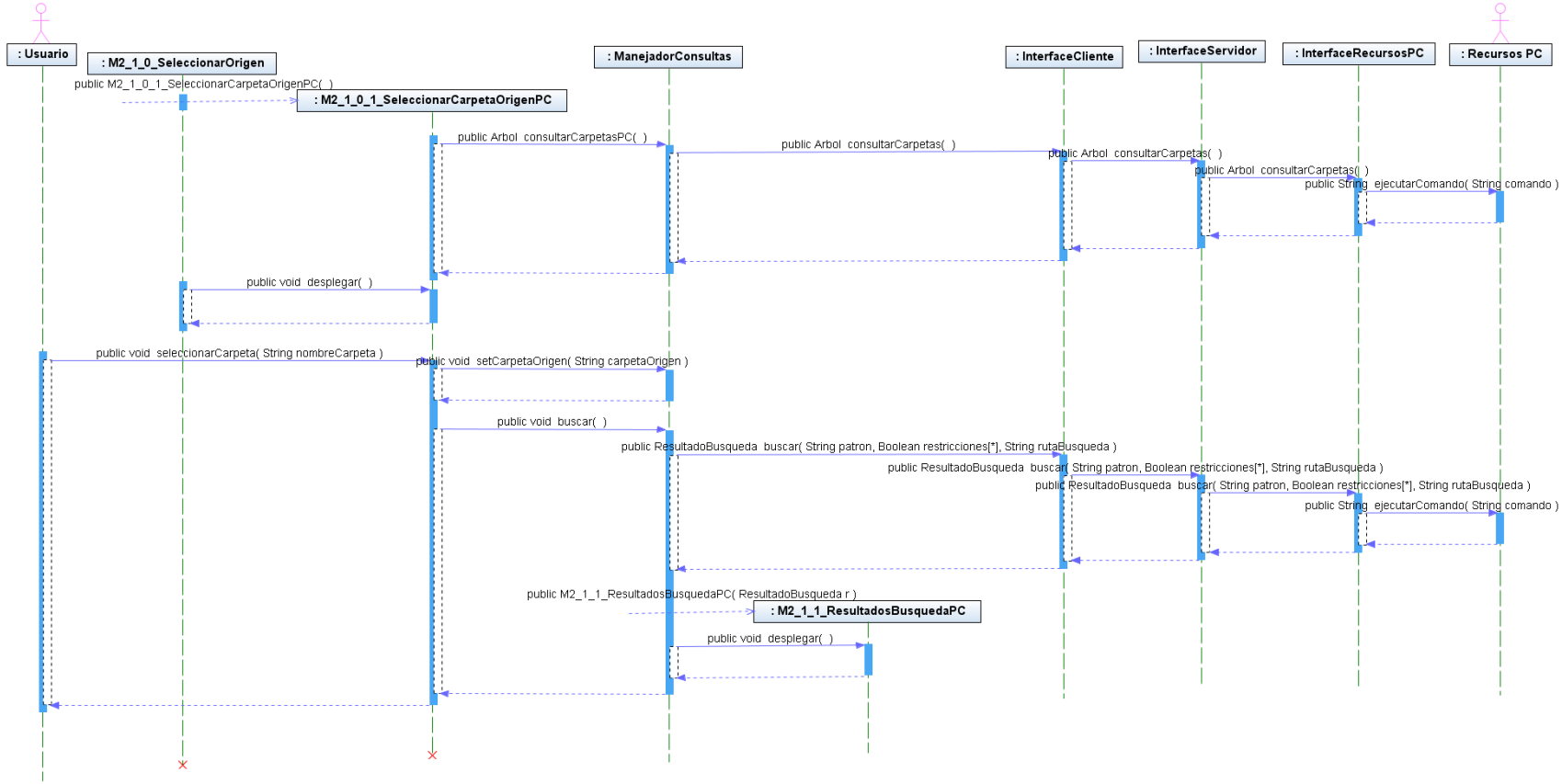


Figura 110. Diagrama de Secuencia Buscar en Todo el Celular

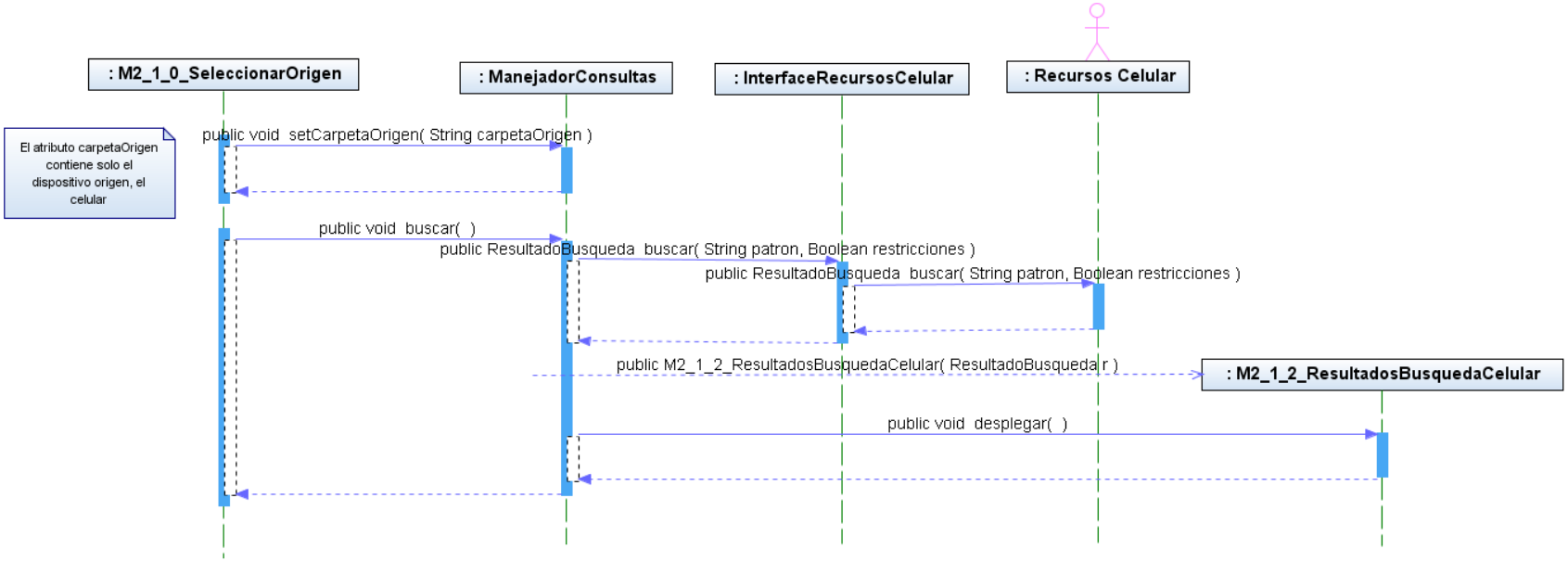


Figura 111. Diagrama de Secuencia Buscar en Todo el PC

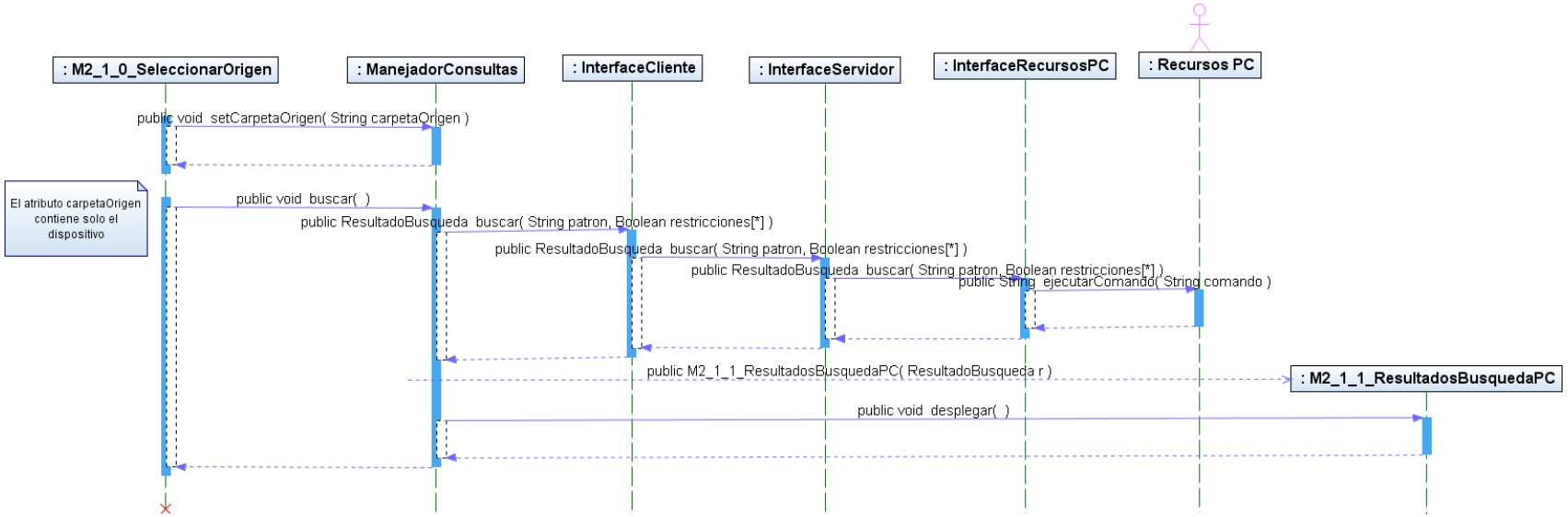


Figura 112. Diagrama de Secuencia Cambiar Permisos

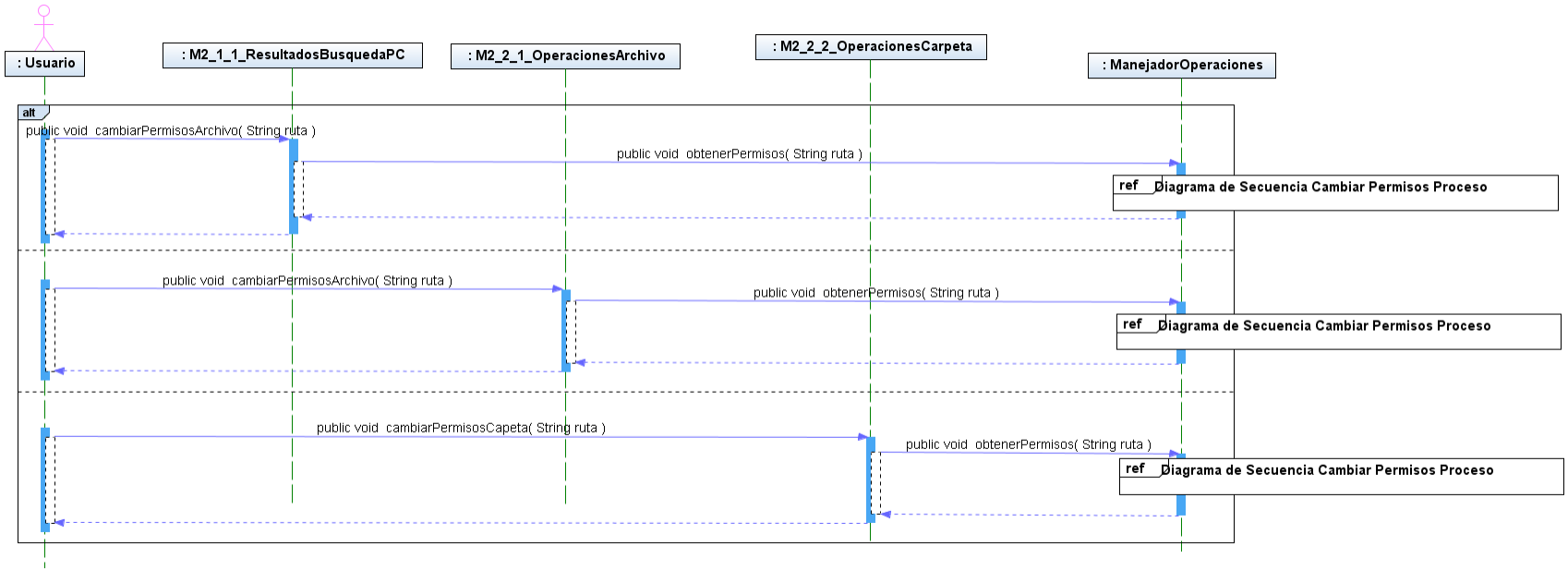


Figura 113. Diagrama de Secuencia Cambiar Permisos Proceso

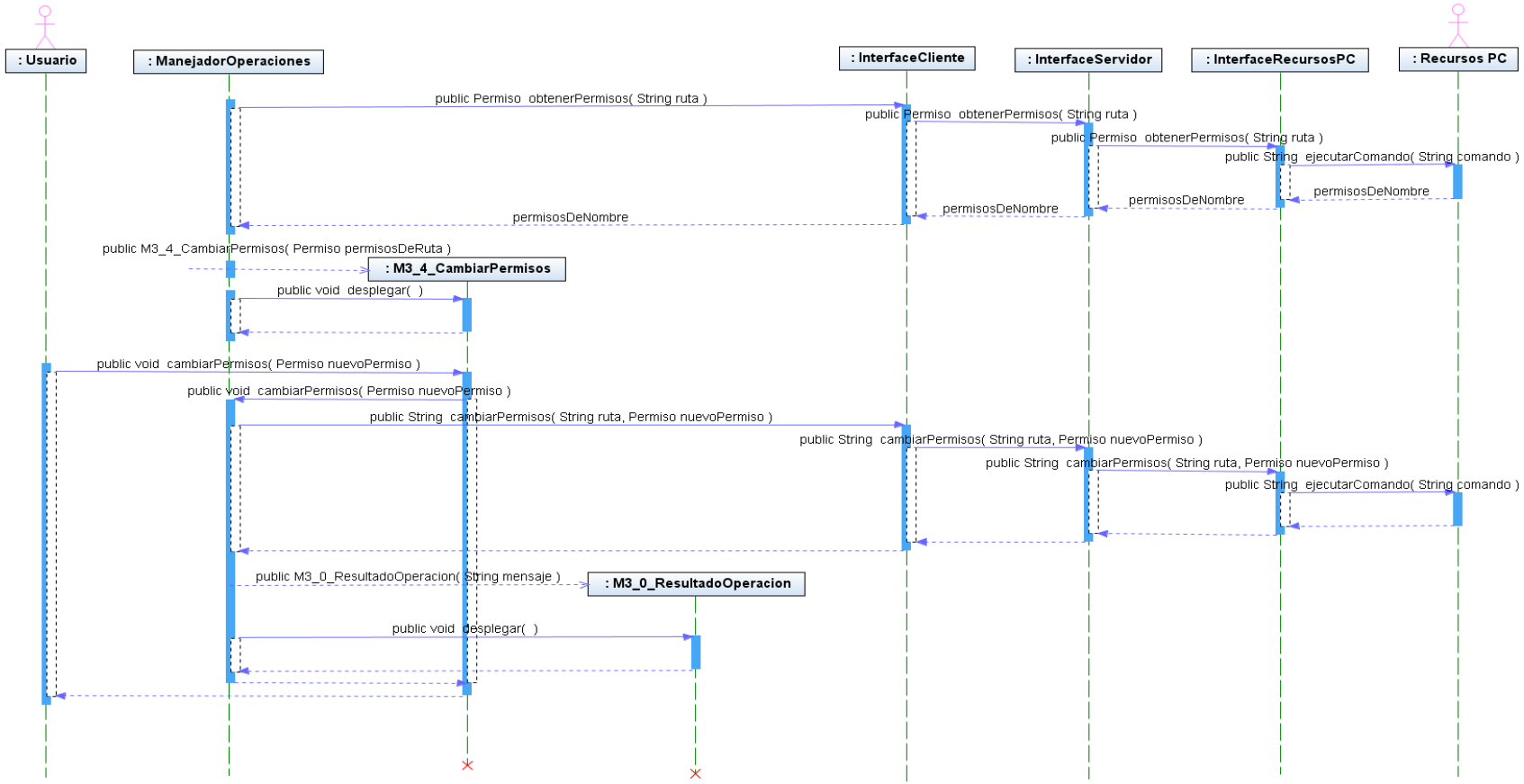


Figura 114. Diagrama de Secuencia Cerrar Sesión 1

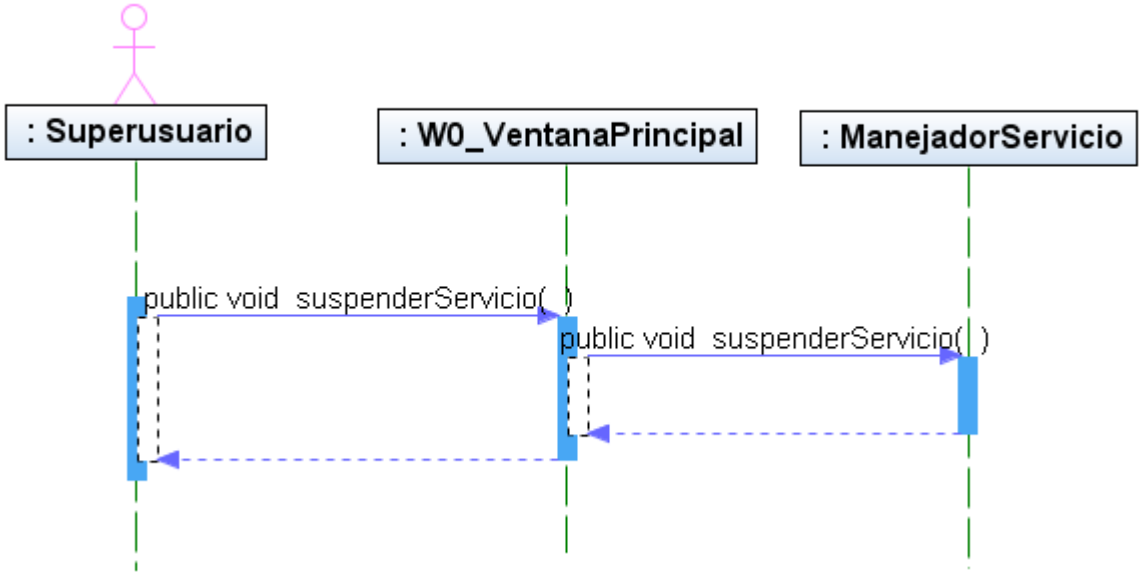


Figura 115. Diagrama de Secuencia Cerrar Sesión 2

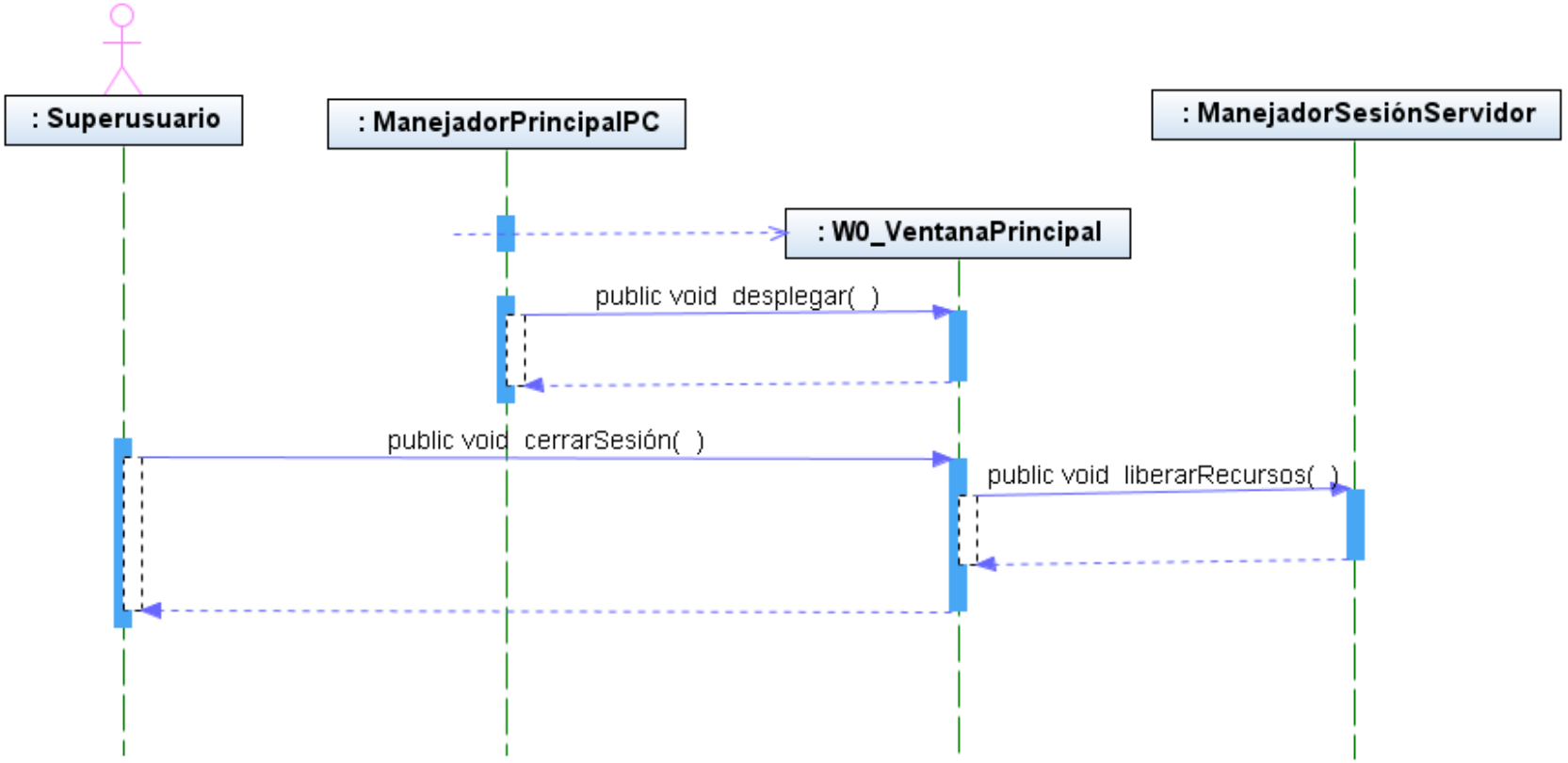


Figura 116. Diagrama de Secuencia Cerrar Sesión 3

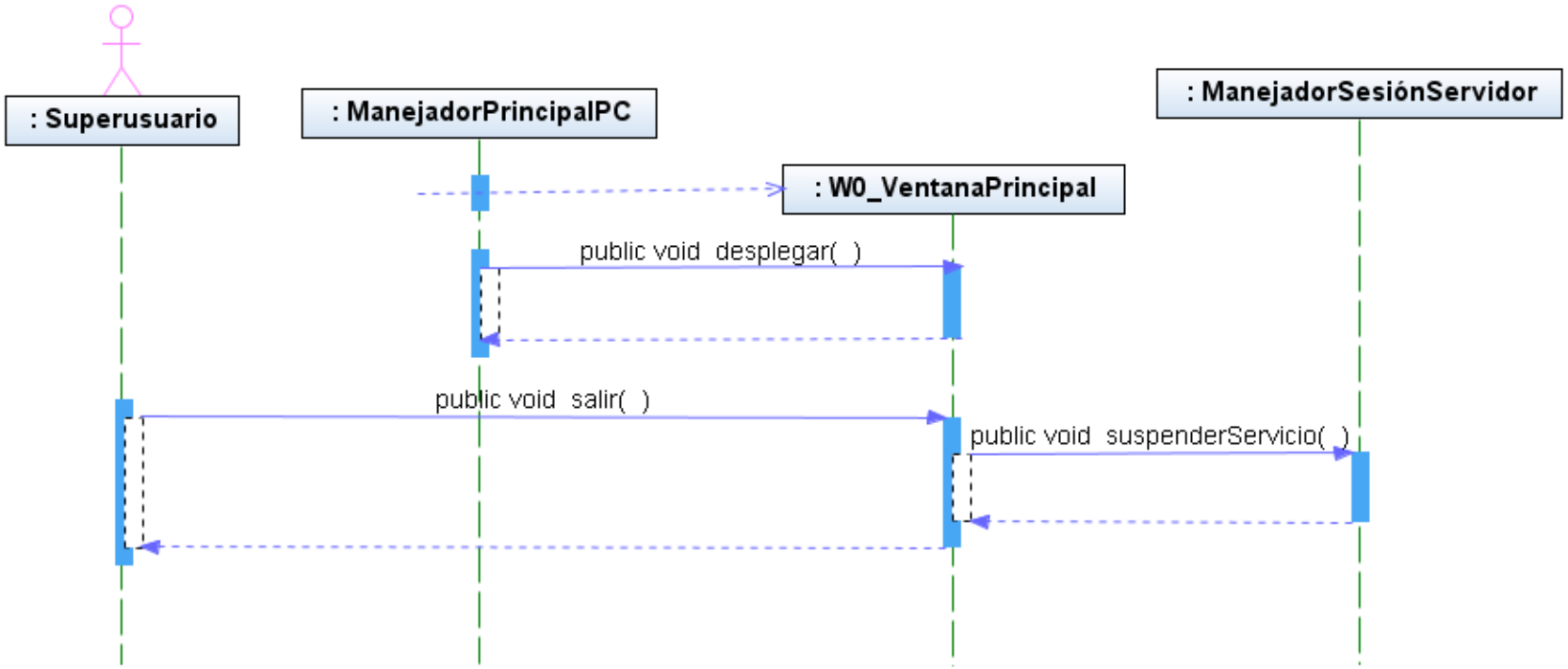


Figura 117. Diagrama de Secuencia Cerrar Sesión 4

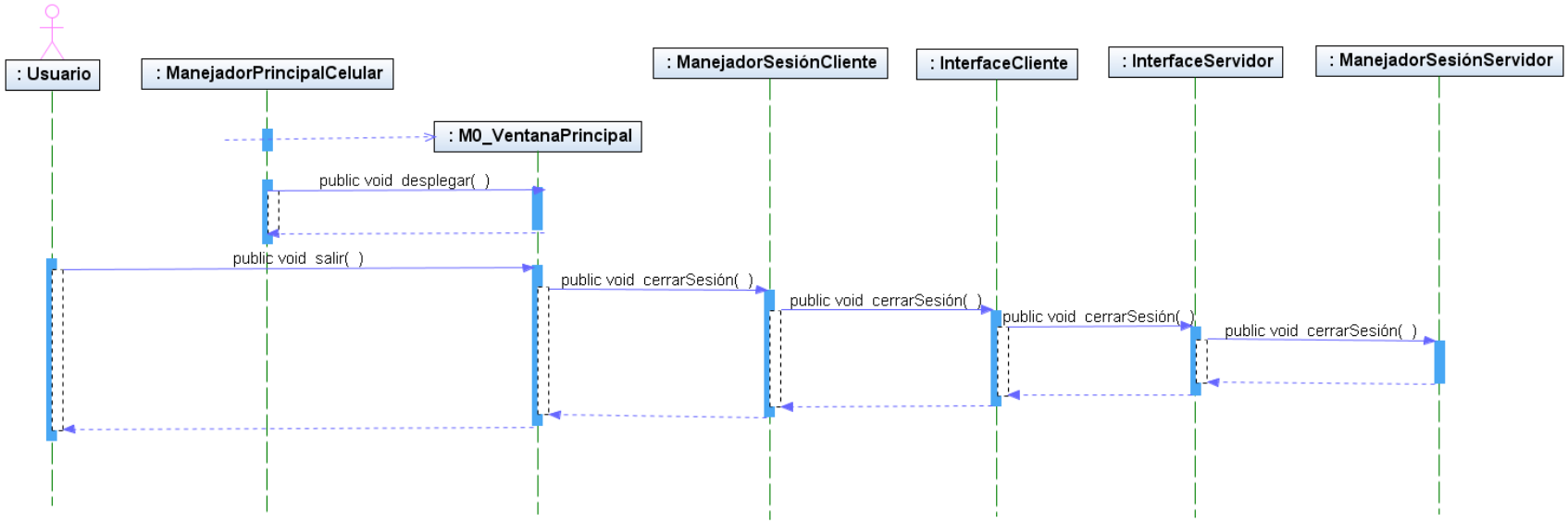


Figura 118. Diagrama de Secuencia Consultar Archivos o Carpetas

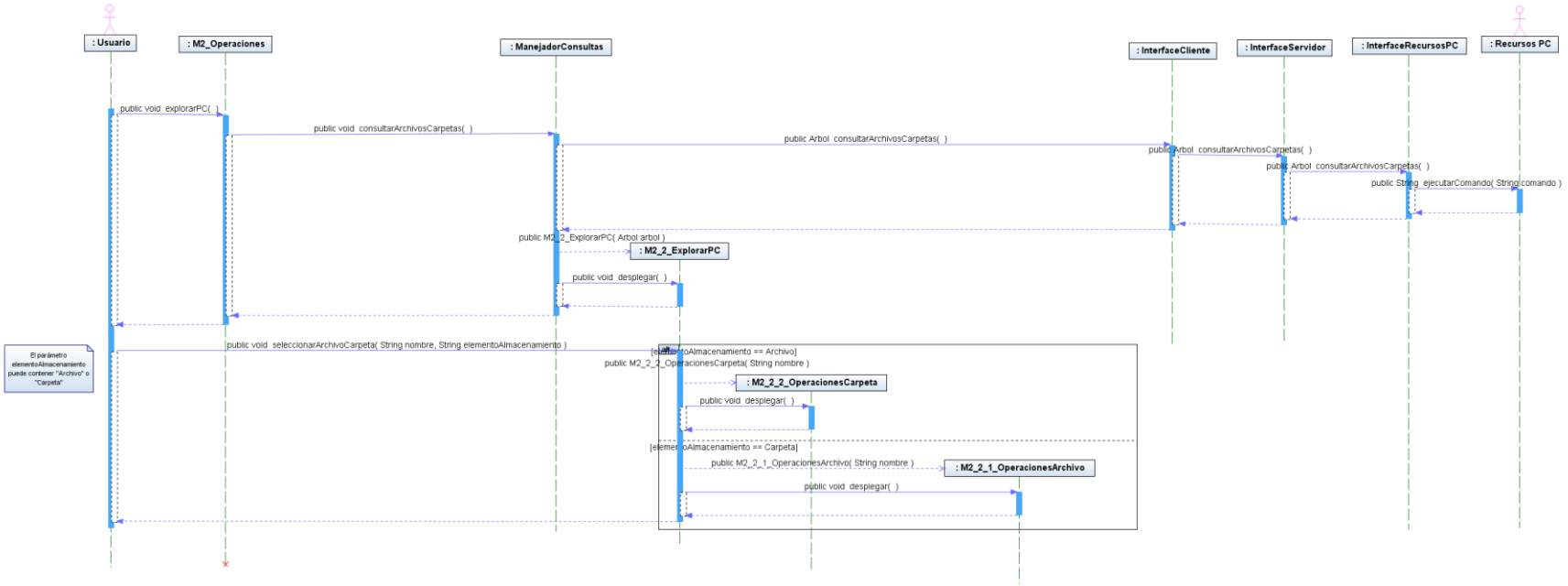


Figura 119. Diagrama de Secuencia Consultar Información de Red

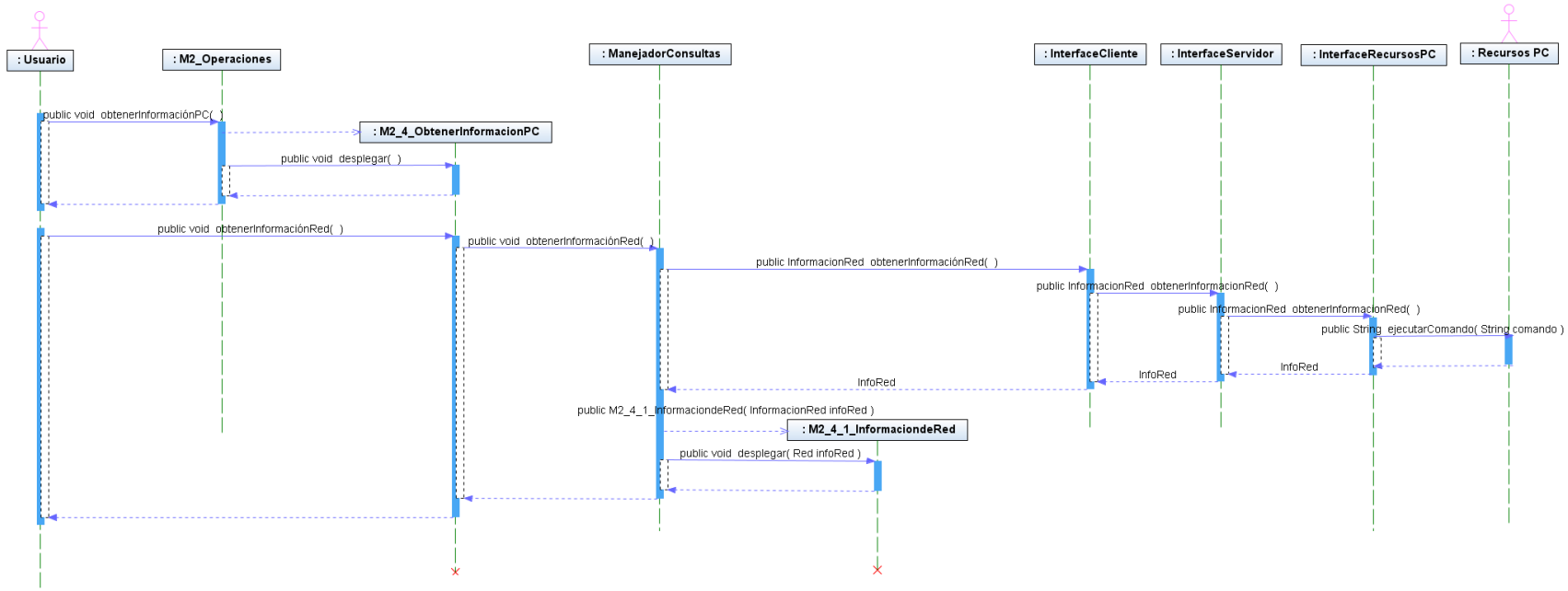


Figura 120. Diagrama de Secuencia Consultar información del Sistema

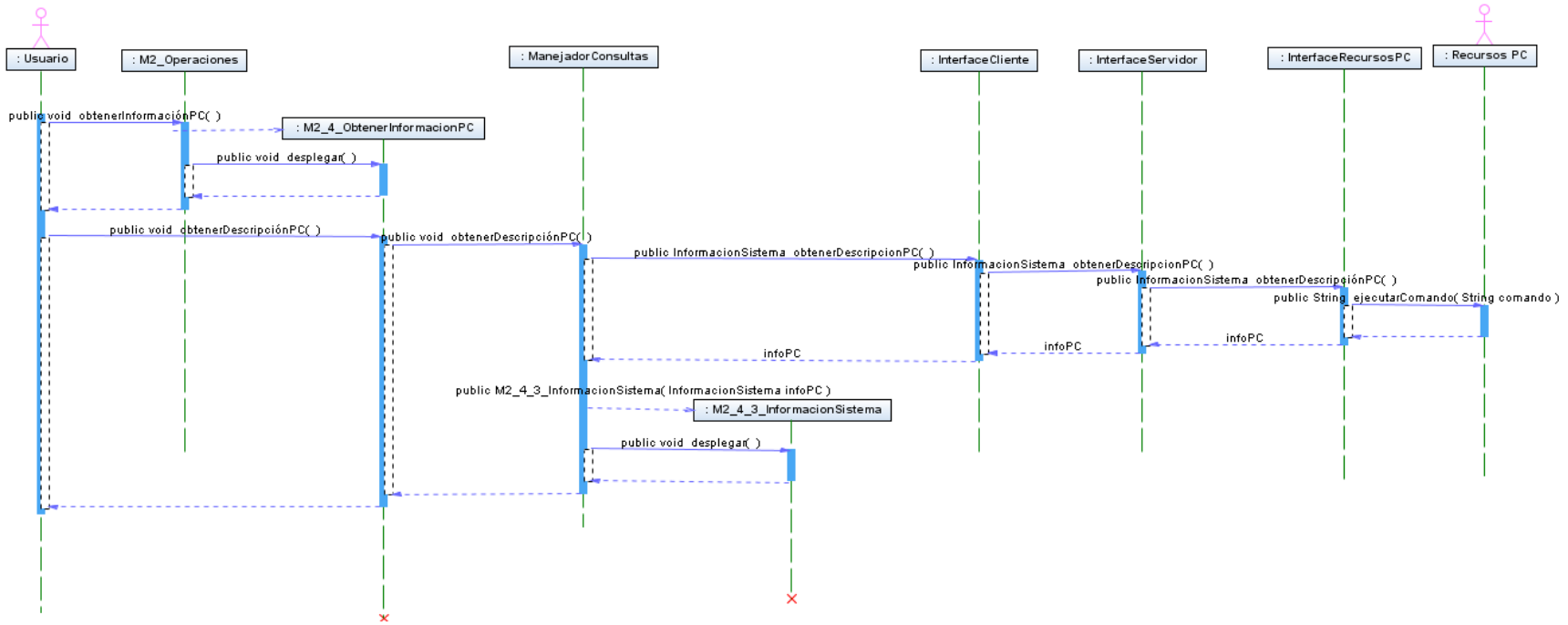


Figura 121. Diagrama de Secuencia Consultar Información Procesos

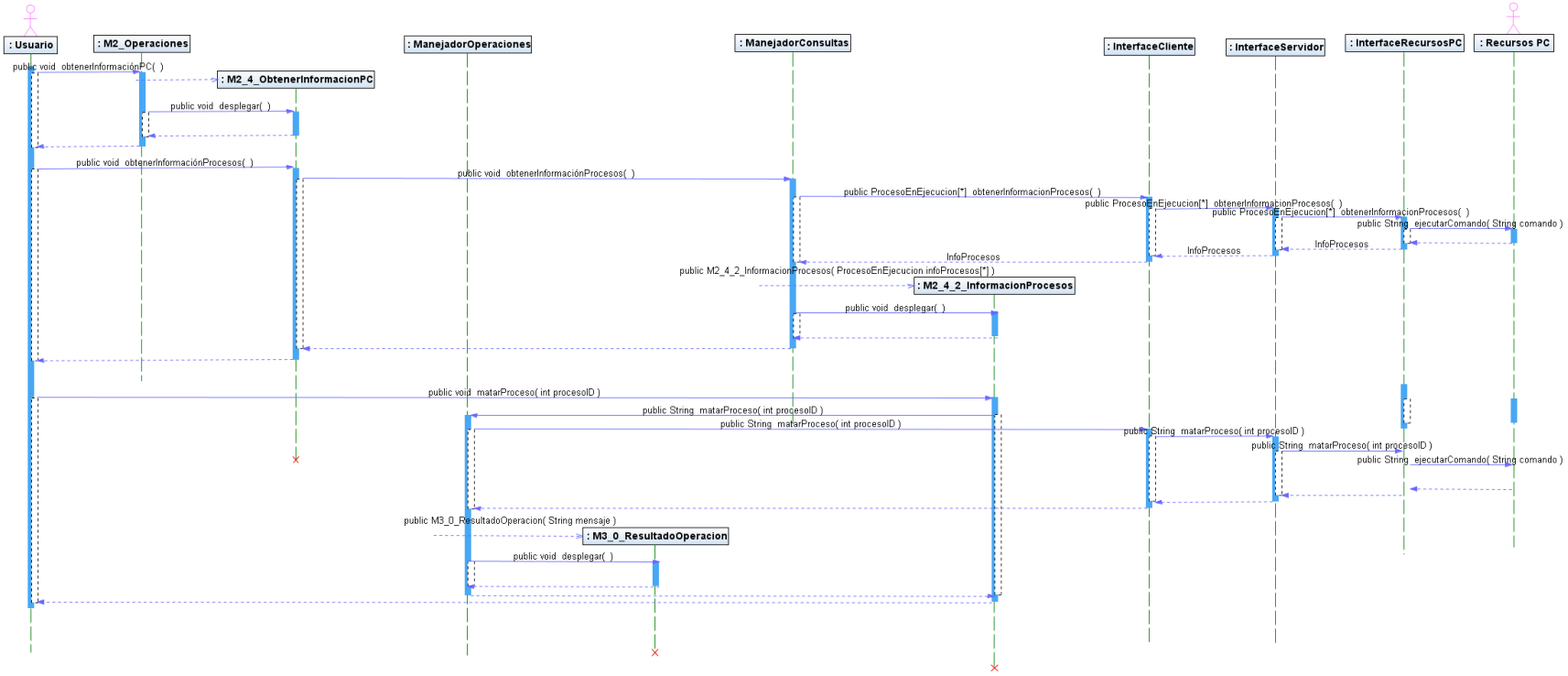


Figura 122. Diagrama de Secuencia Copiar

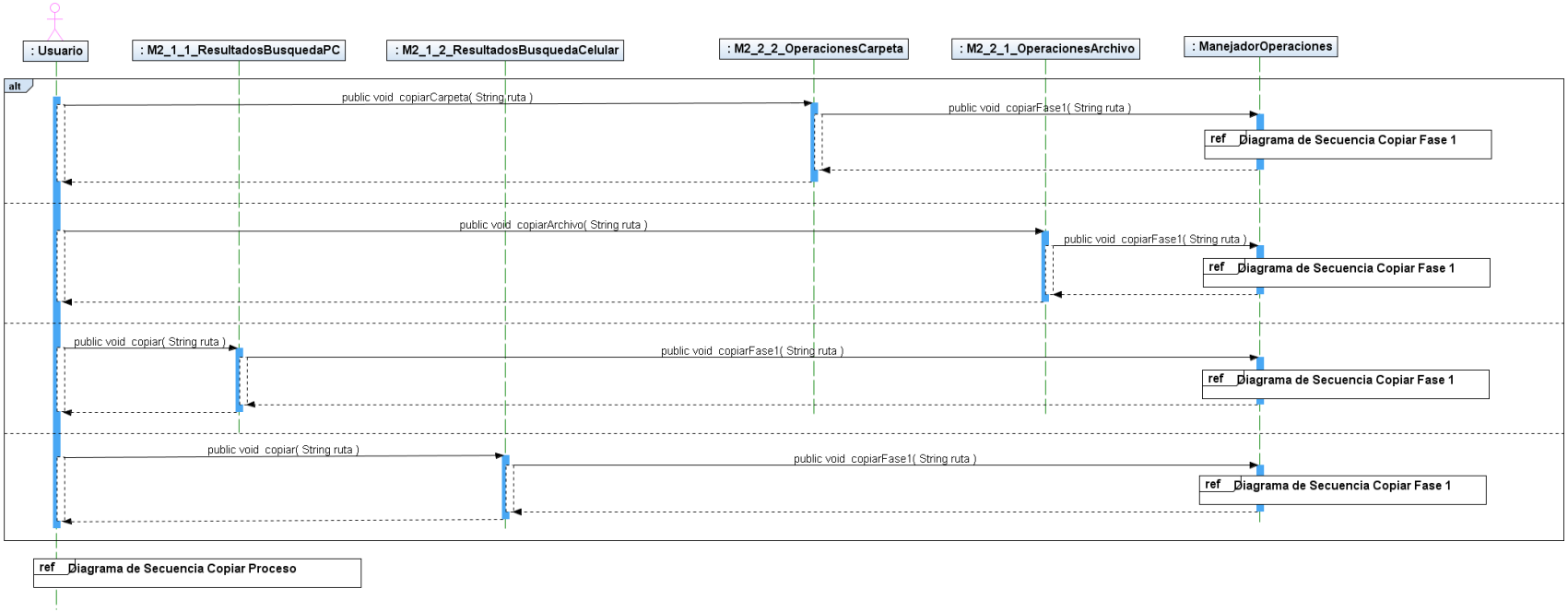


Figura 123. Diagrama de Secuencia Copiar Archivo de Celular a PC

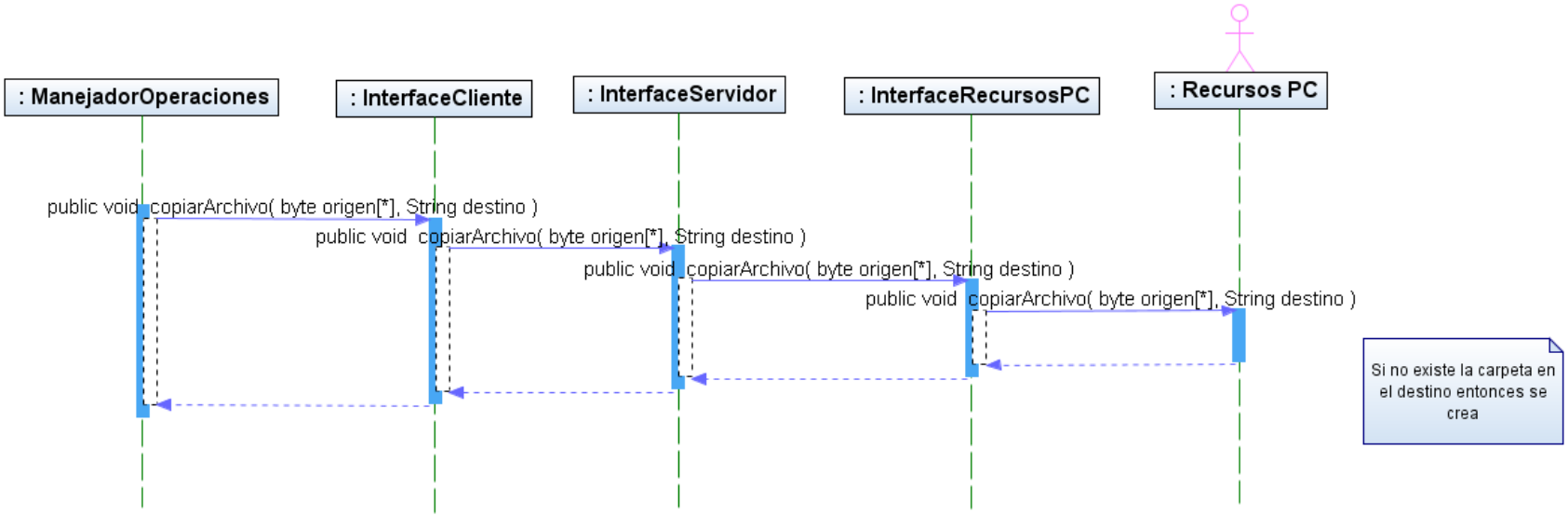


Figura 124. Diagrama de Secuencia Copiar Archivo de PC a Celular

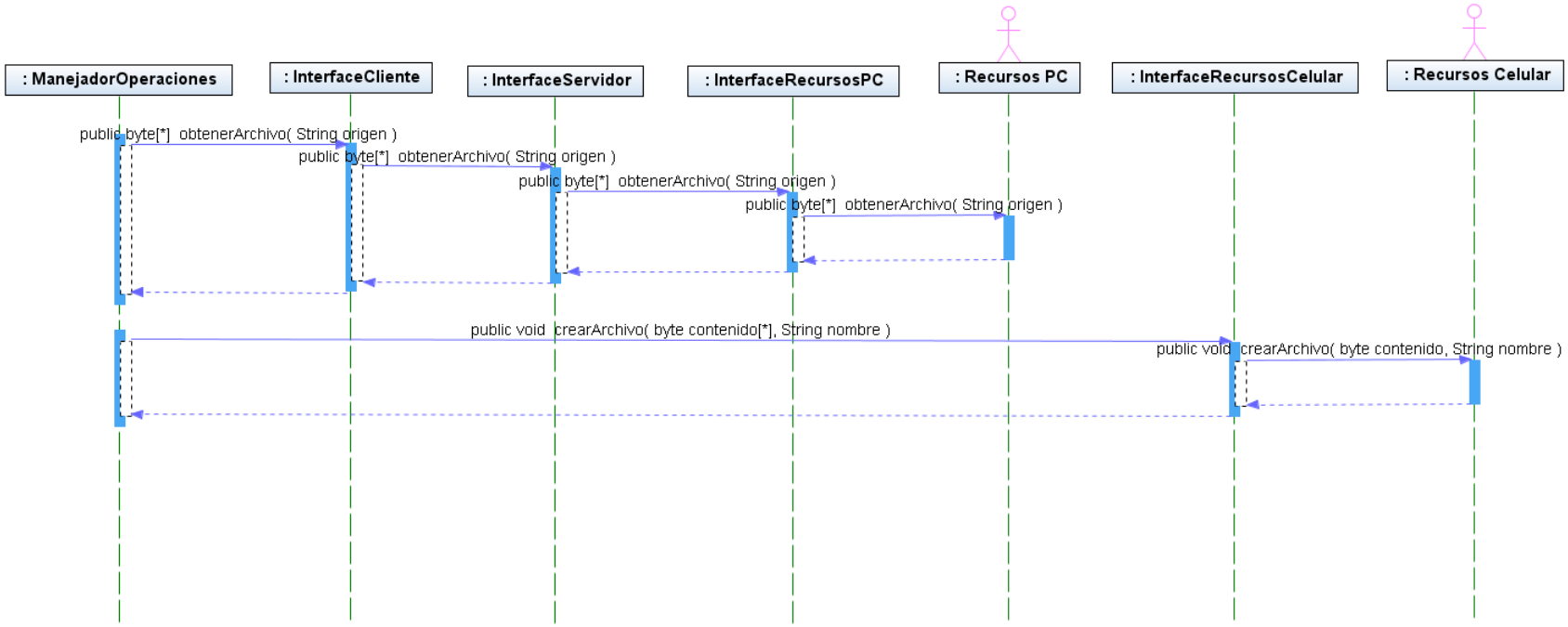


Figura 125. Diagrama de Secuencia Copiar Carpeta de Celular a PC

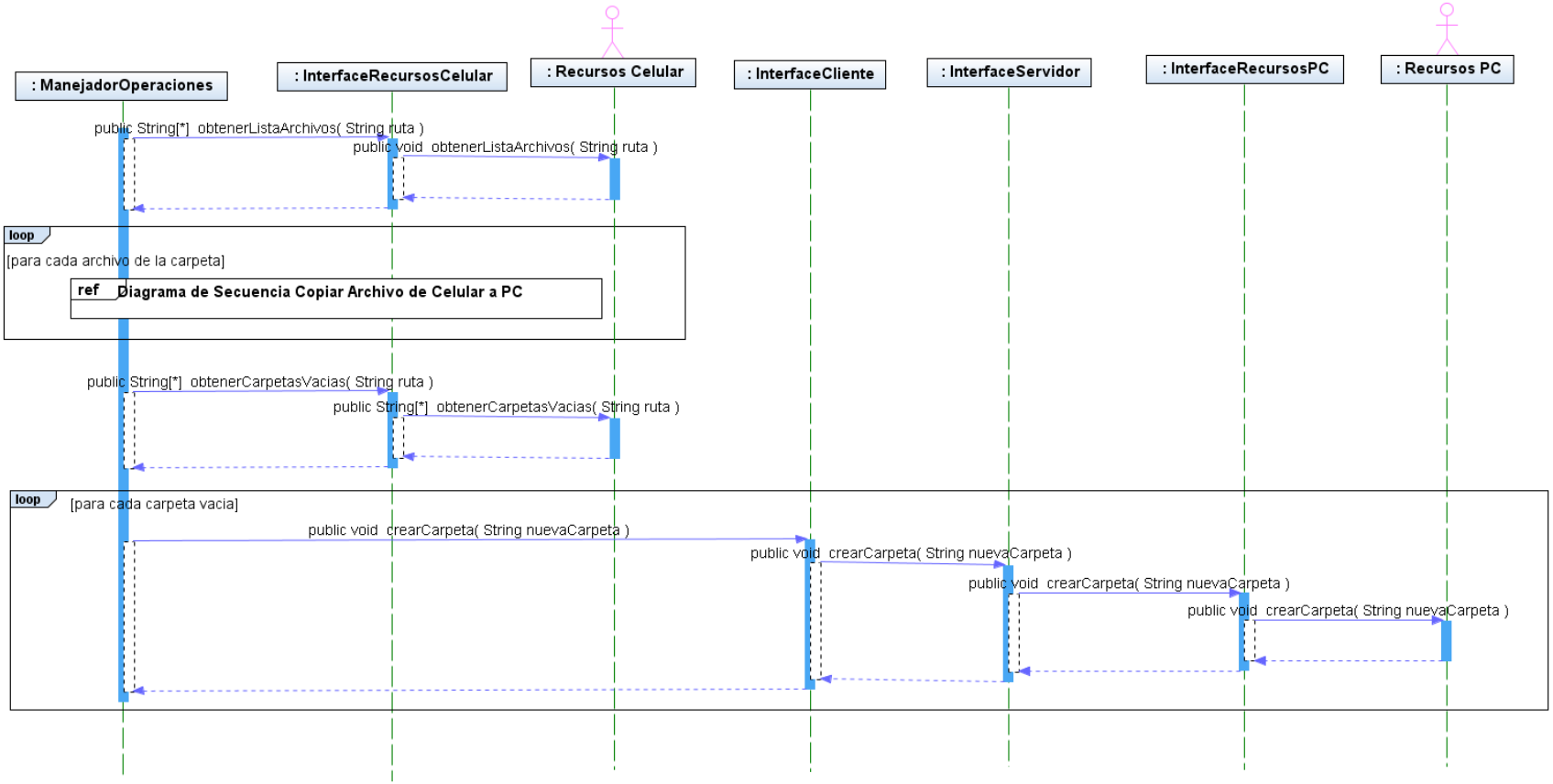


Figura 126. Diagrama de Secuencia Copiar Carpeta de PC a Celular

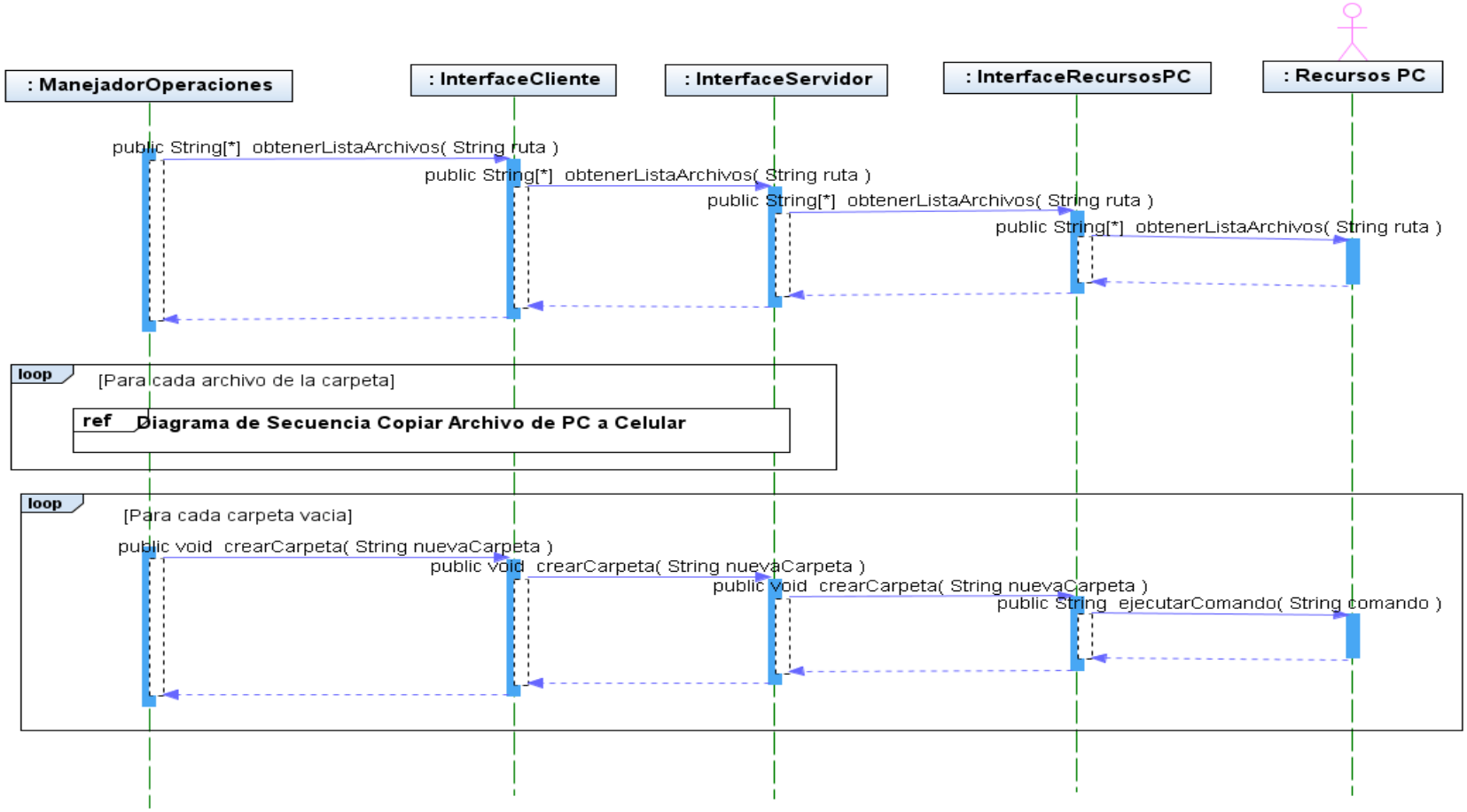


Figura 127. Diagrama de Secuencia Copiar de Celular a Celular

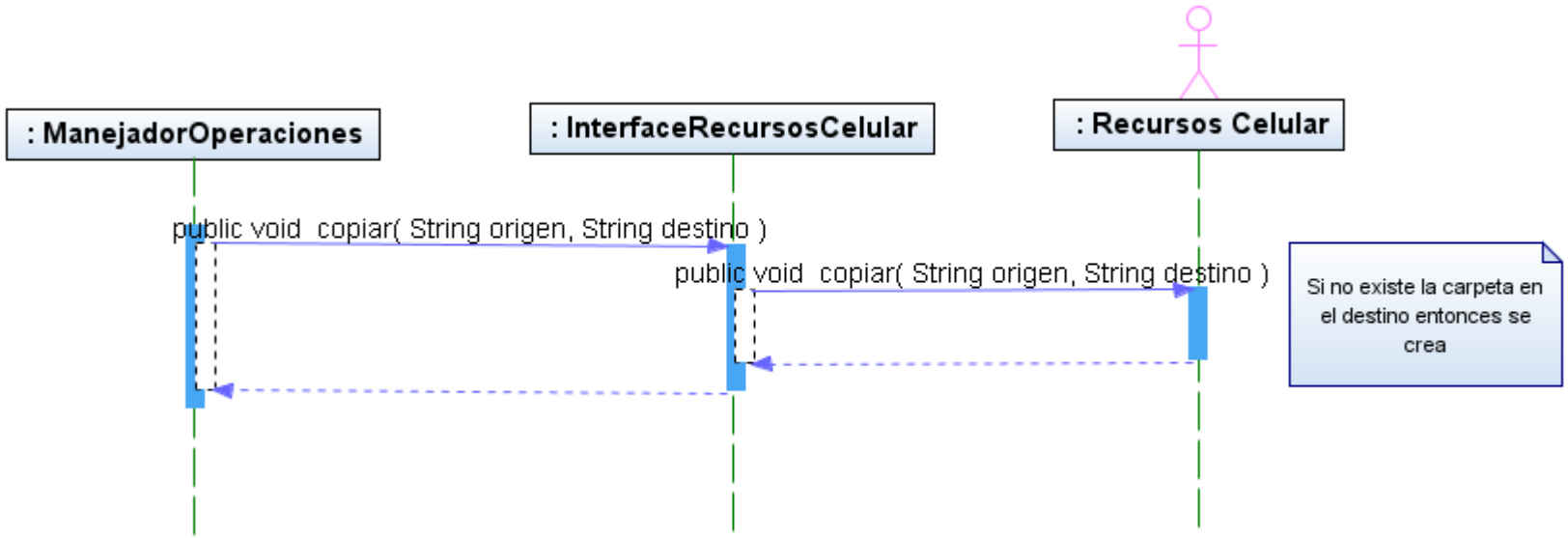


Figura 128. Diagrama de Secuencia Copiar de PC a PC

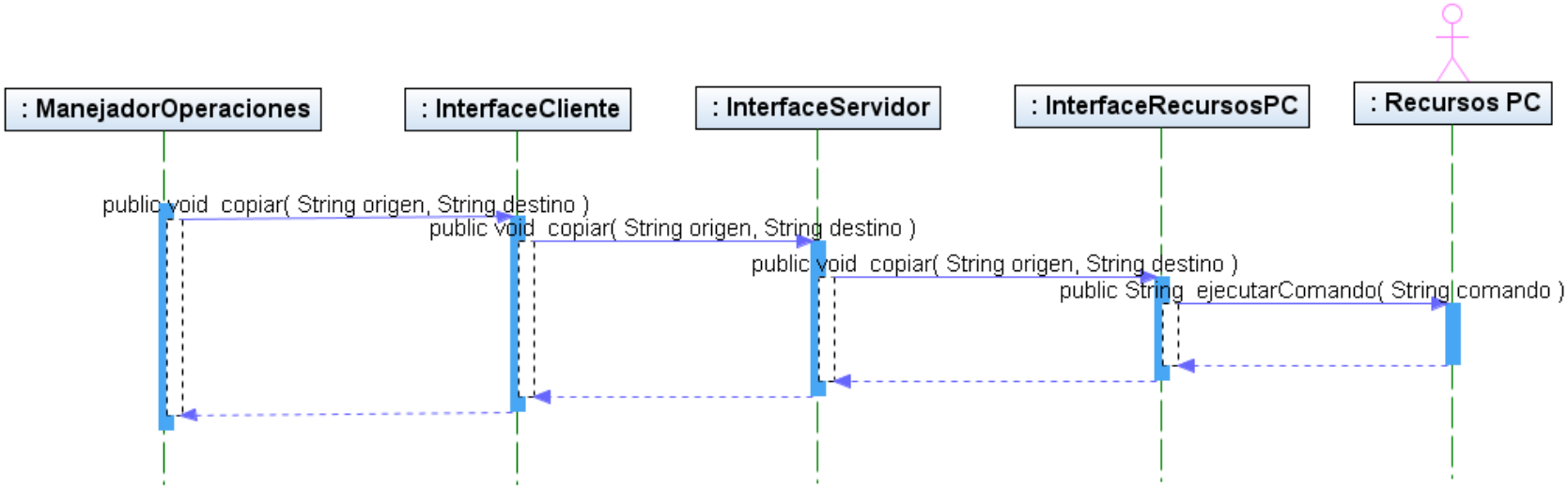


Figura 129. Diagrama de Secuencia Copiar Fase 1

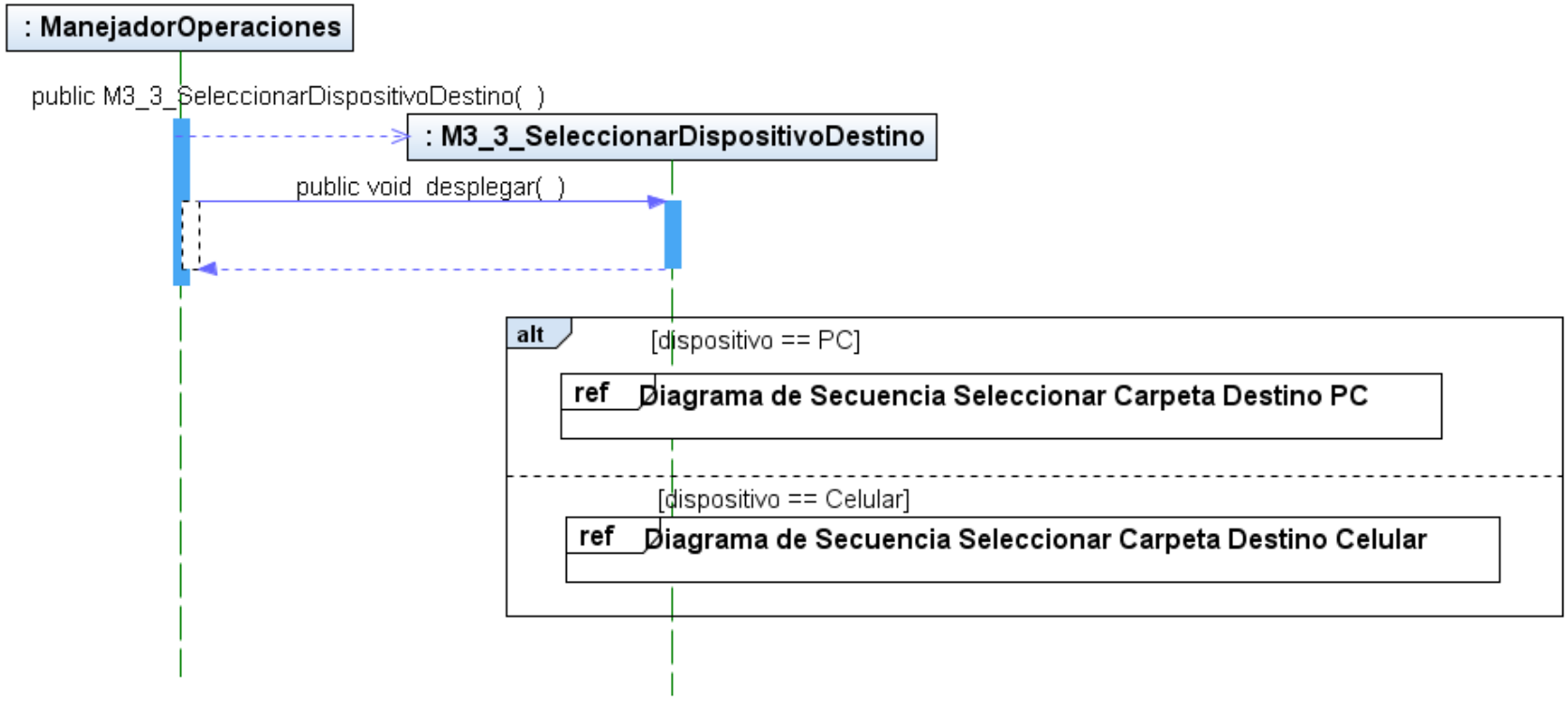


Figura 130. Diagrama de Secuencia Copiar Proceso

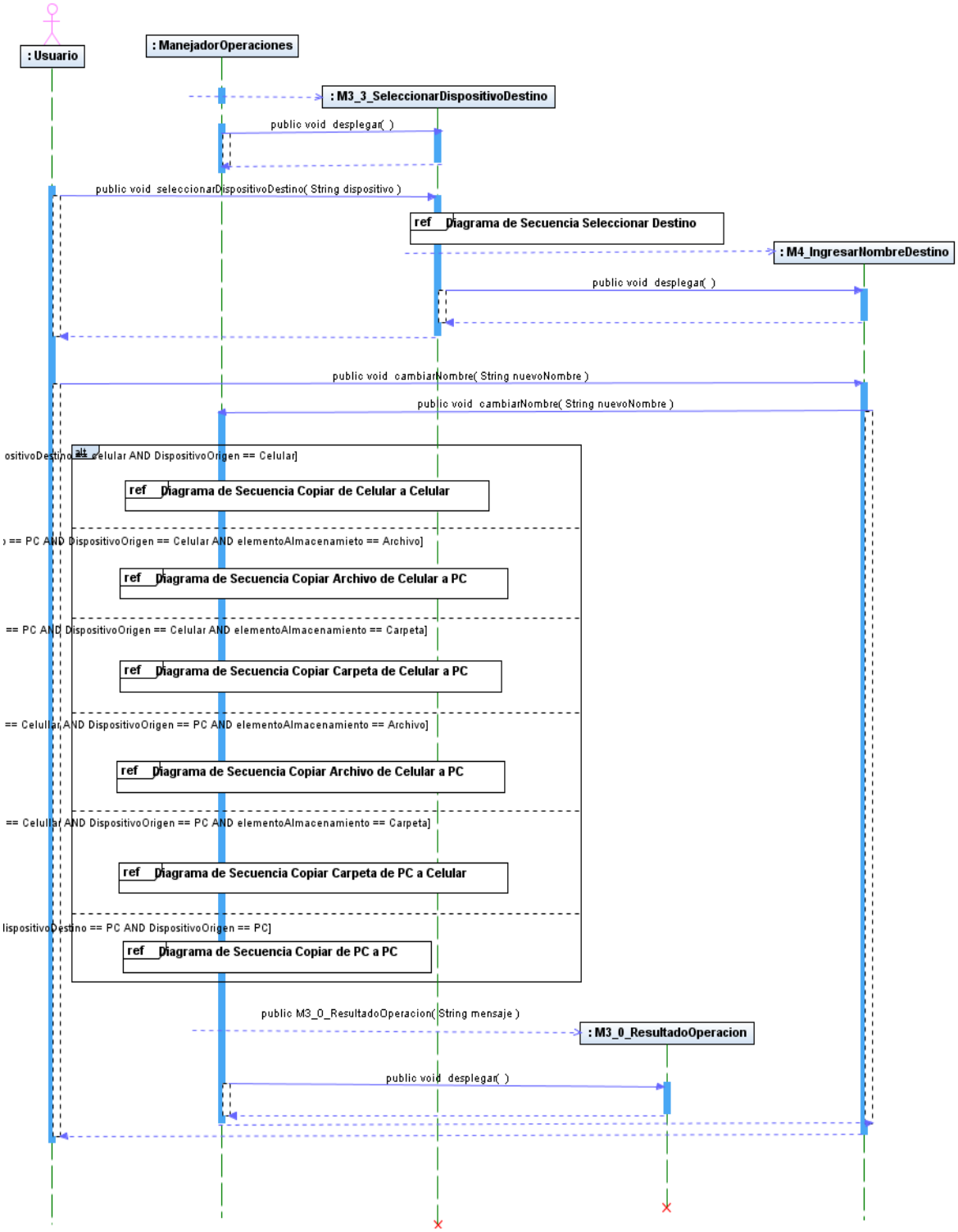


Figura 131. Diagrama de Secuencia Crear Carpeta

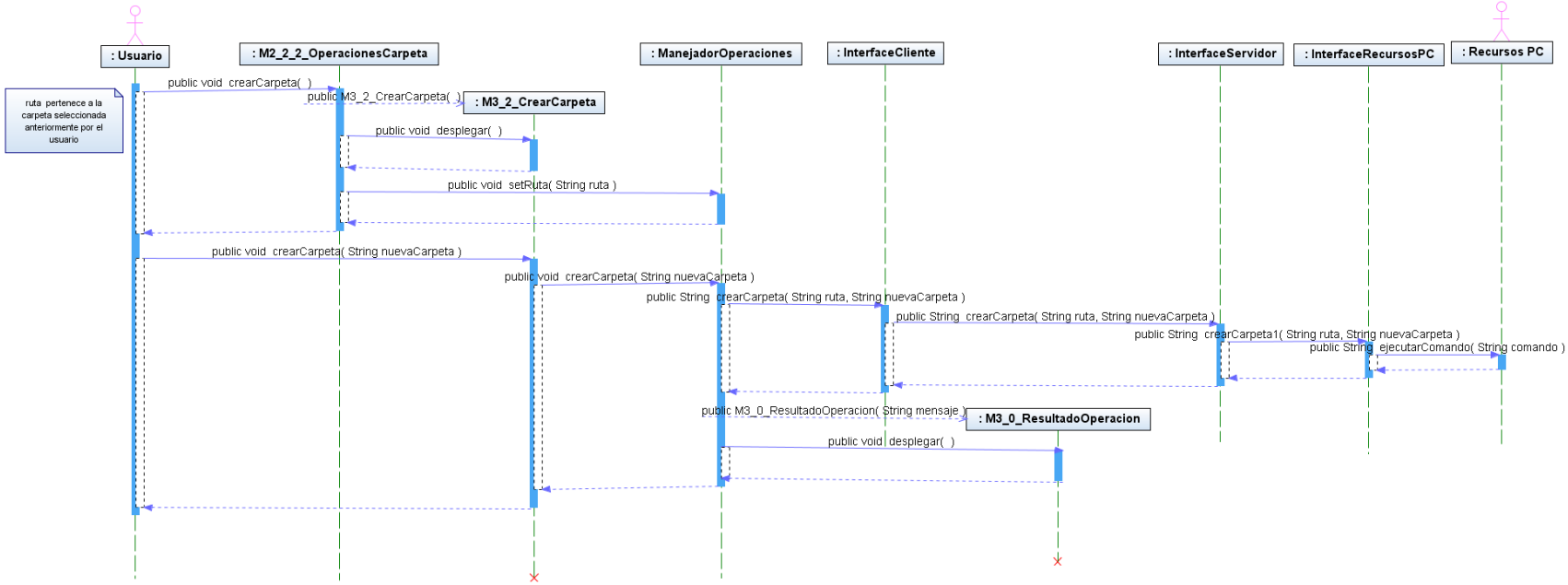


Figura 132. Diagrama de Secuencia Ejecutar Comando Consola

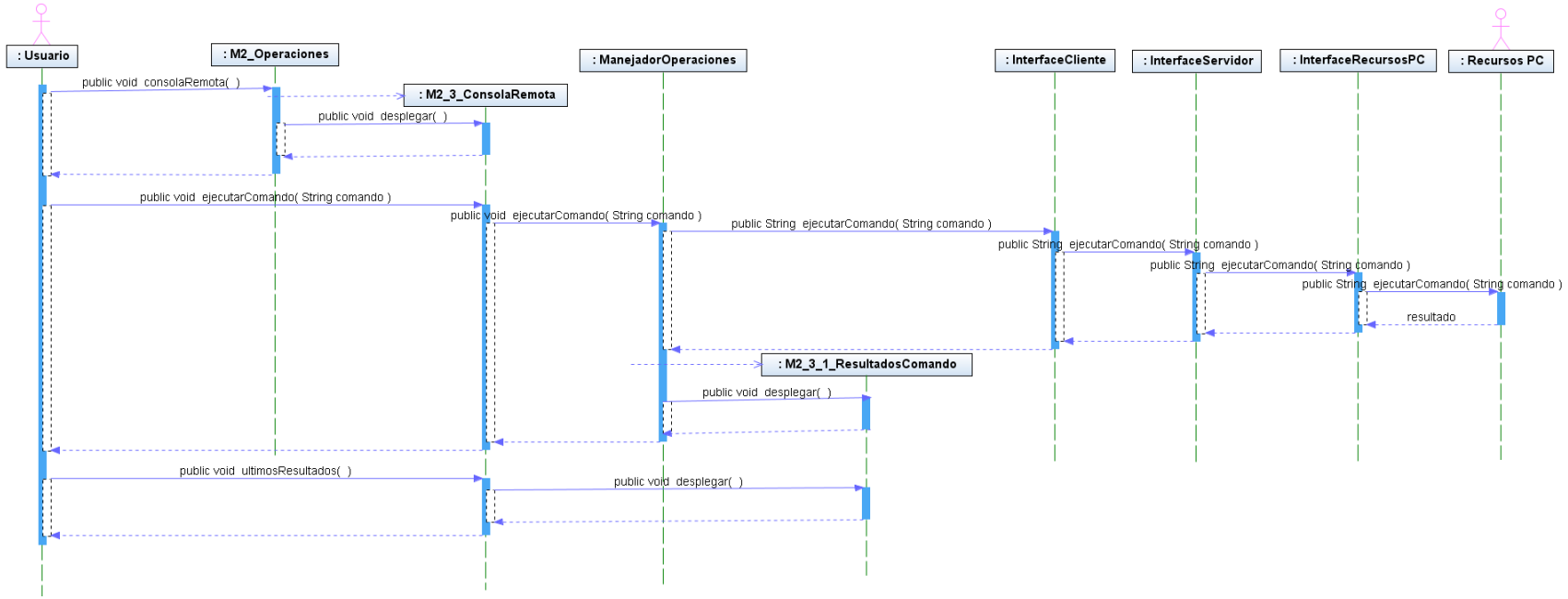


Figura 133. Diagrama de Secuencia Eliminar Archivo

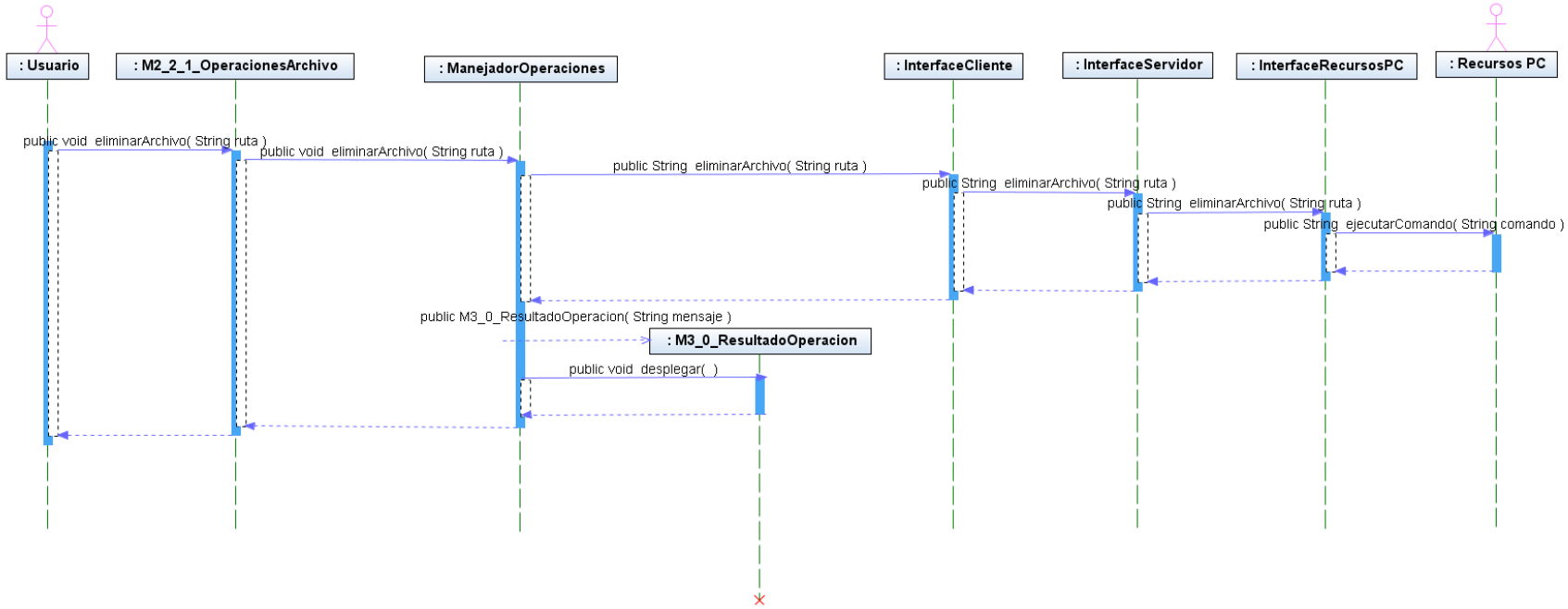


Figura 134. Diagrama de Secuencia Eliminar Carpeta

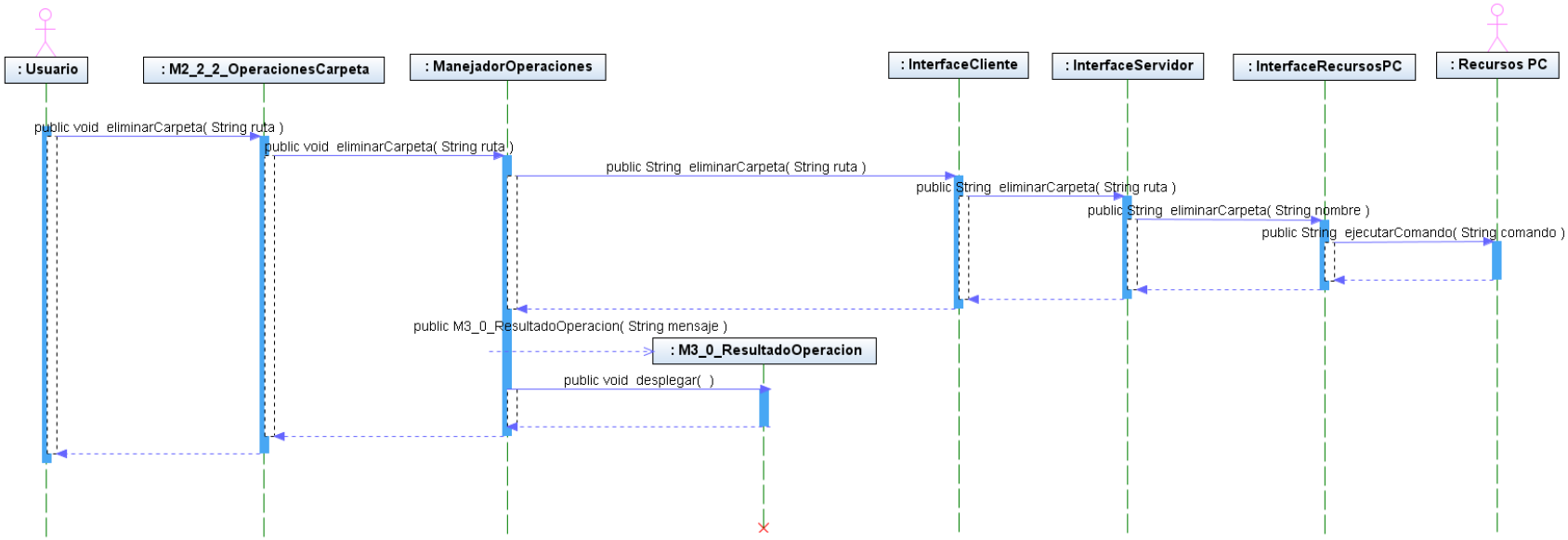


Figura 135. Diagrama de Secuencia Eliminar Origen

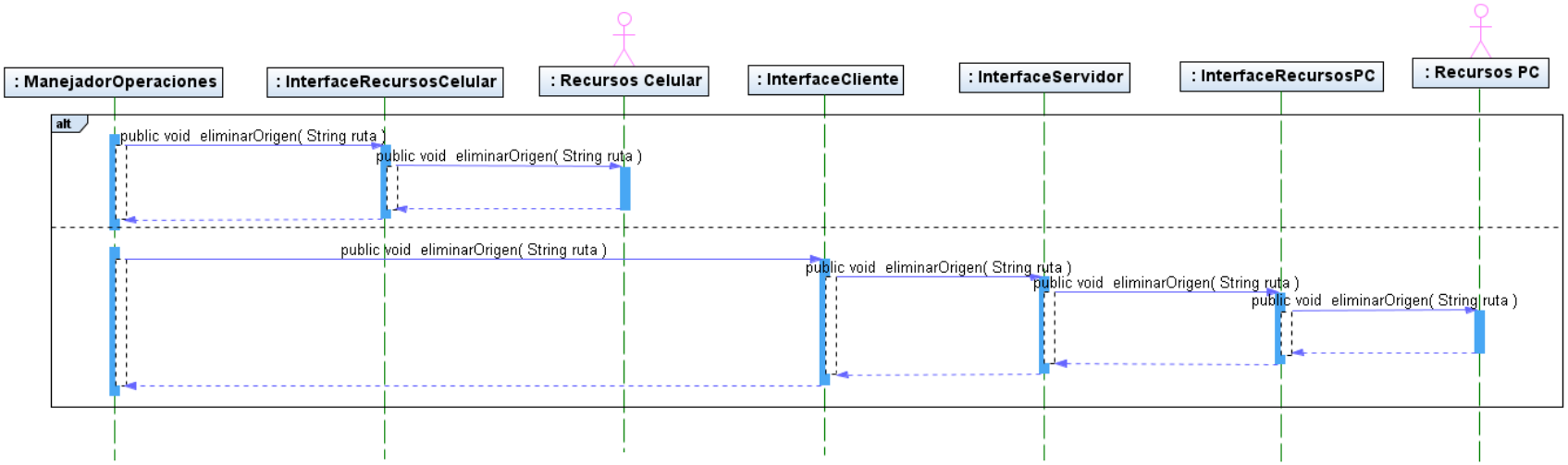


Figura 136. Diagrama de Secuencia Eliminar PC

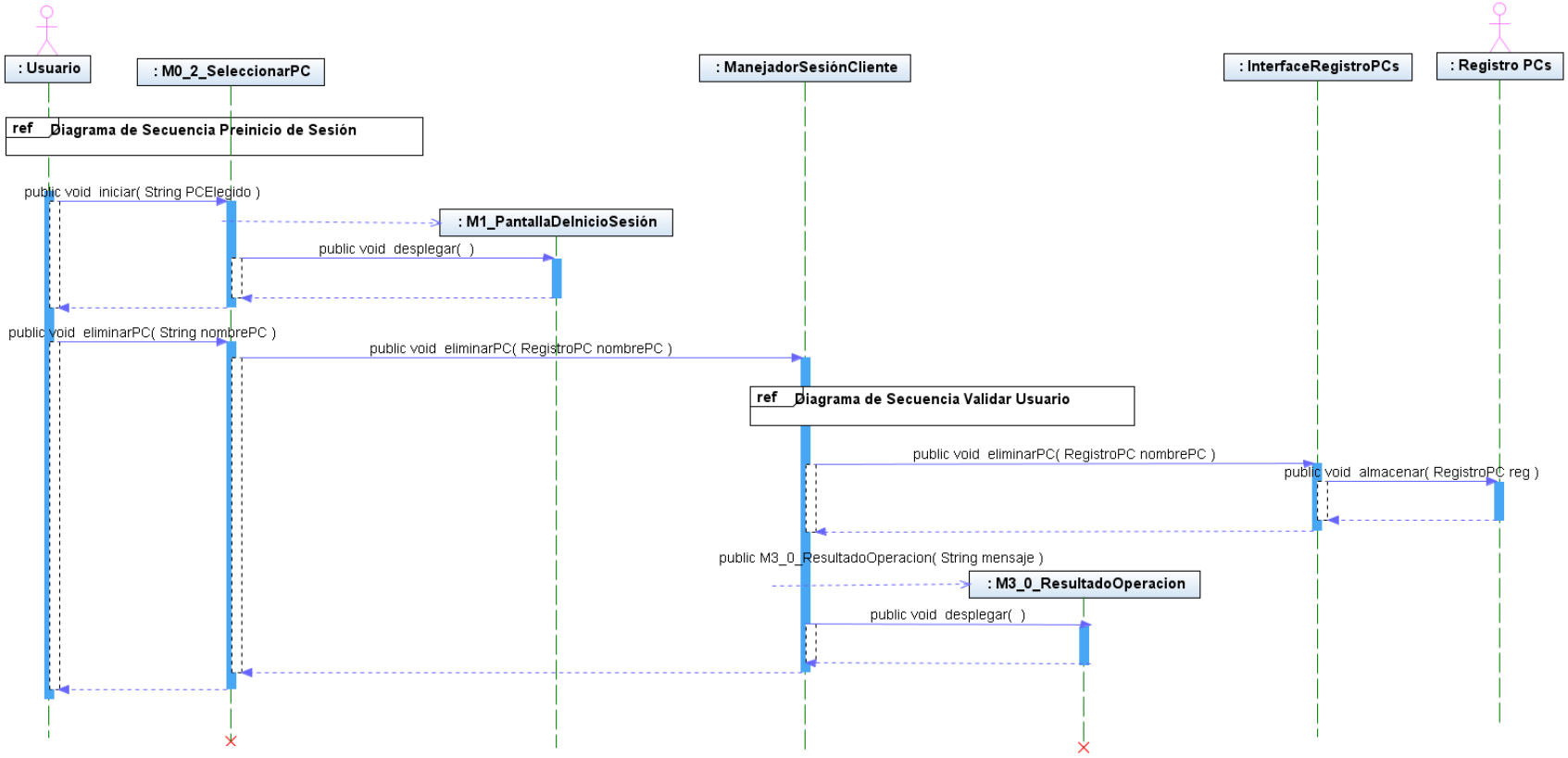


Figura 137. Diagrama de Secuencia Eliminar Usuario

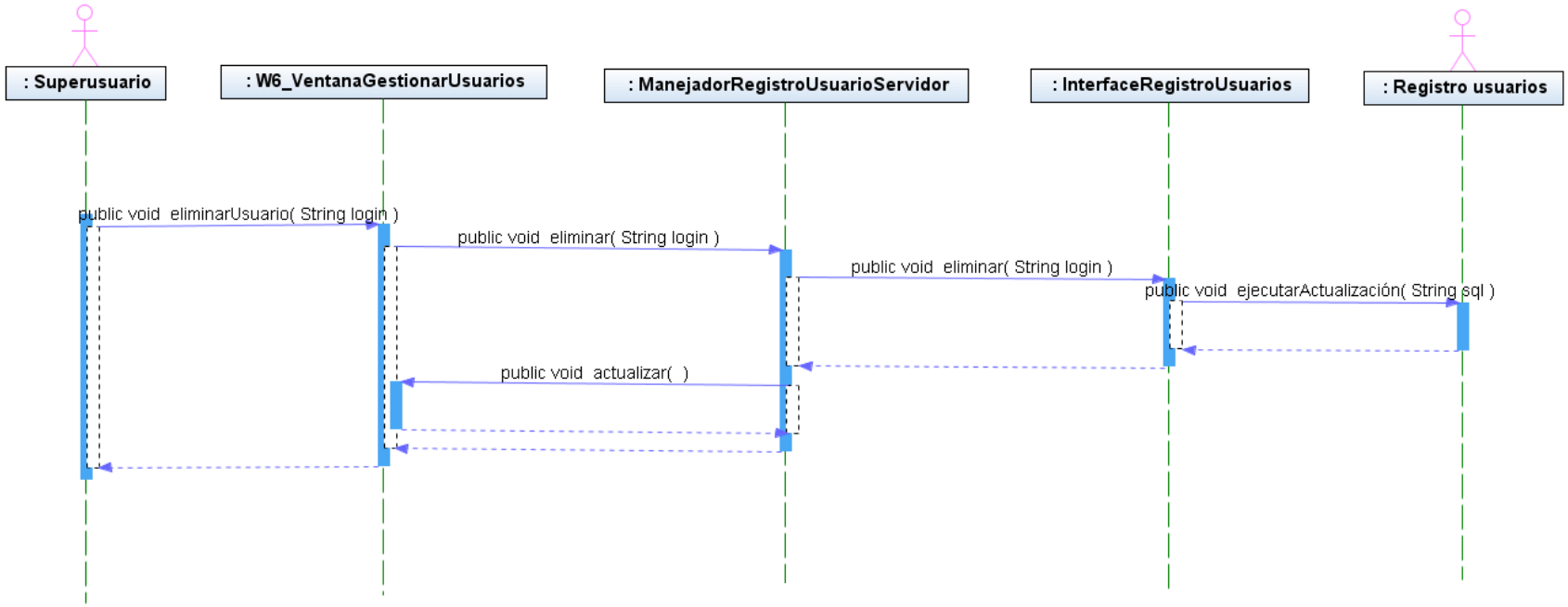


Figura 138. Diagrama de Secuencia Gestionar Servidor

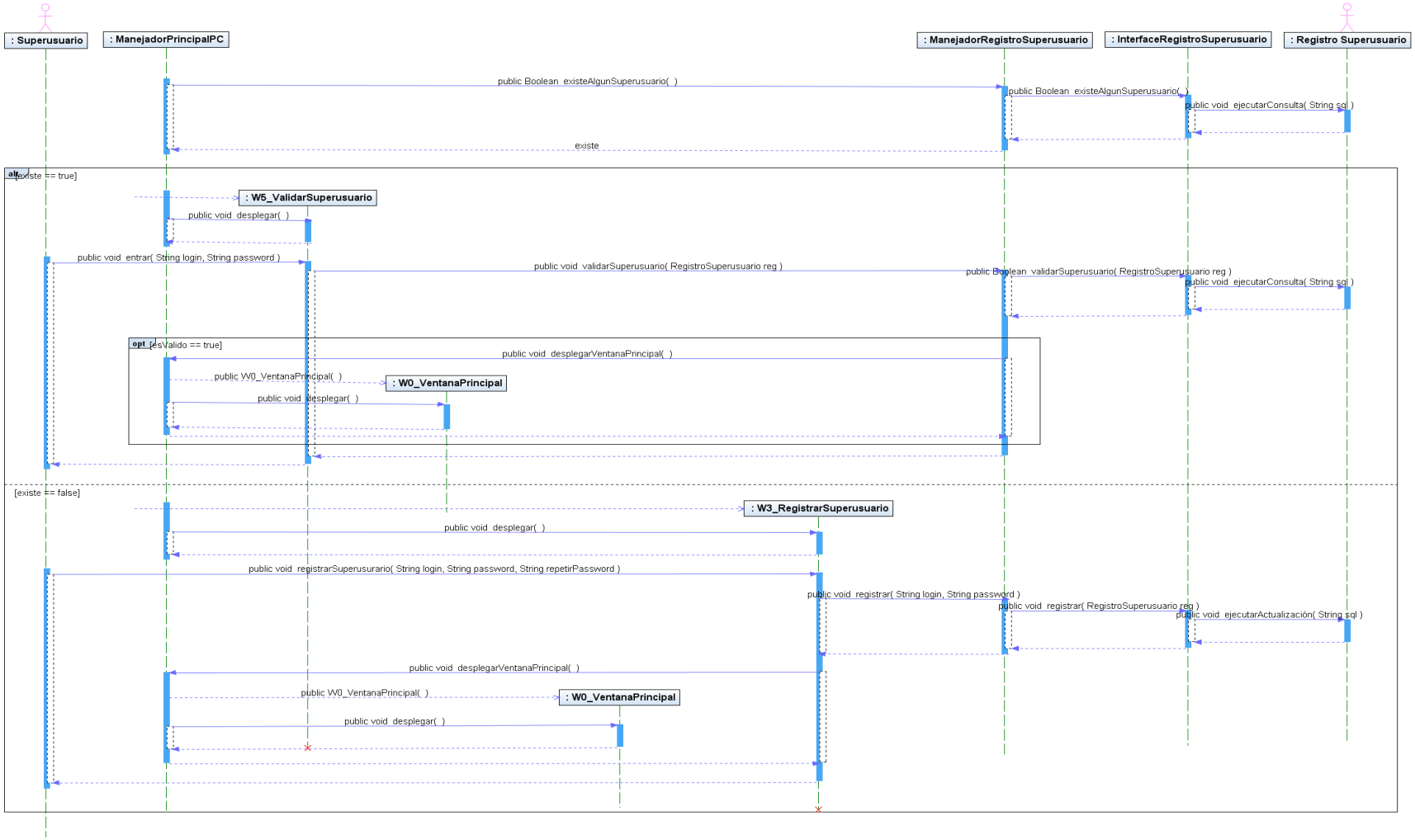


Figura 139. Diagrama de Secuencia Gestionar Usuarios

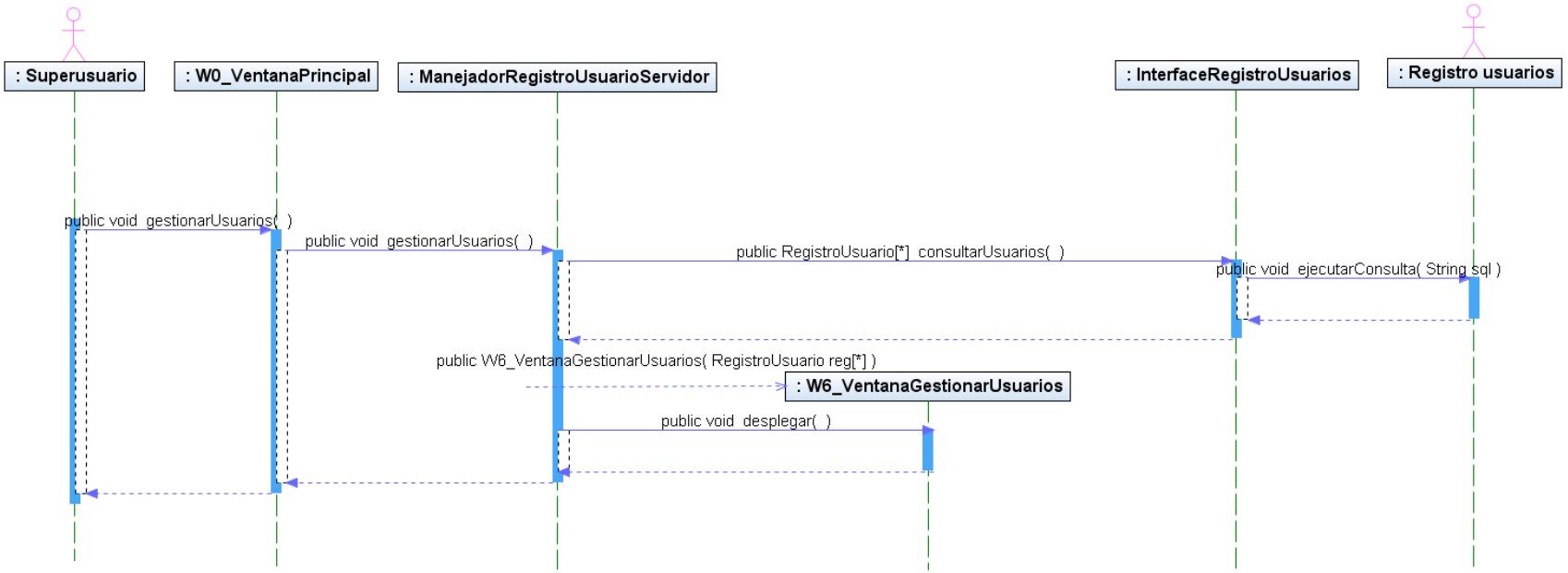


Figura 140. Diagrama de Secuencia Iniciar Servicio

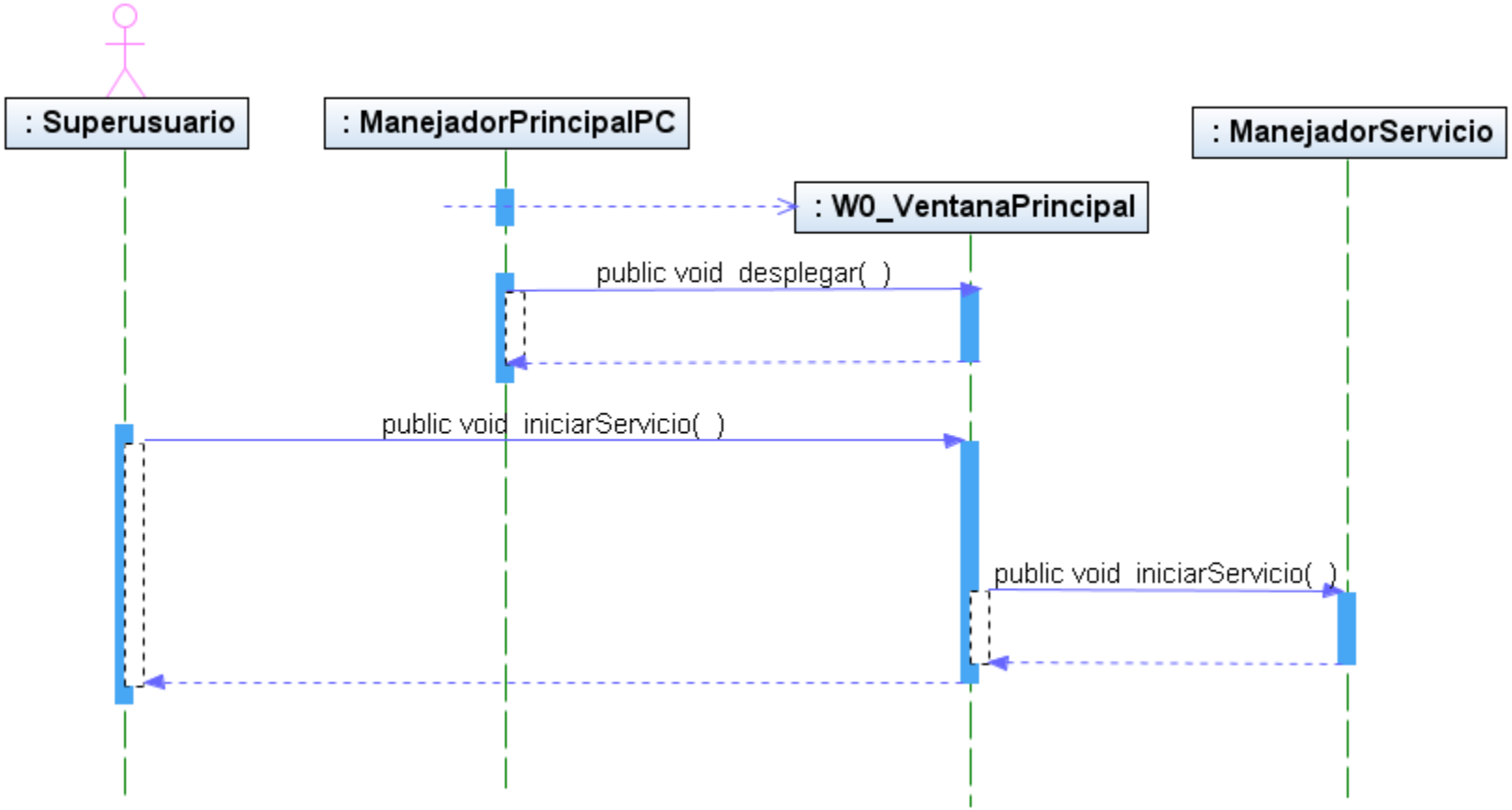


Figura 141. Diagrama de Secuencia Iniciar Sesión

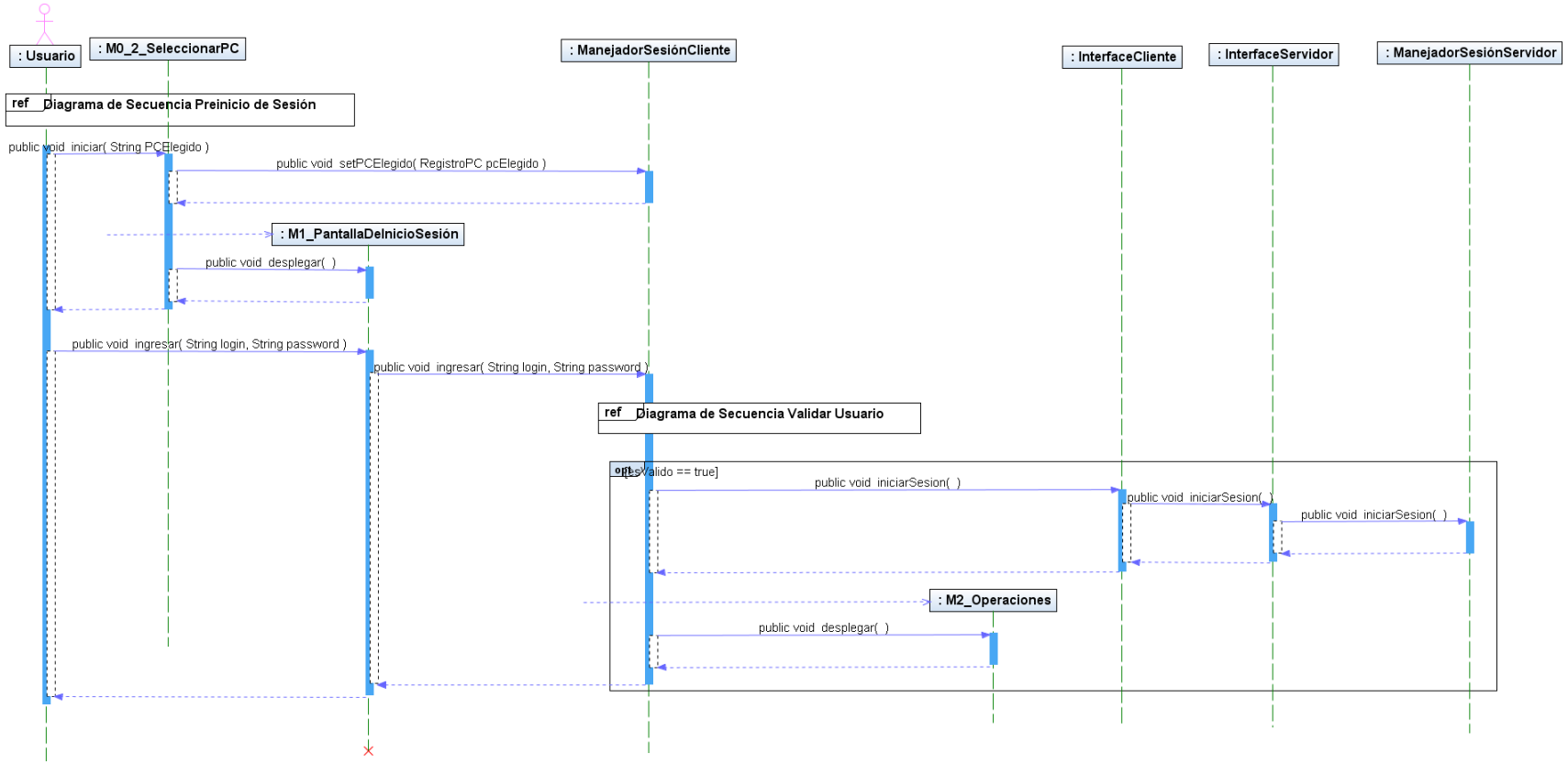


Figura 142. Diagrama de Secuencia Iniciar Sesión por Defecto

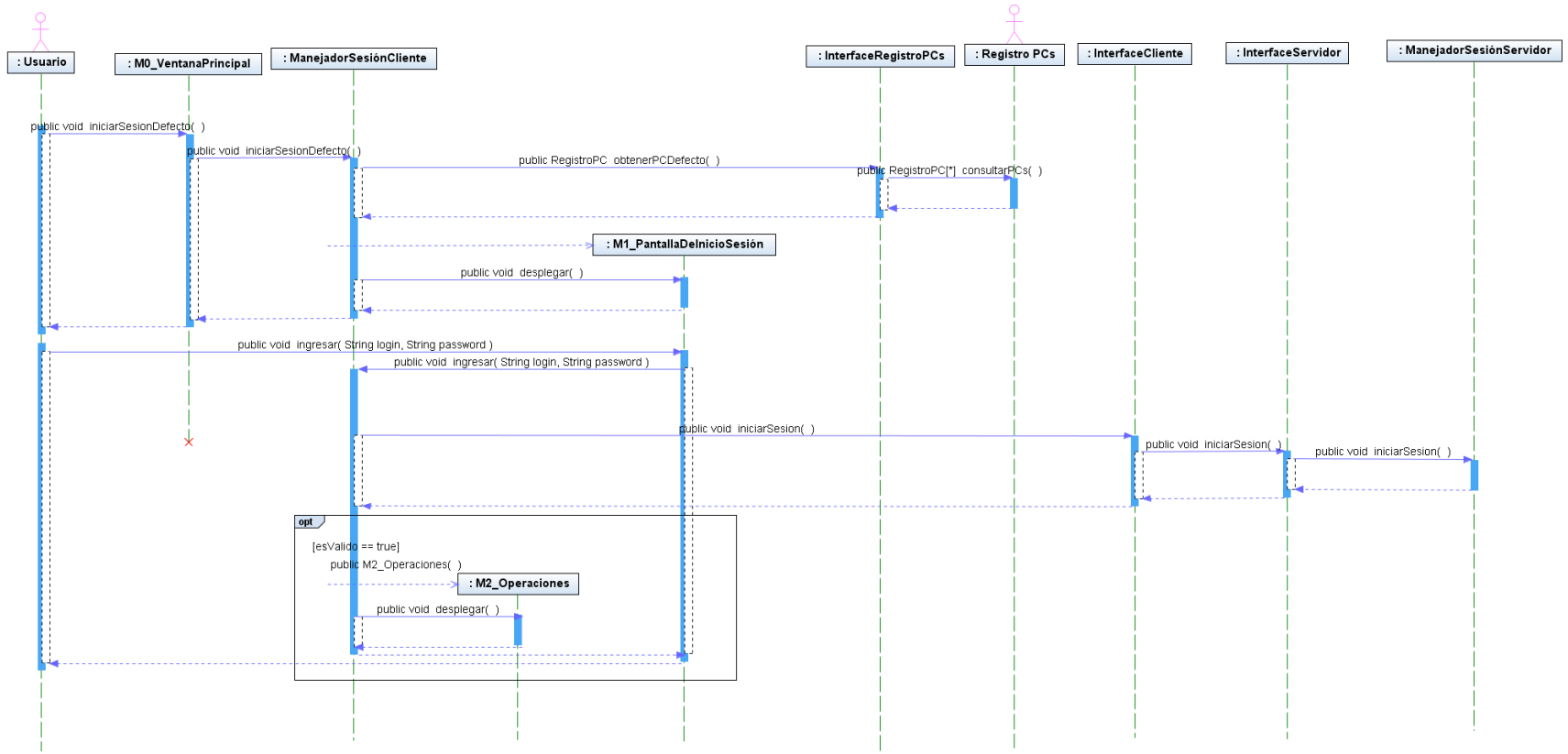


Figura 143. Diagrama de Secuencia Inicio Cliente

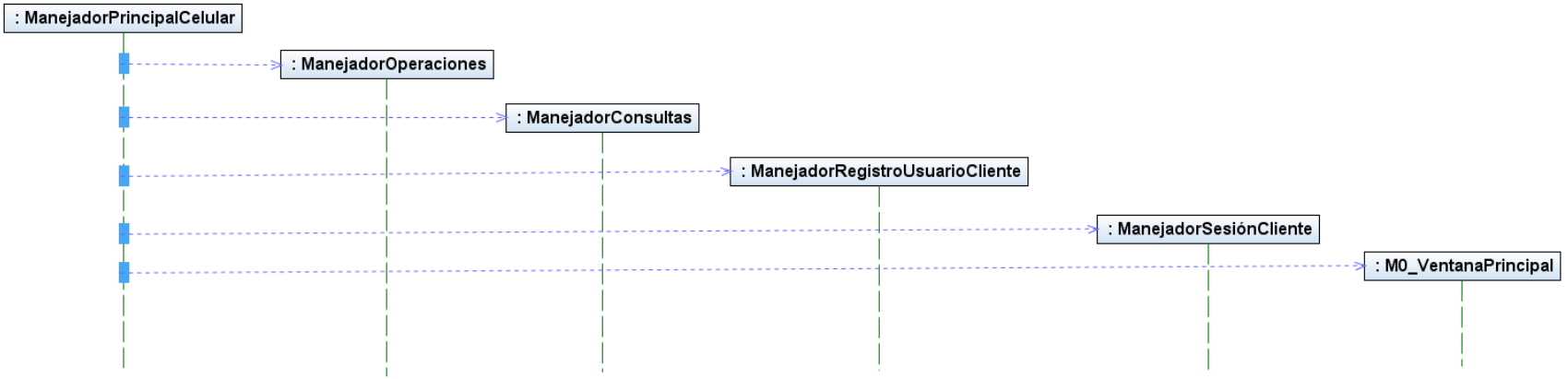


Figura 144. Diagrama de Secuencia Inicio Servidor

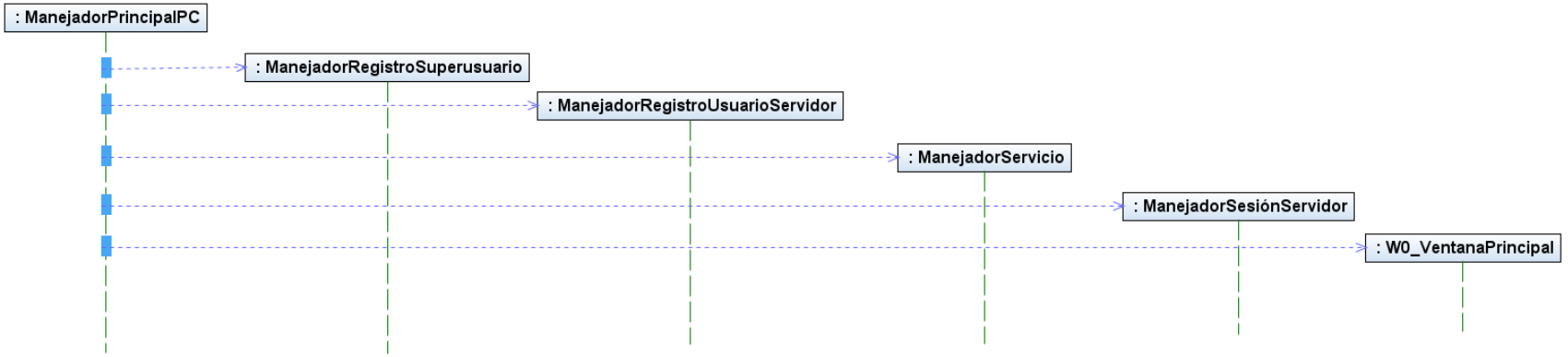


Figura 145. Diagrama de Secuencia Modificar Password

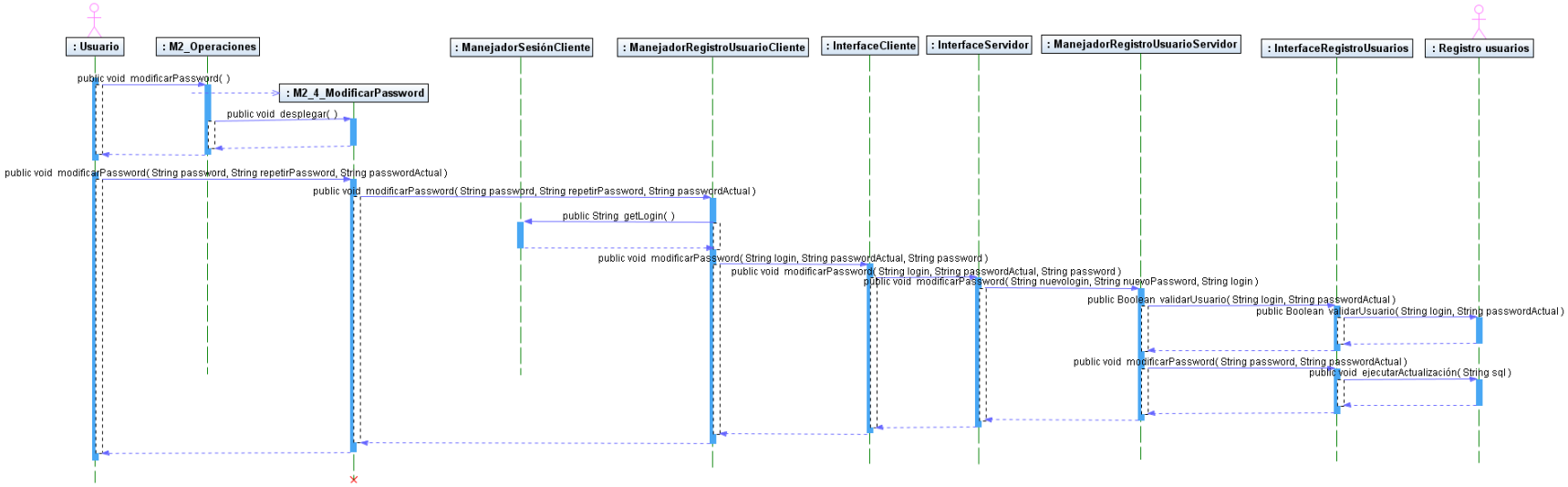


Figura 146. Diagrama de Secuencia Modificar Superusuario

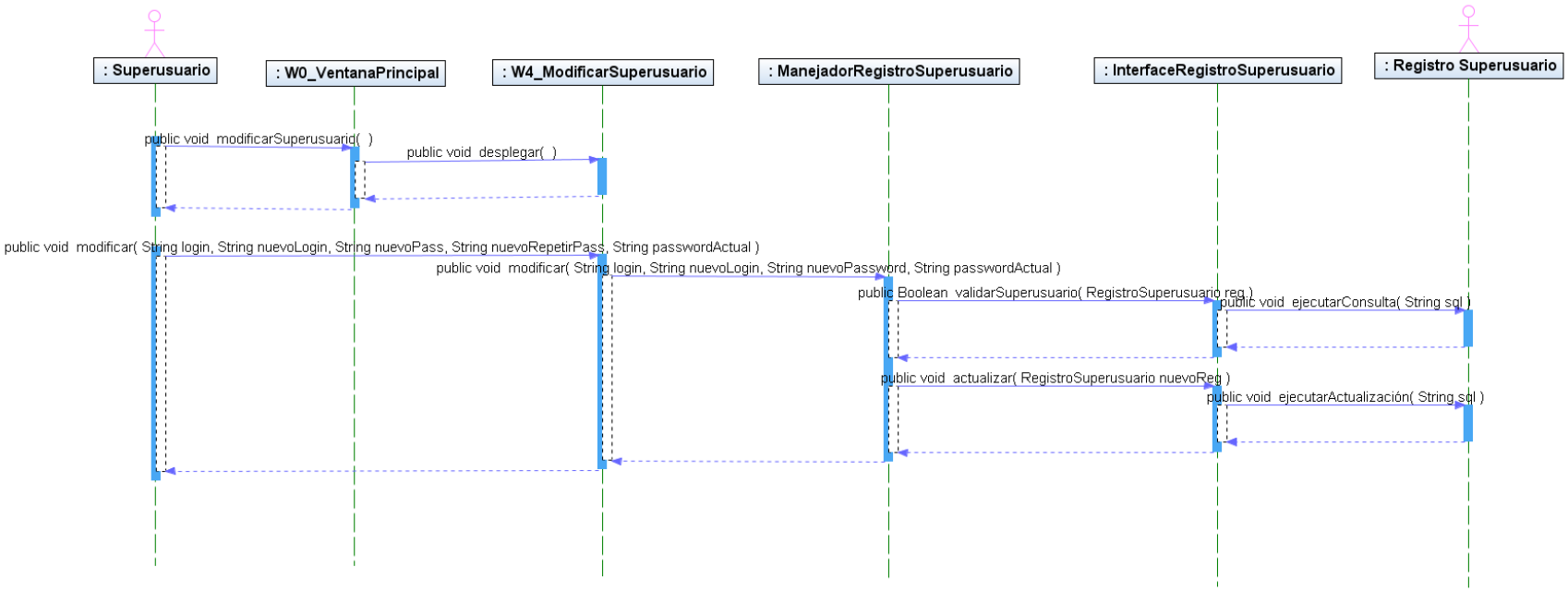


Figura 147. Diagrama de Secuencia Modificar Usuario

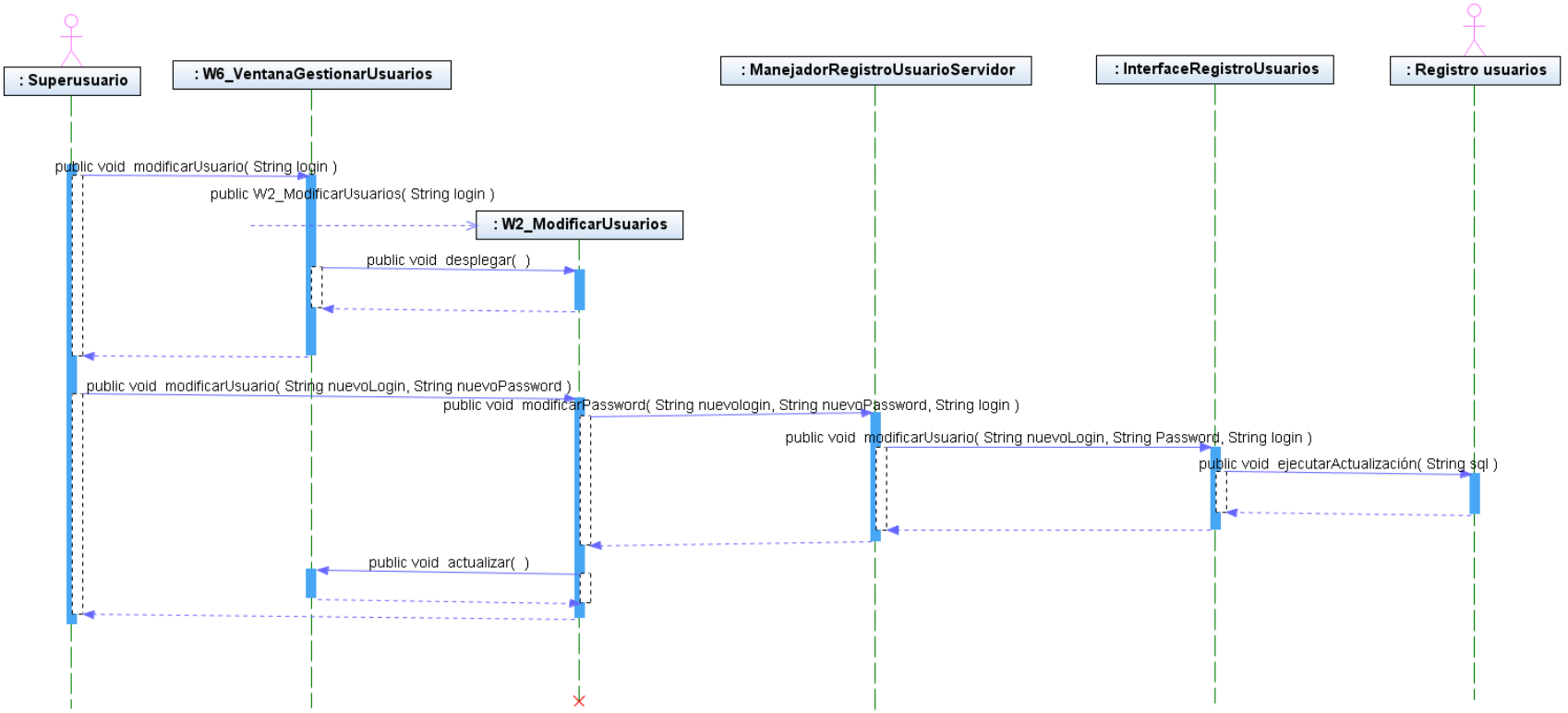


Figura 148. Diagrama de Secuencia Mover

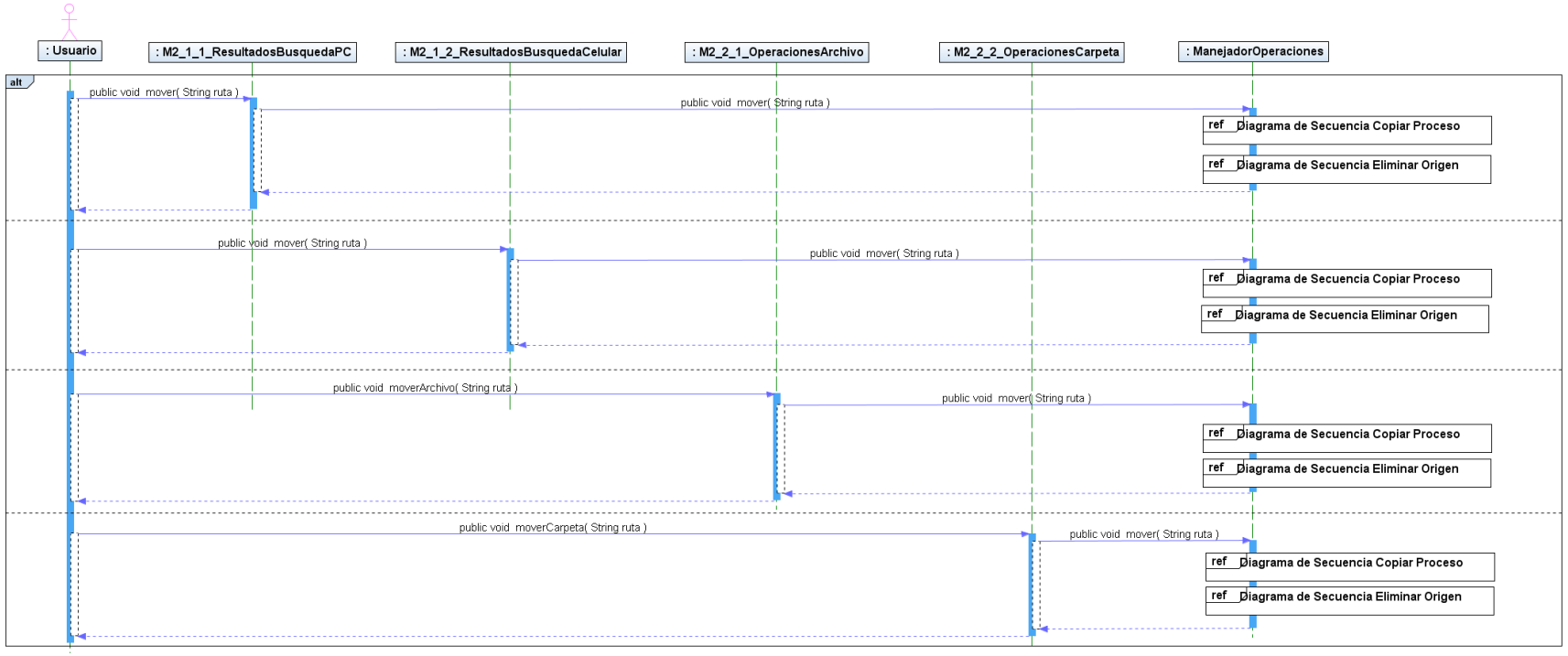


Figura 149. Diagrama de Secuencia Predeterminar PC

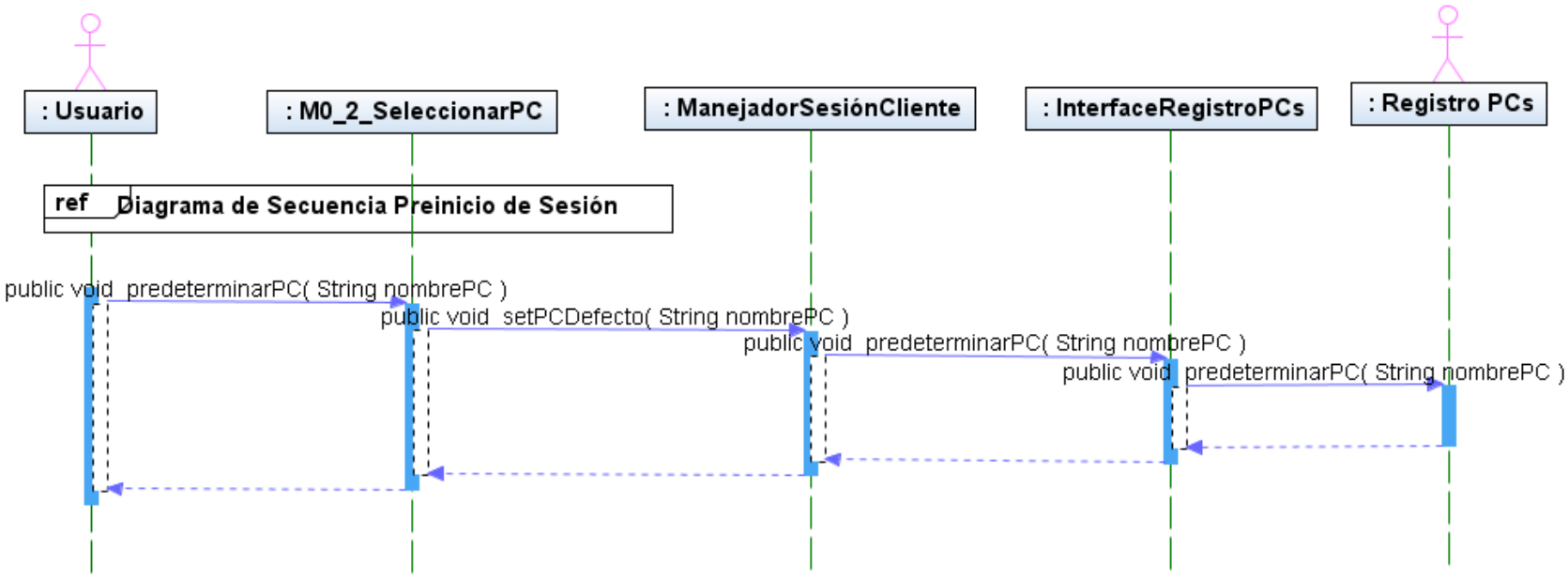


Figura 150. Diagrama de Secuencia Preinicio de Sesión

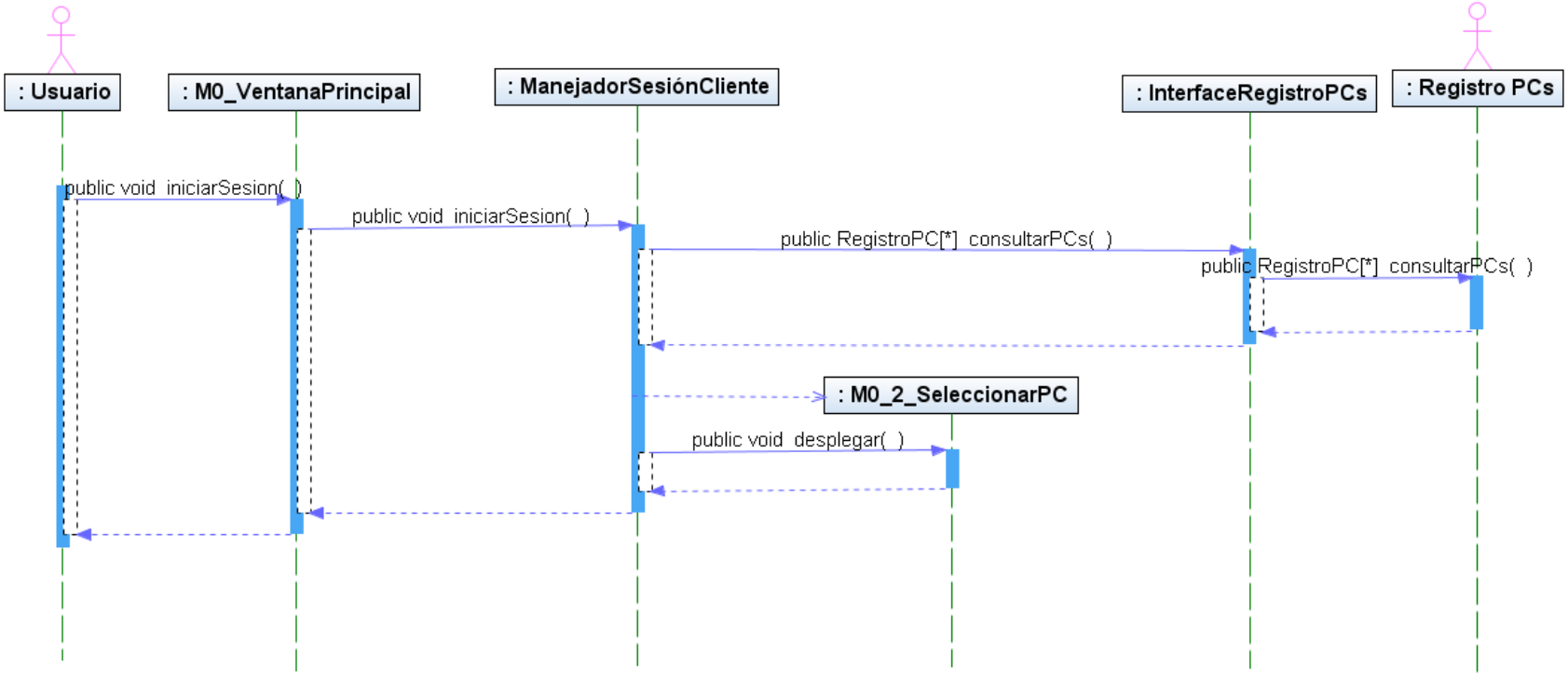


Figura 151. Diagrama de Secuencia Registrar usuario

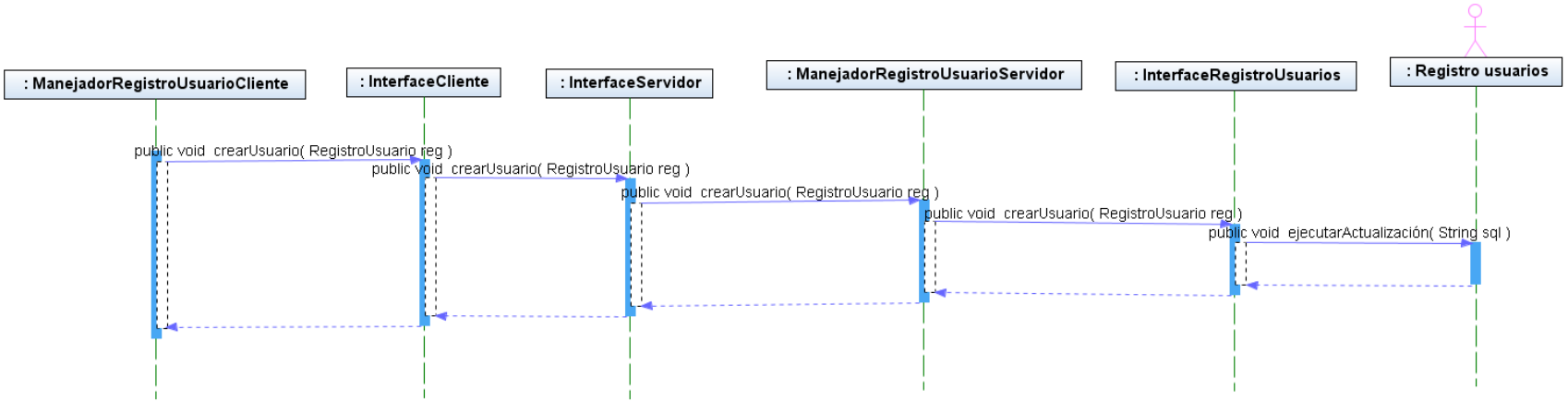


Figura 152. Diagrama de Secuencia Renombrar

Figura 153. Diagrama de Secuencia Renombrar Proceso

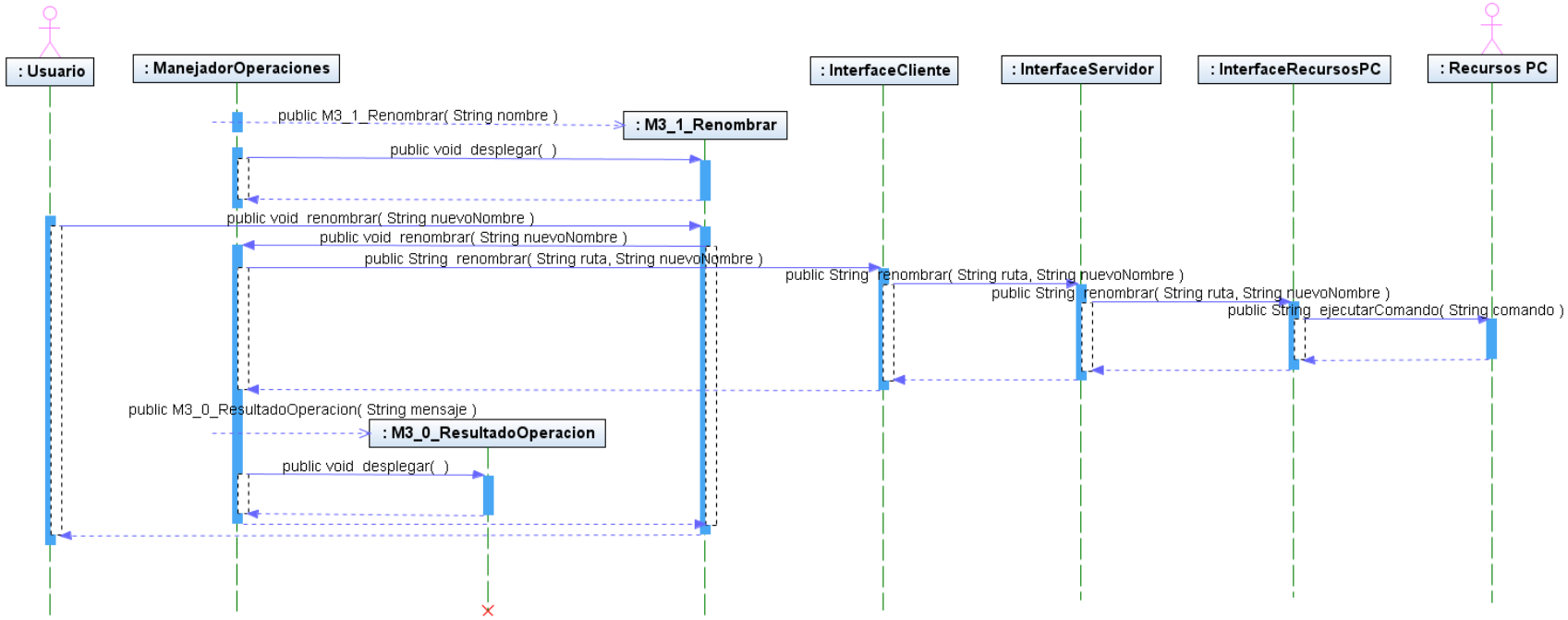


Figura 154. Diagrama de Secuencia Seleccionar Destino

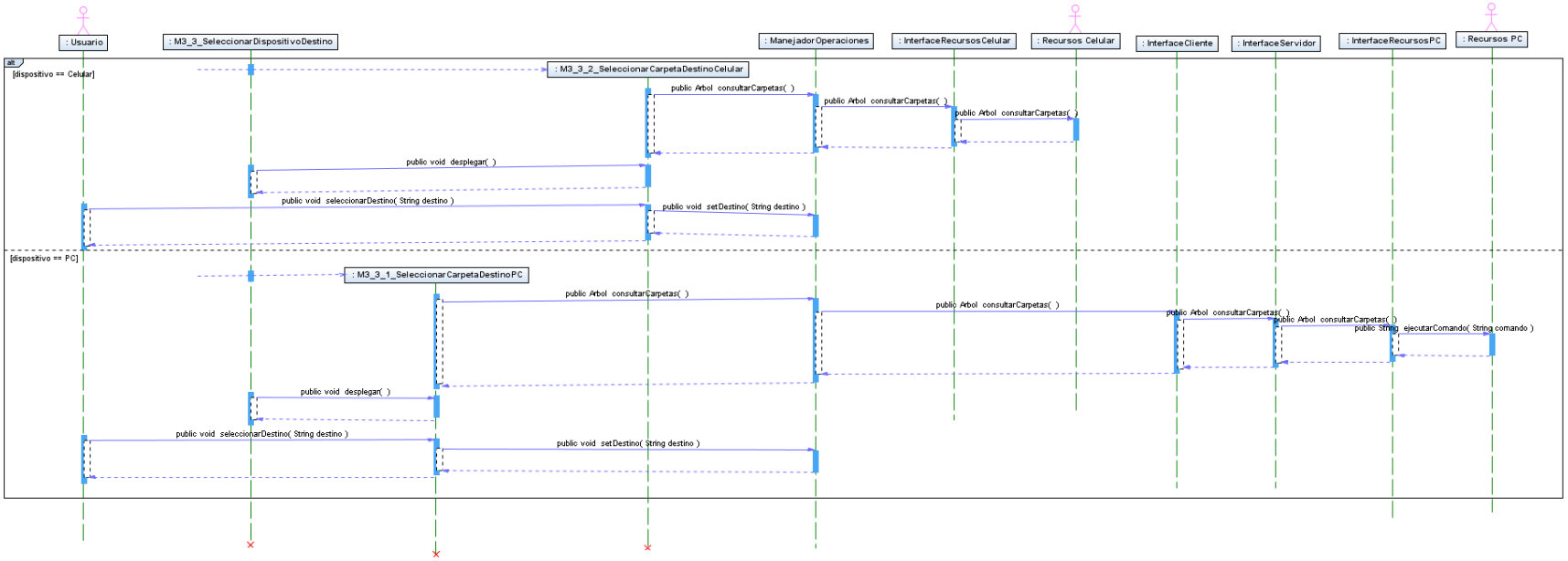


Figura 155. Diagrama de Secuencia Suspender Servicio

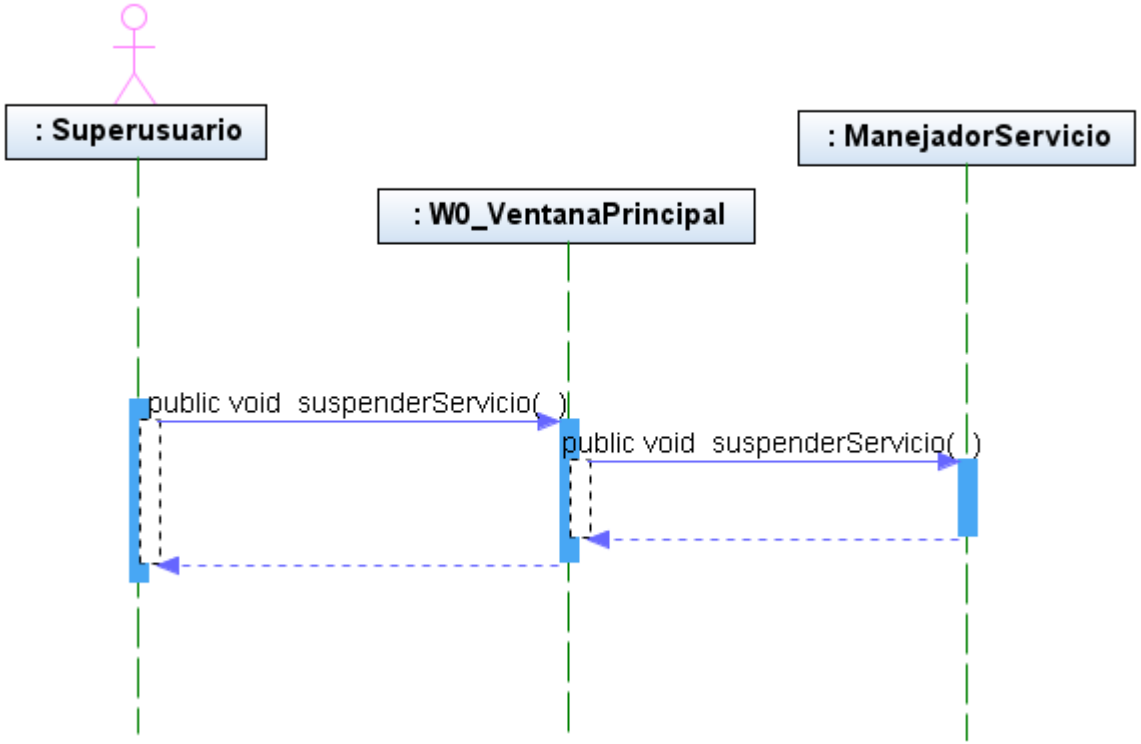


Figura 156. Diagrama de Secuencia Validar Superusuario desde el Celular

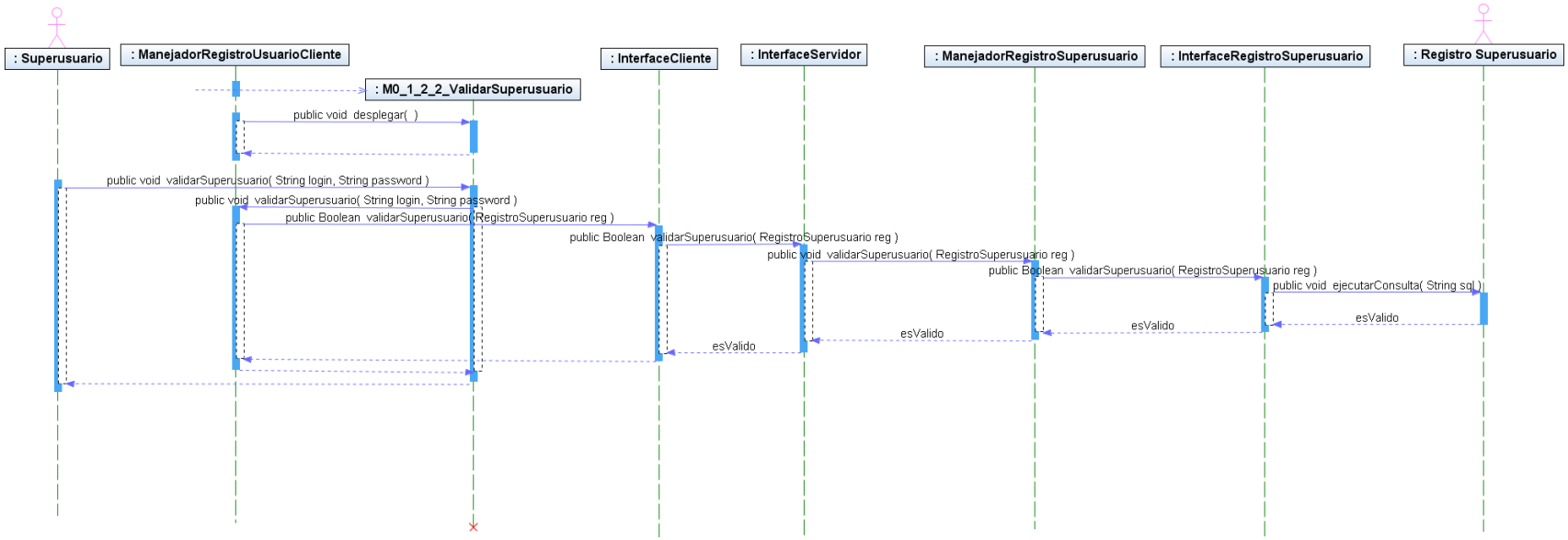


Figura 157. Diagrama de Secuencia Validar Usuario

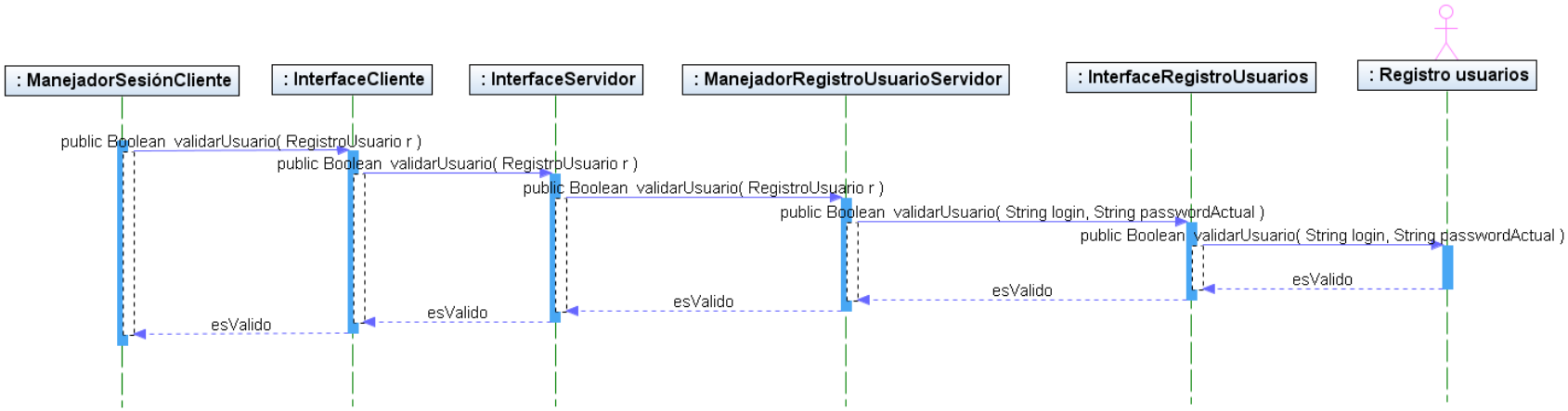


DIAGRAMA DE SUBSISTEMAS

Figura 158. Diagrama de Subsistemas

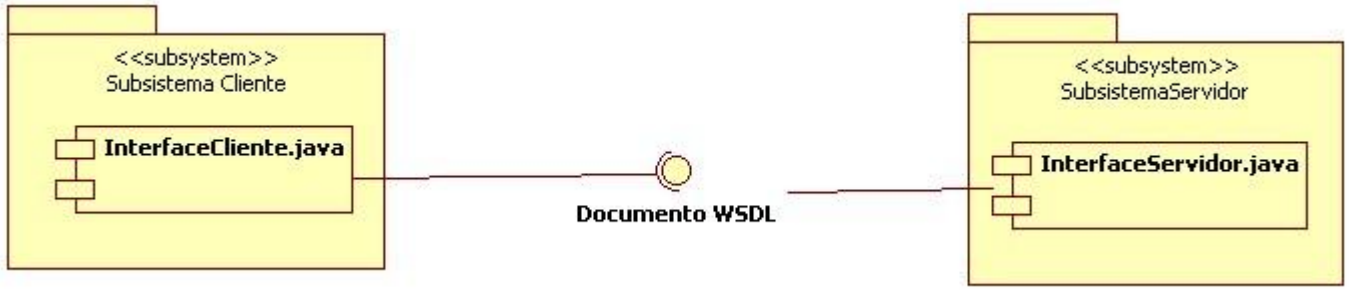


DIAGRAMA DEL SUBSISTEMA CLIENTE

Figura 159. Diagrama del Subsistema Cliente

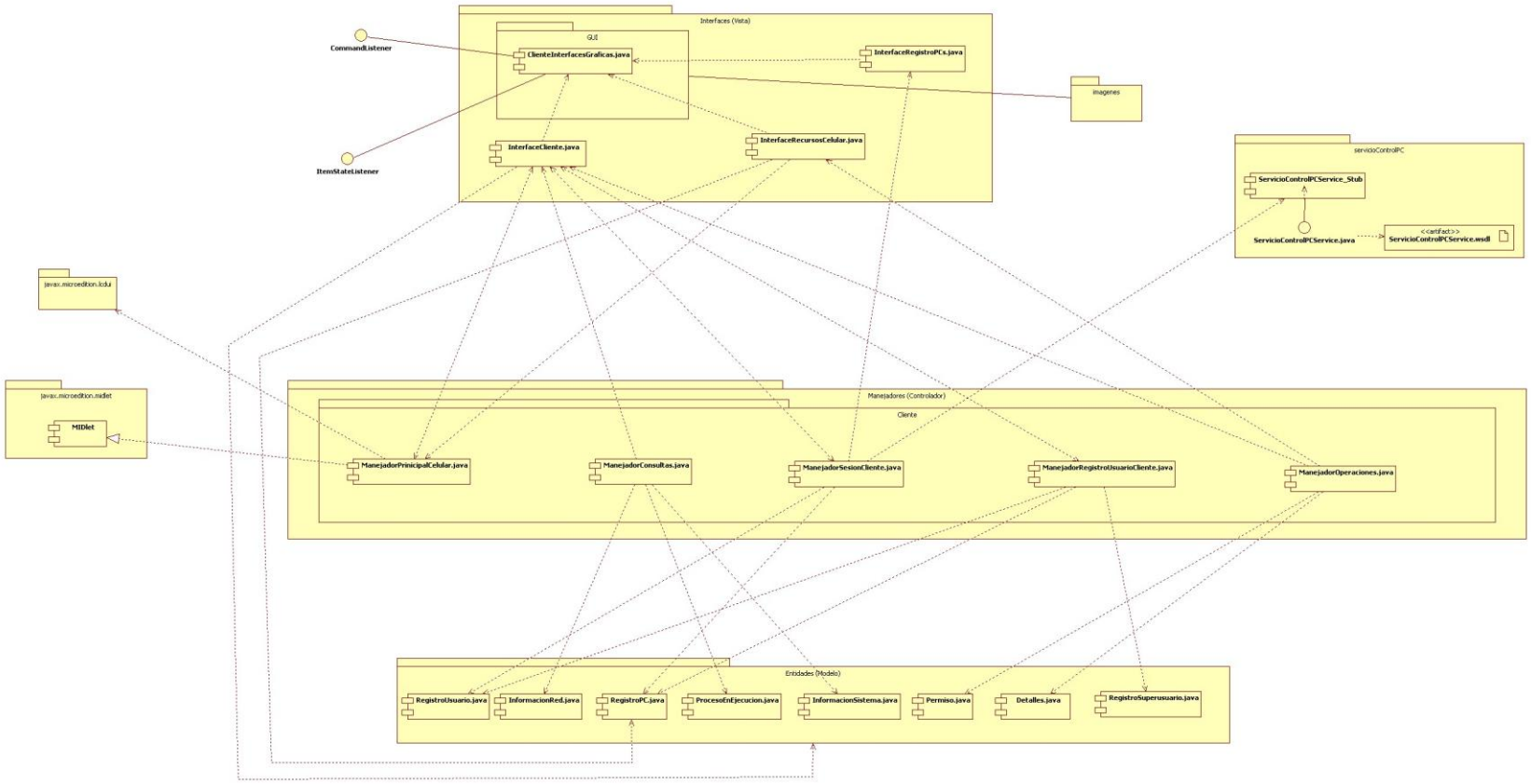


DIAGRAMA DEL SUBSISTEMA SERVIDOR

Figura 160. Diagrama del Subsistema Servidor

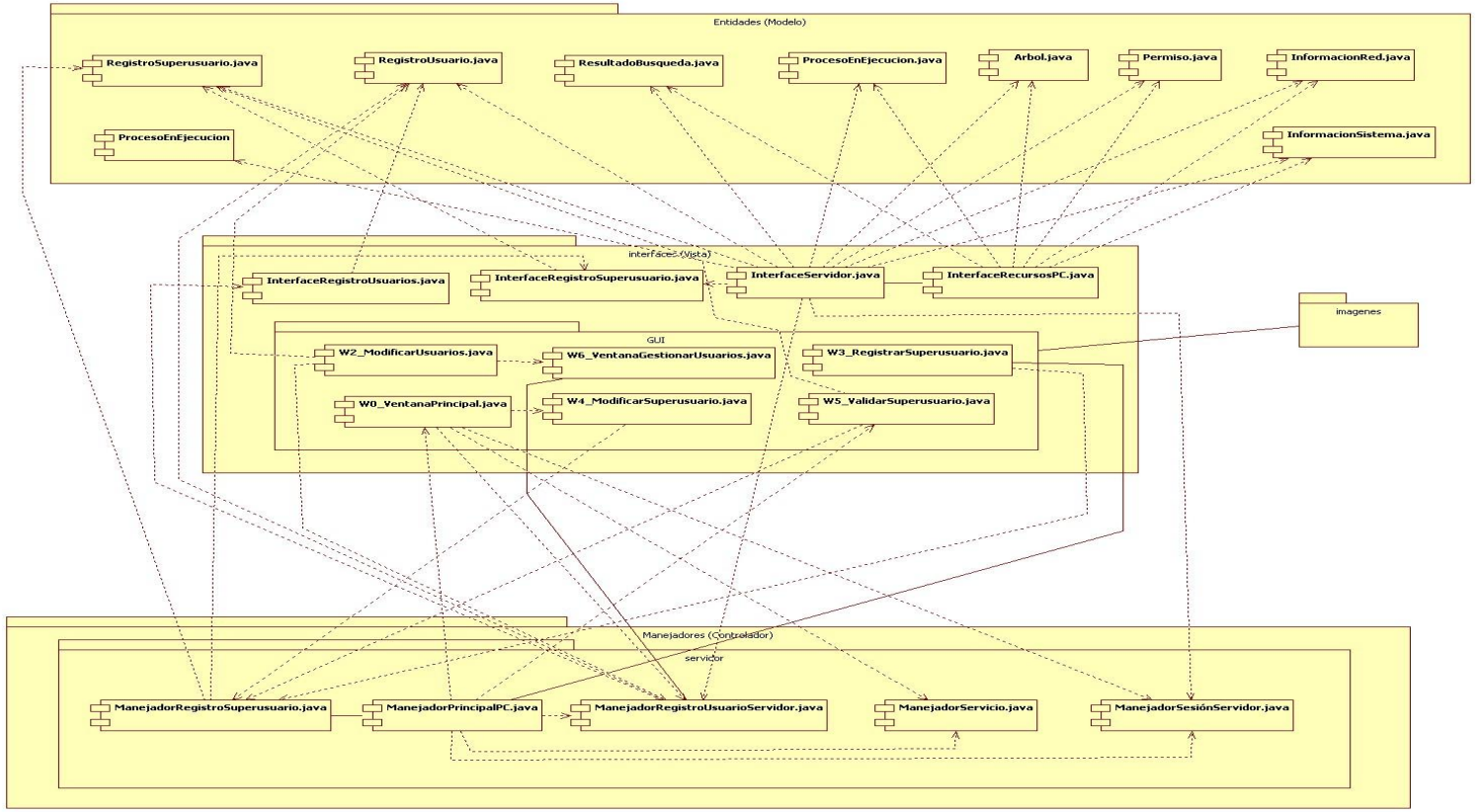


DIAGRAMA DE COLABORACIÓN DEL SUBSISTEMA SERVIDOR

Figura 161. Diagrama de Colaboración del Subsistema Servidor

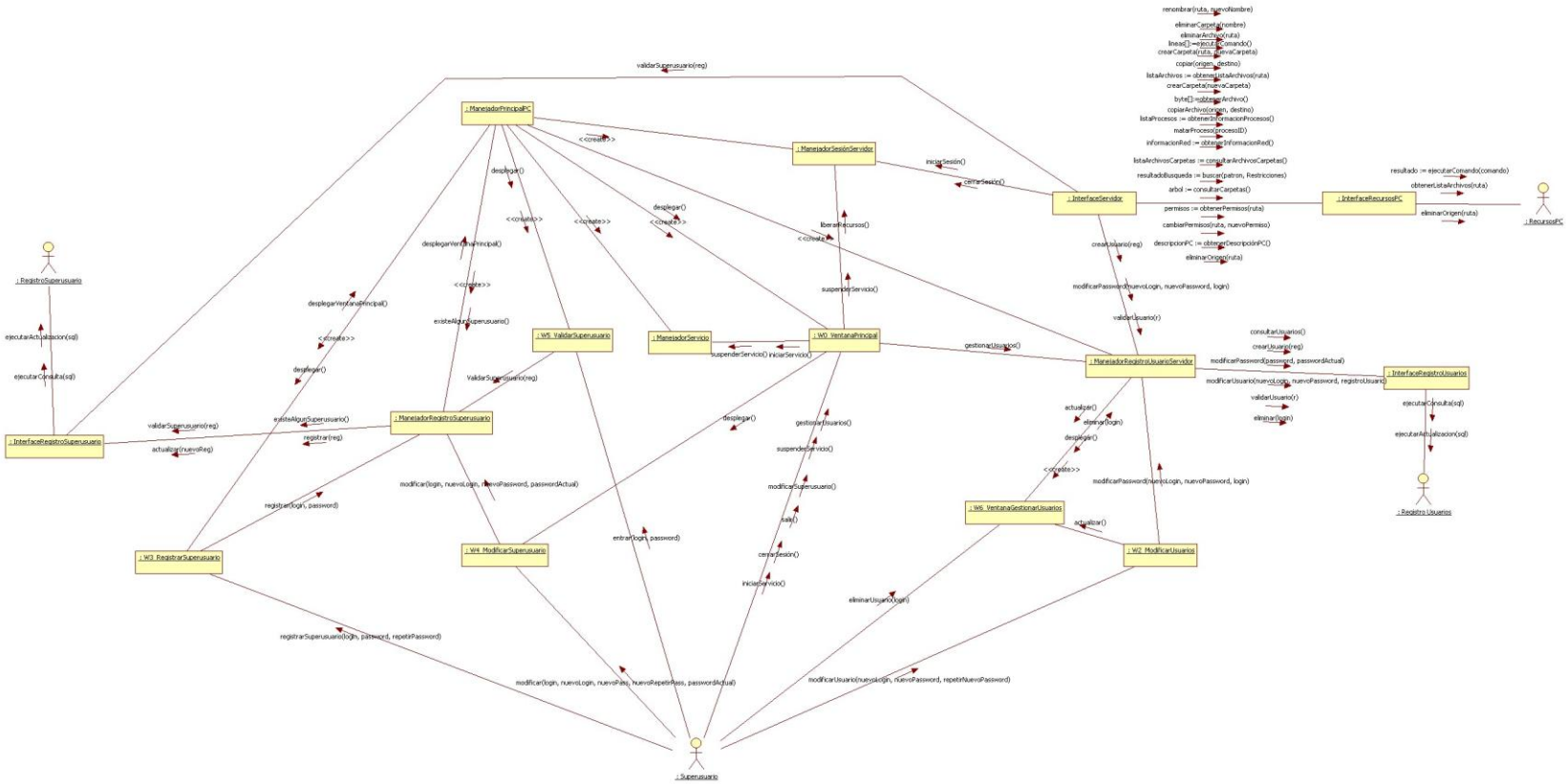
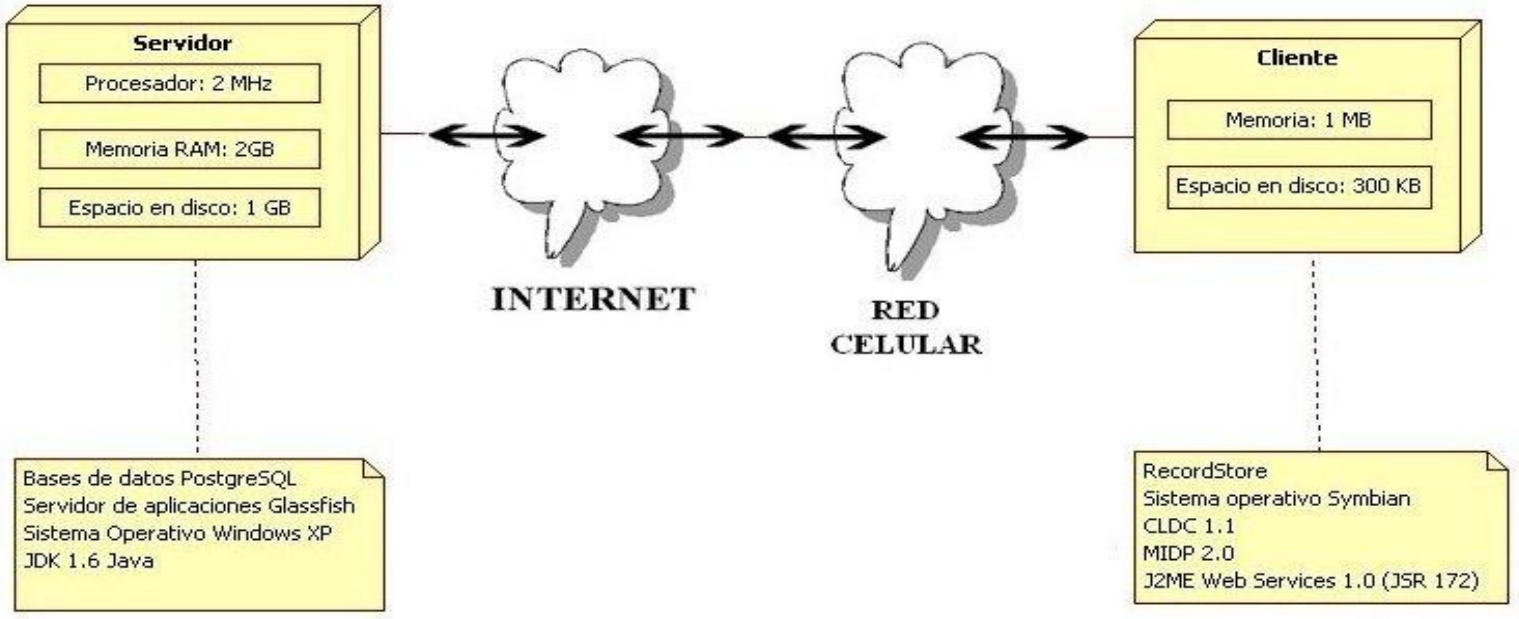


DIAGRAMA DE DESPLIEGUE

Figura 162. Diagrama de Despliegue



7. DIAGRAMAS DE CLASES

A continuación se muestran los diagramas de clases, los cuales son producto del estudio hecho a la secuencia y a sus objetos generados.

Figura 163. Diagrama de clases Parte1

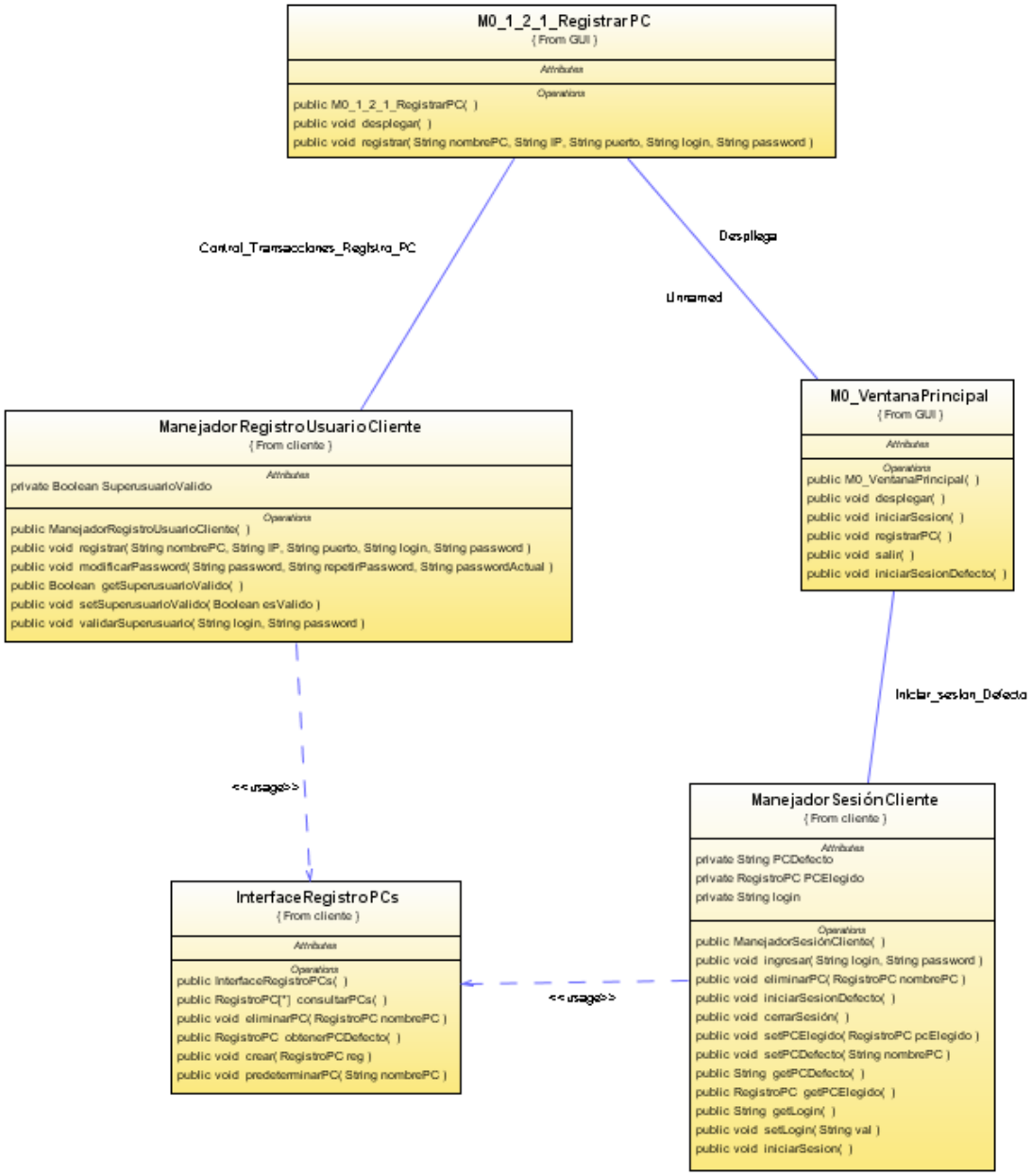


Figura 164. Diagrama de clases Parte 2

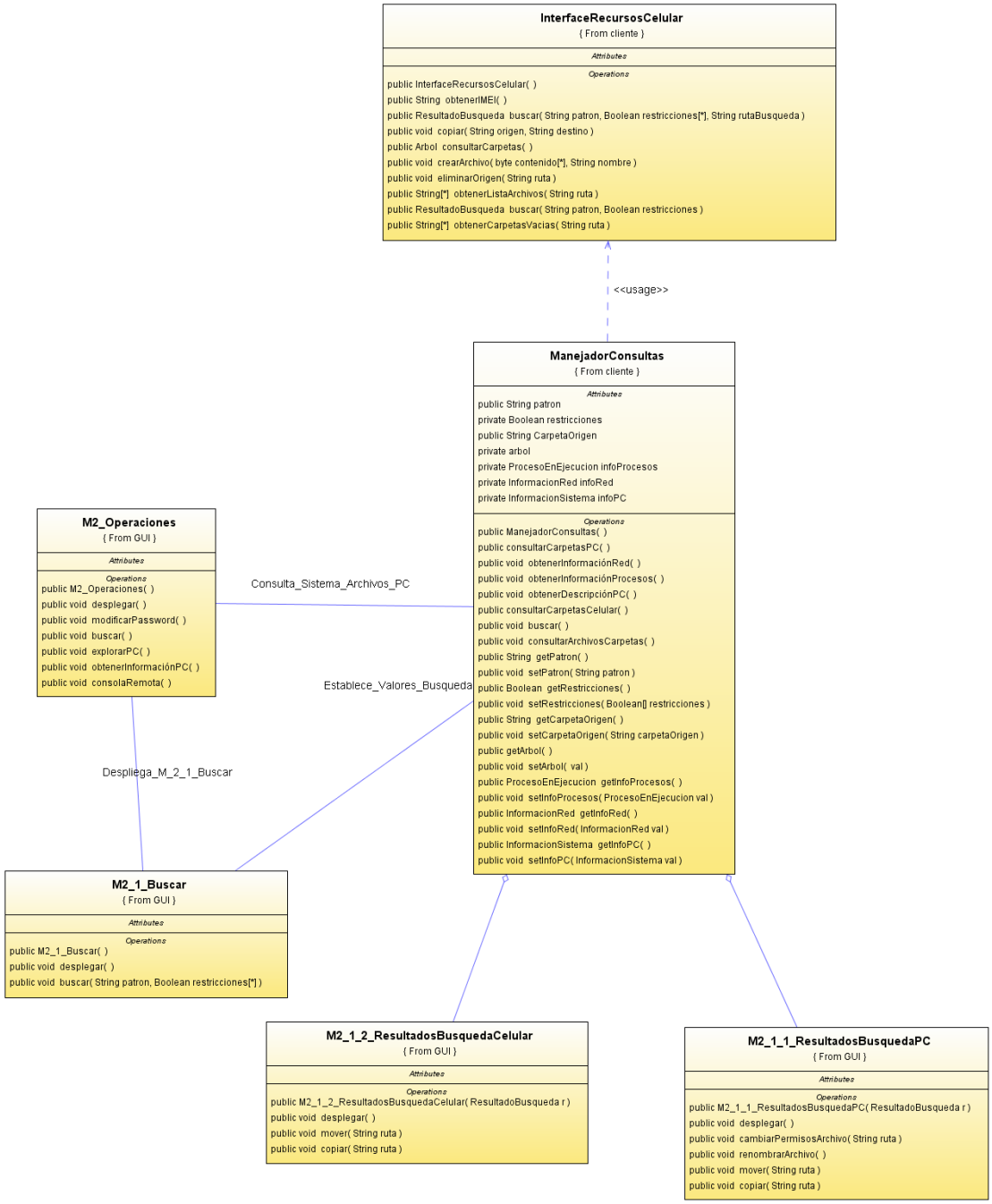


Figura 165. Diagrama de clases Parte 3

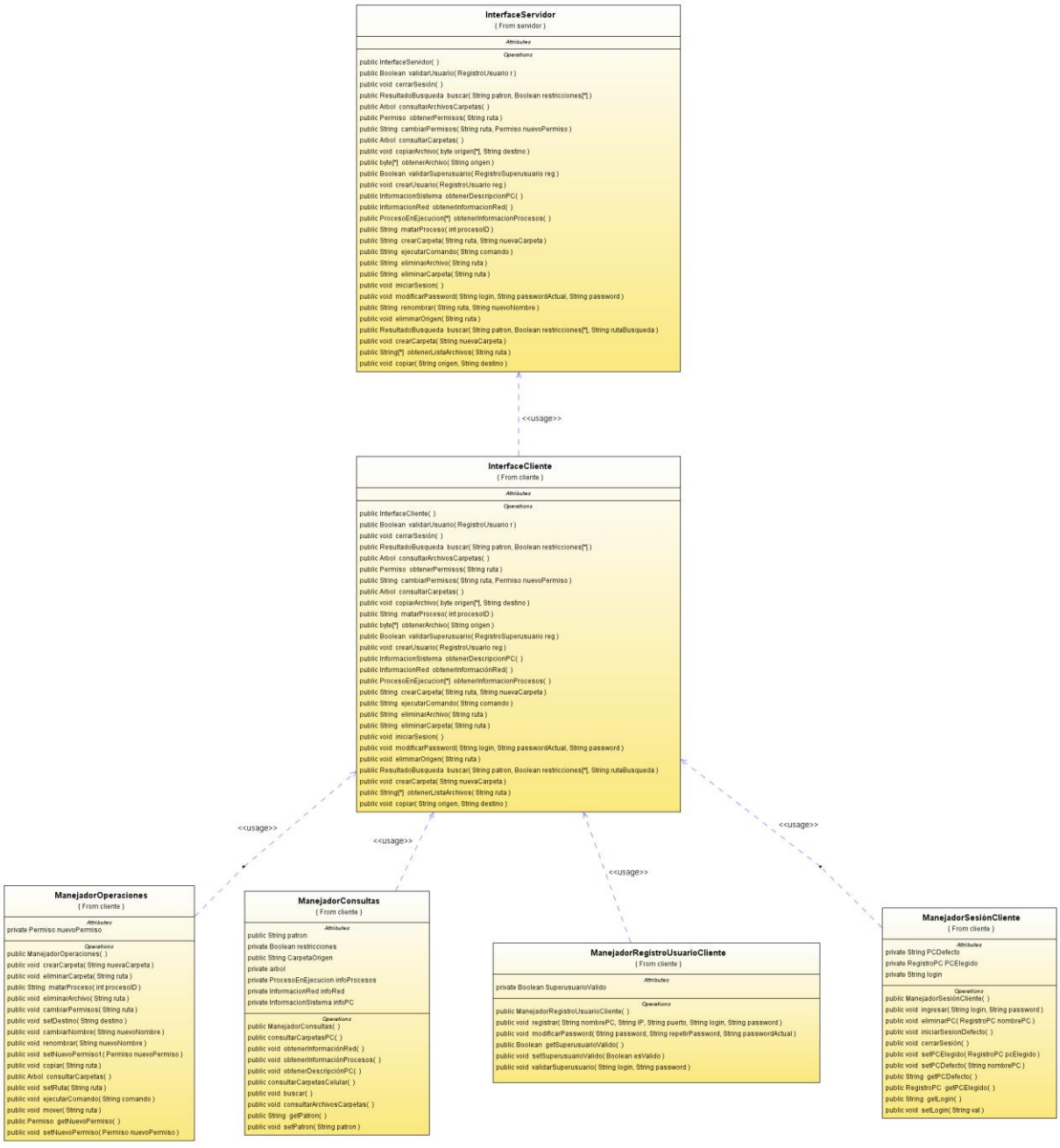


Figura 166. Diagrama de clases Parte 4

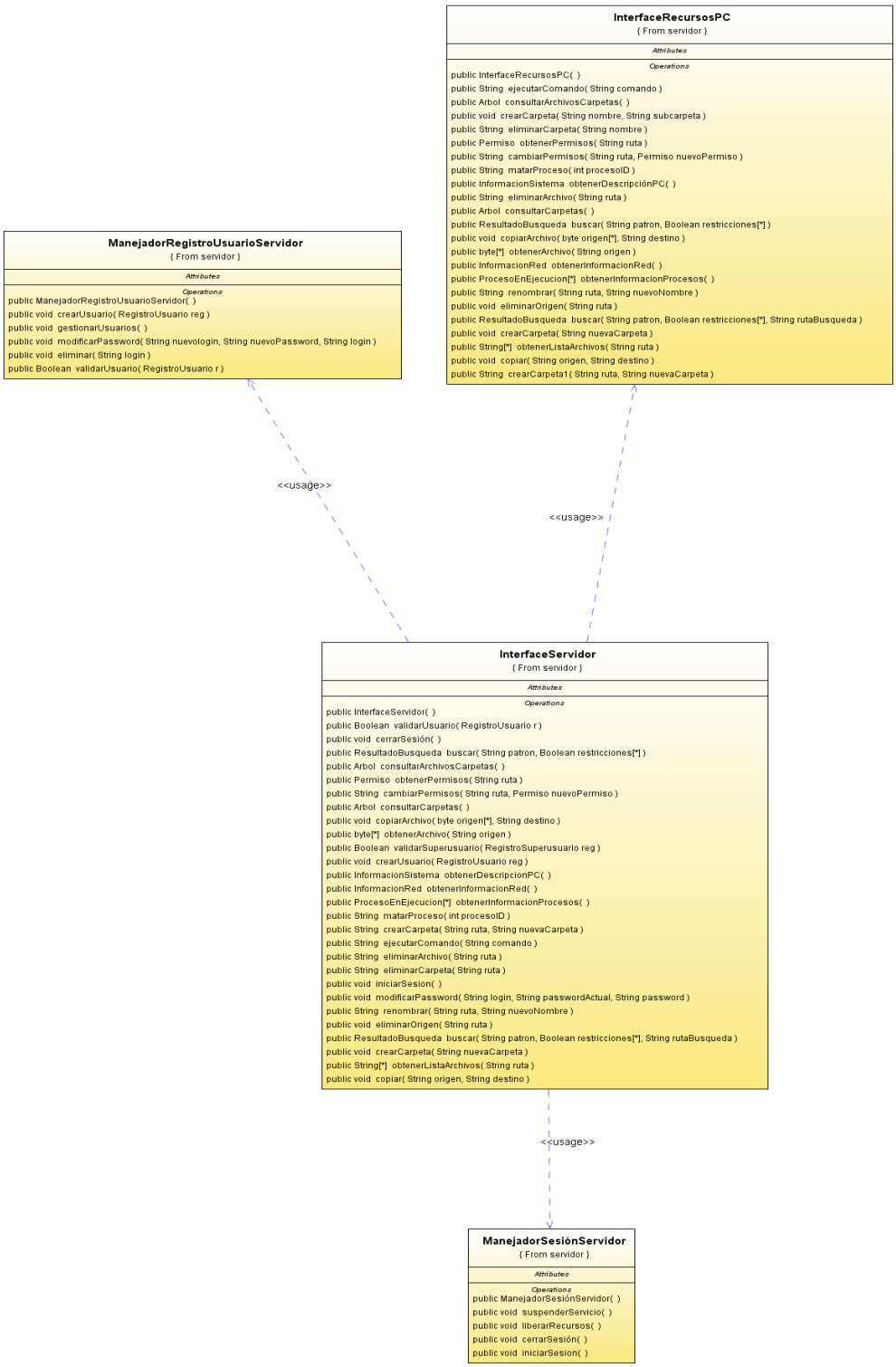


Figura 167. Diagrama de clases Parte 5

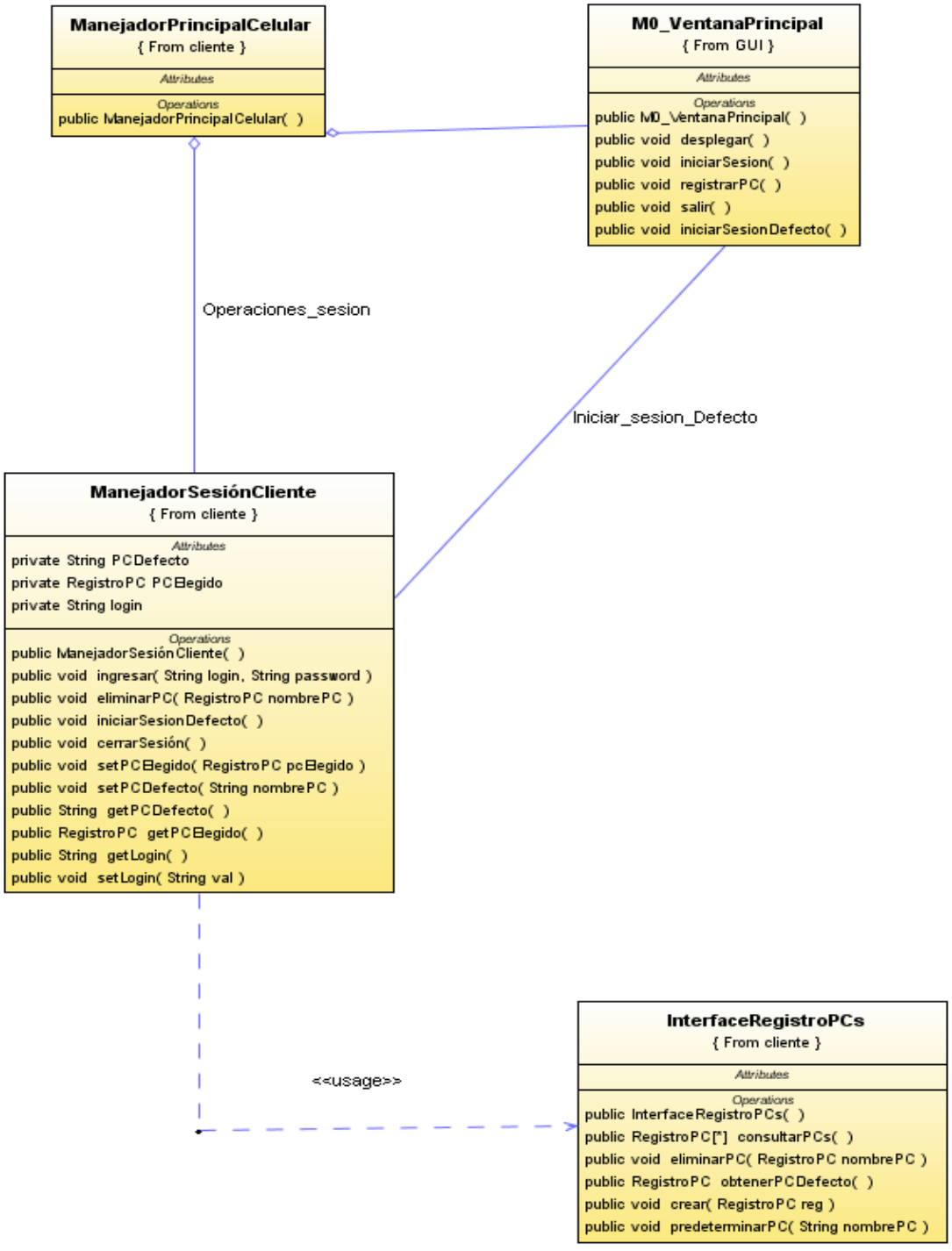


Figura 168. Diagrama de clases Parte 6

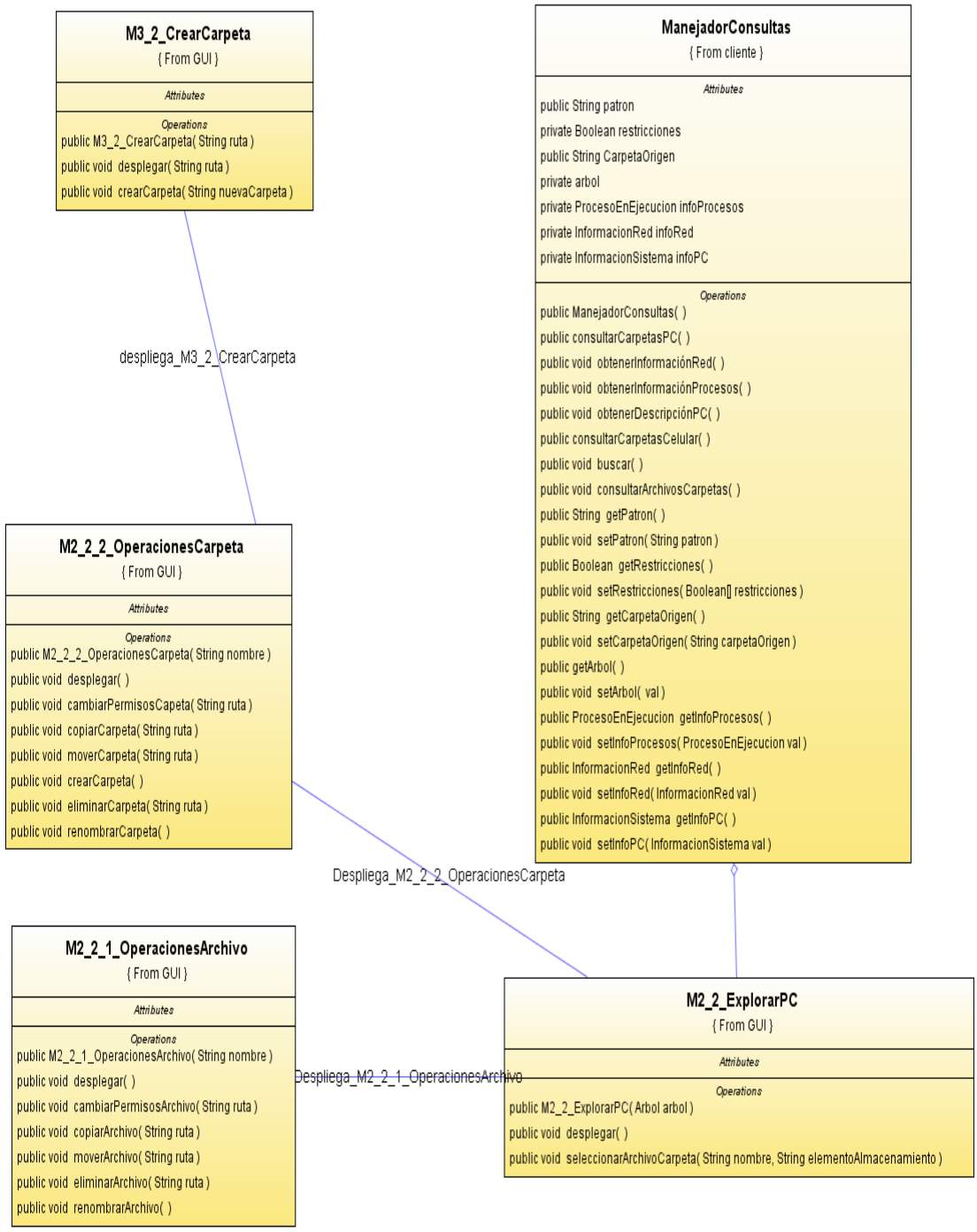


Figura 169. Diagrama de clases Parte 7

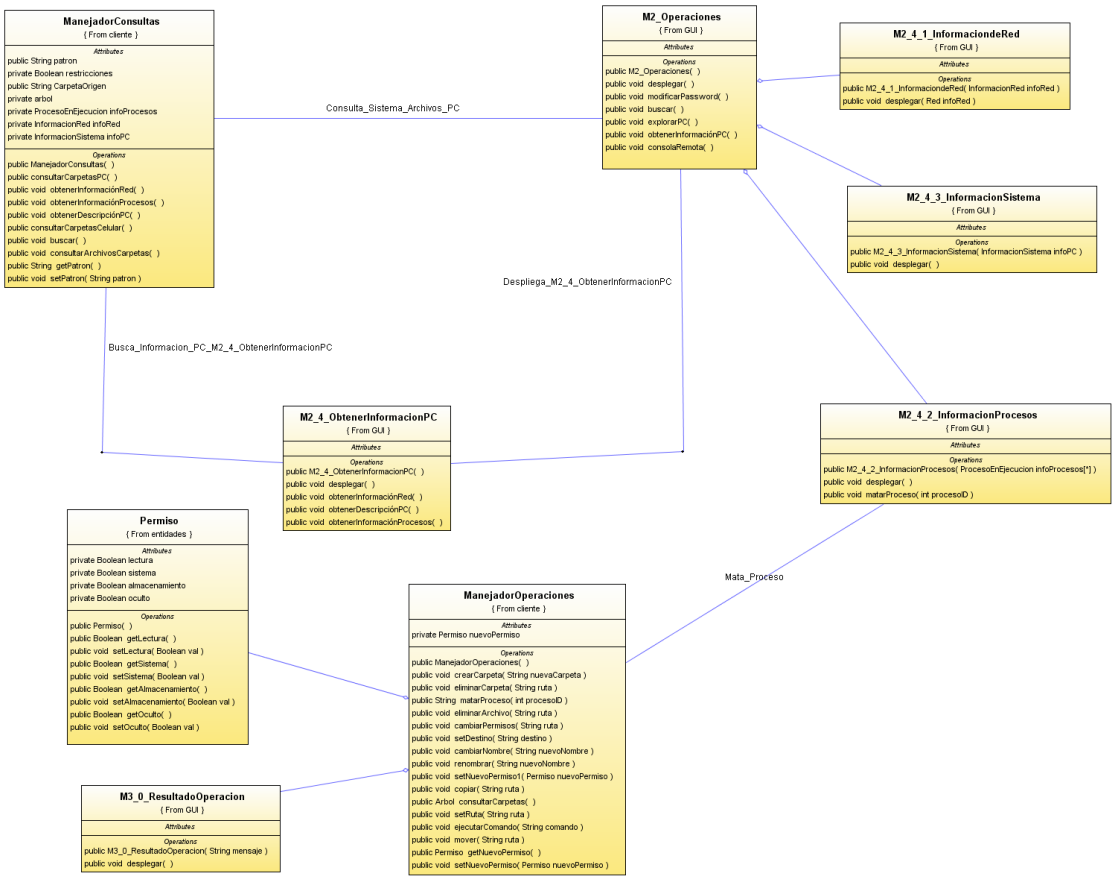


Figura 170. Diagrama de clases Parte 8

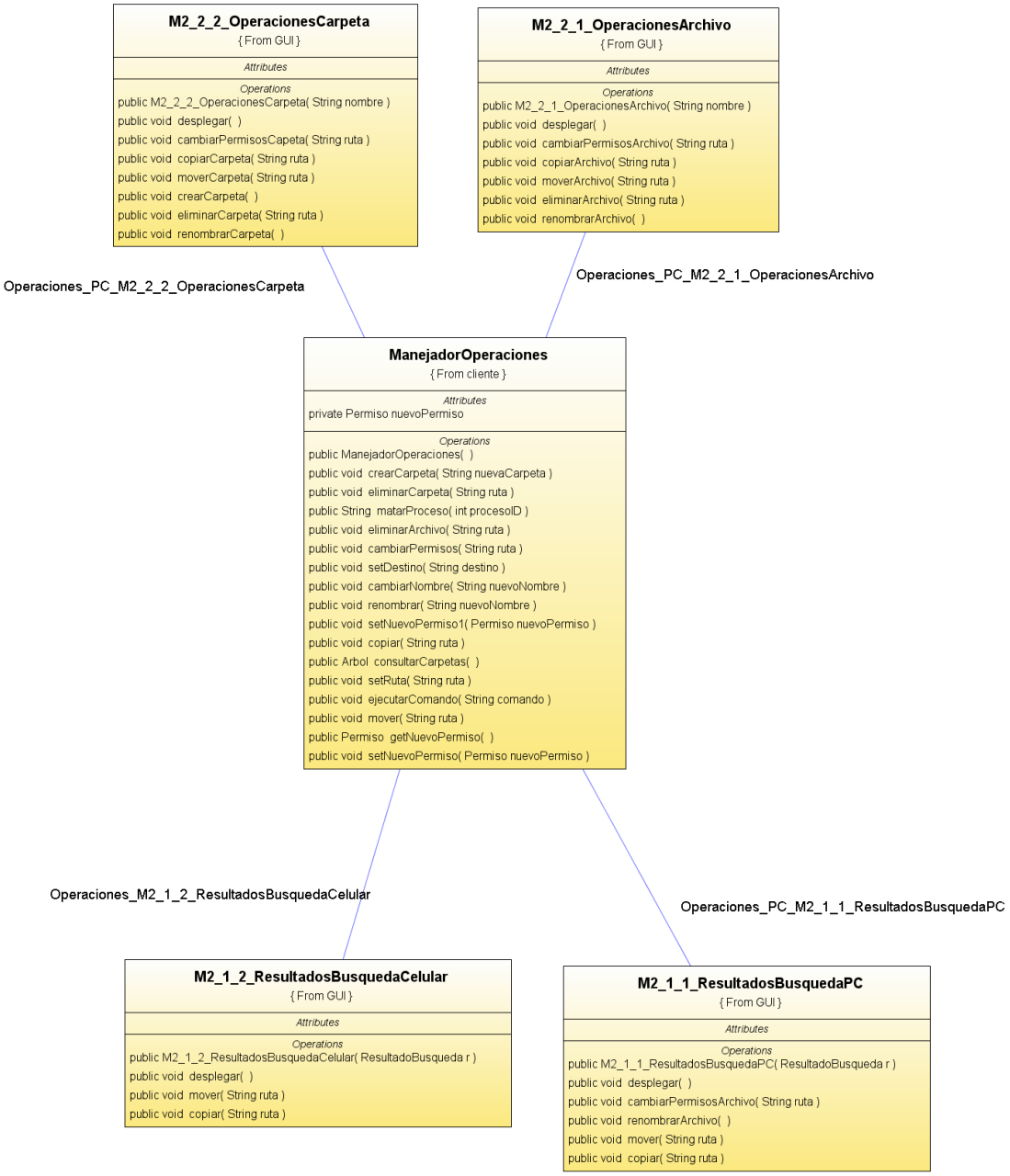


Figura 171. Diagrama de clases Parte 9

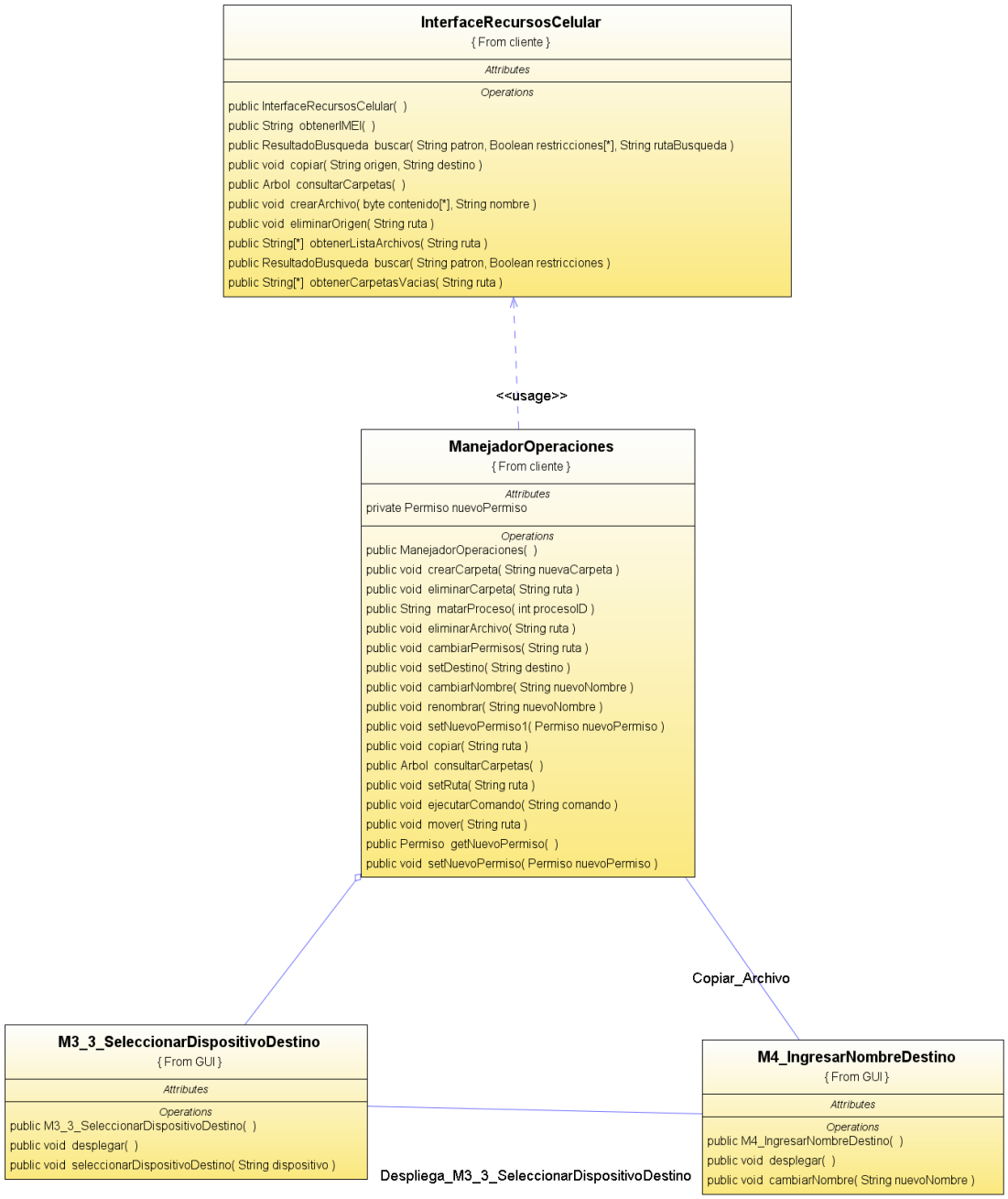


Figura 172. Diagrama de clases Parte 10

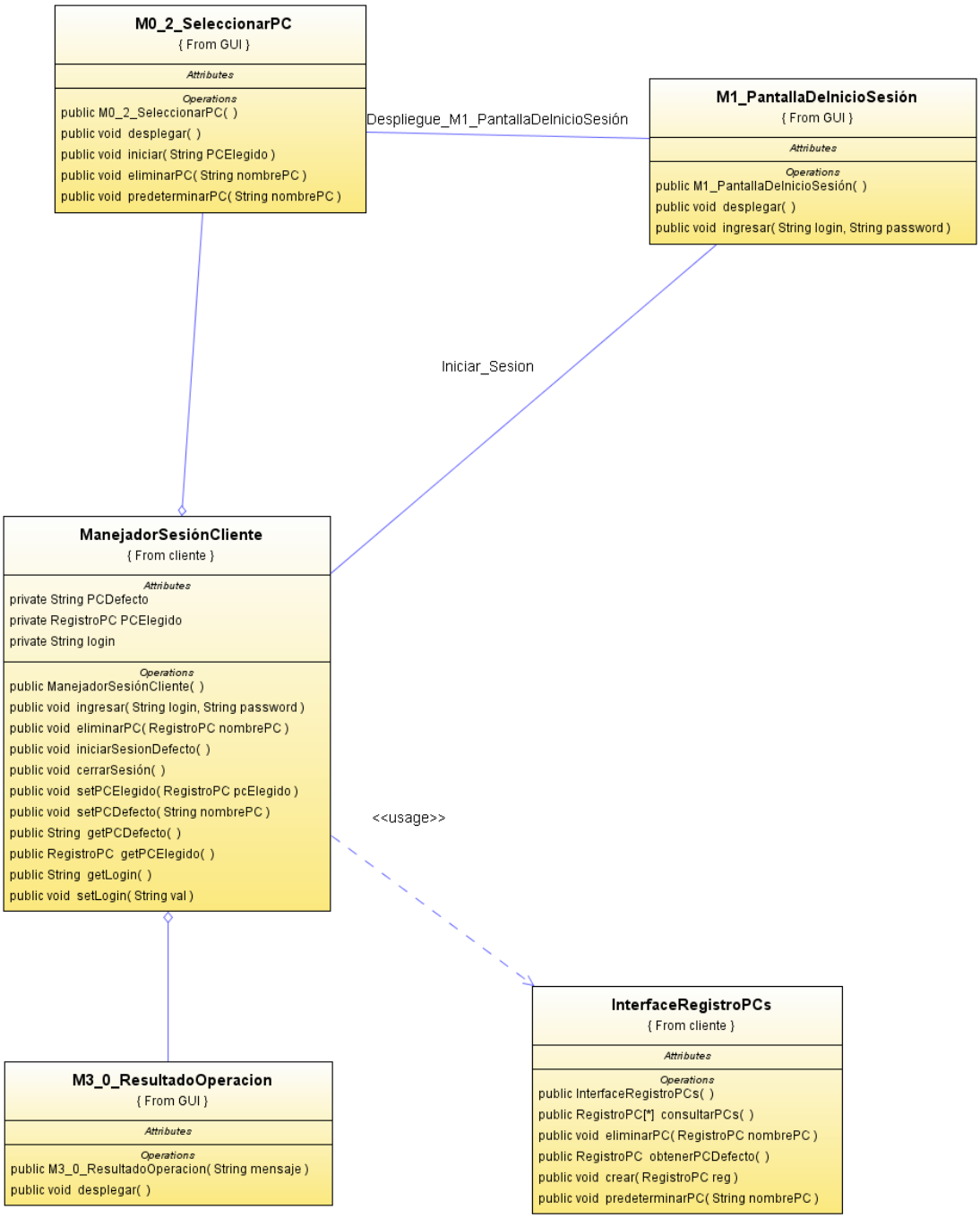


Figura 173. Diagrama de clases Parte 11

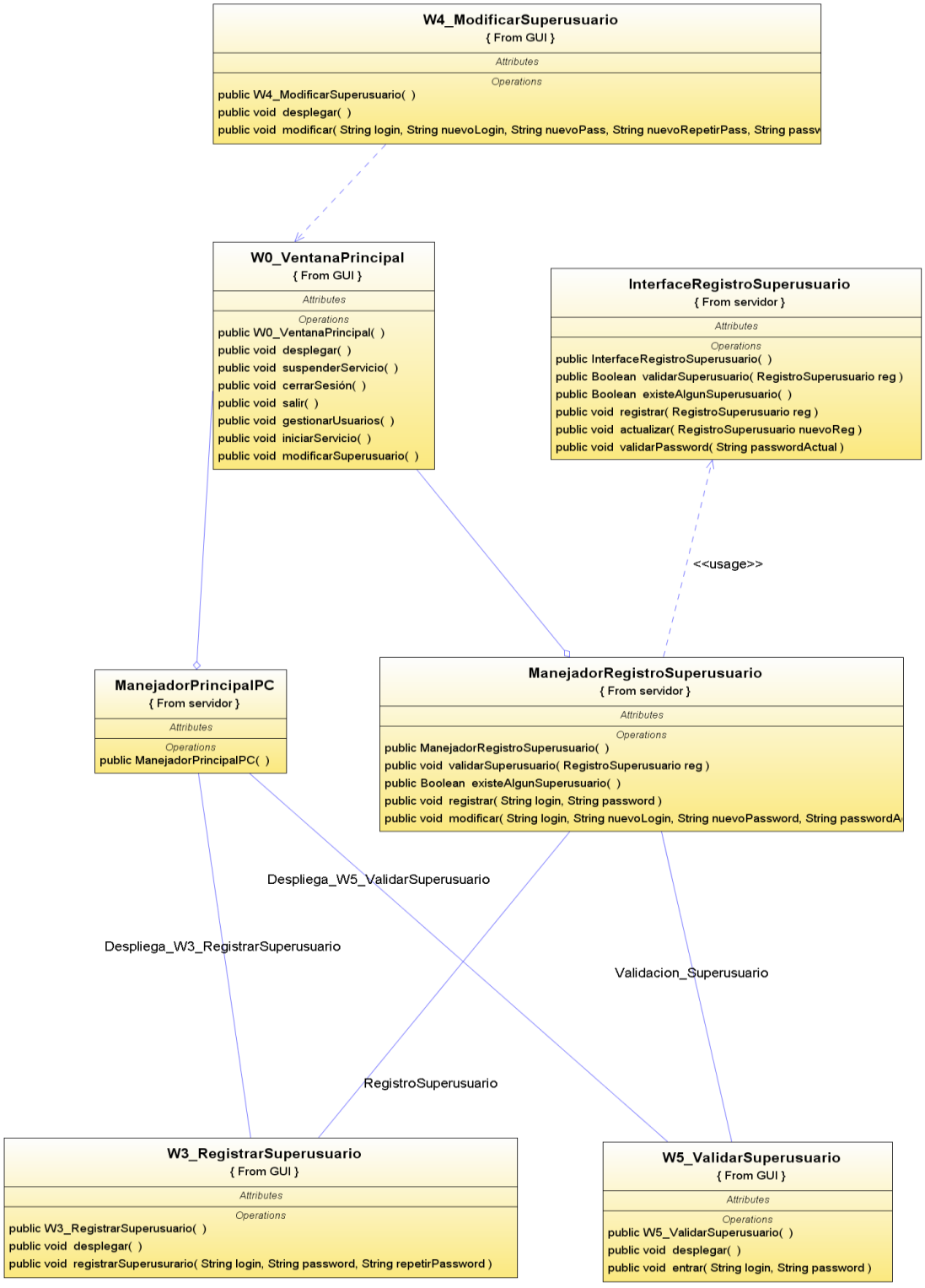


Figura 174. Diagrama de clases Parte 12

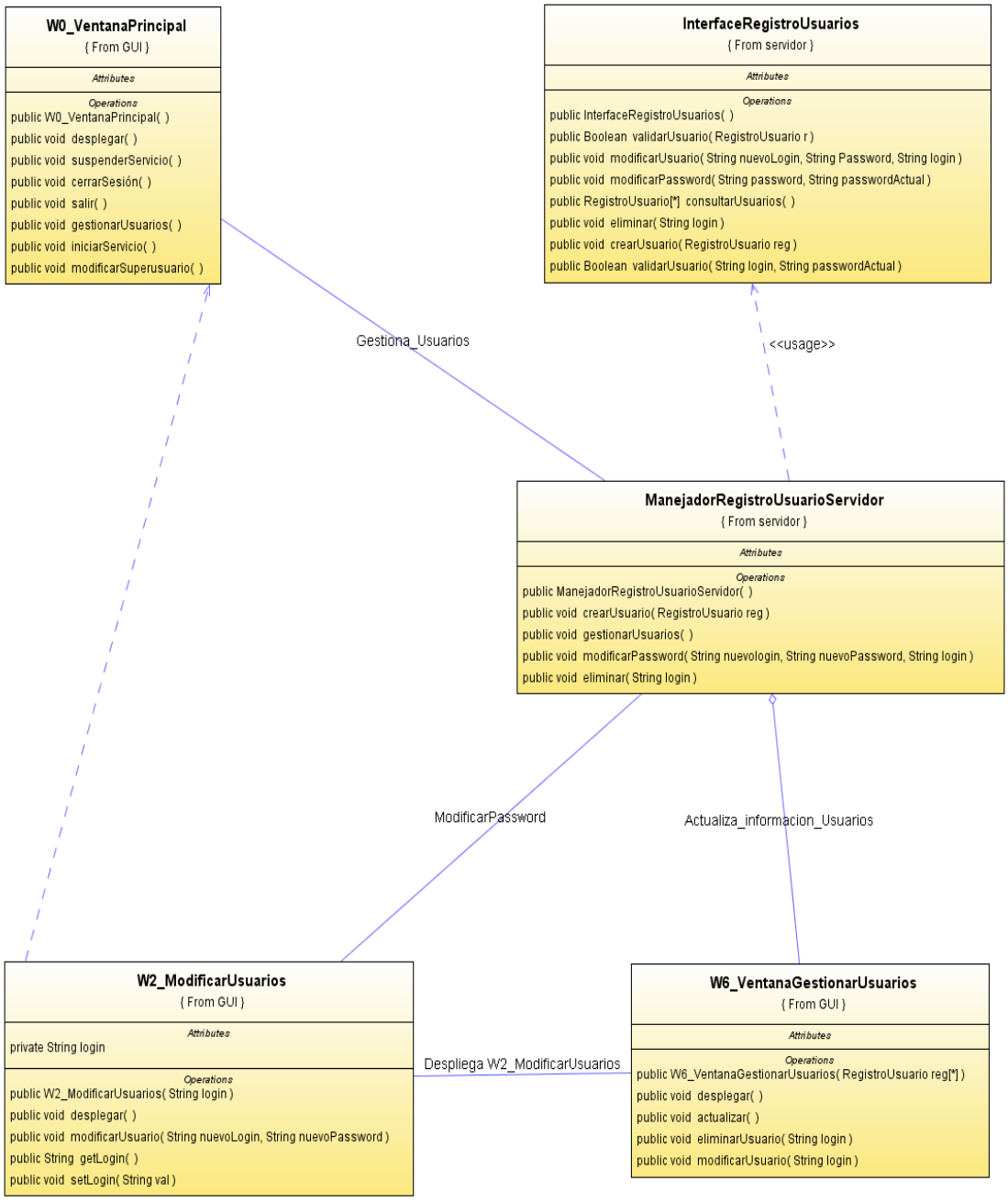


Figura 175. Diagrama de clases Parte 13

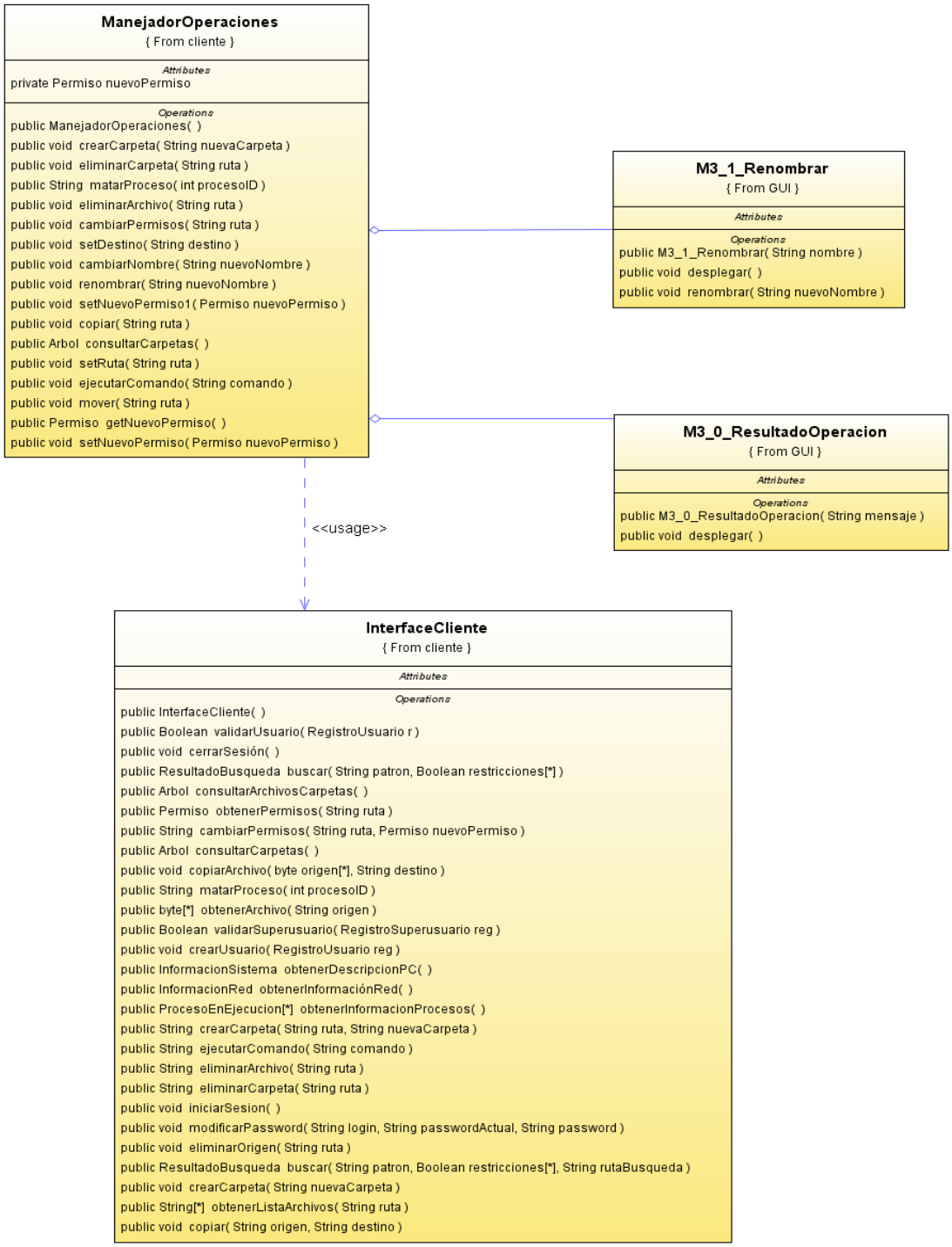


Figura 176. Diagrama de clases Parte 14

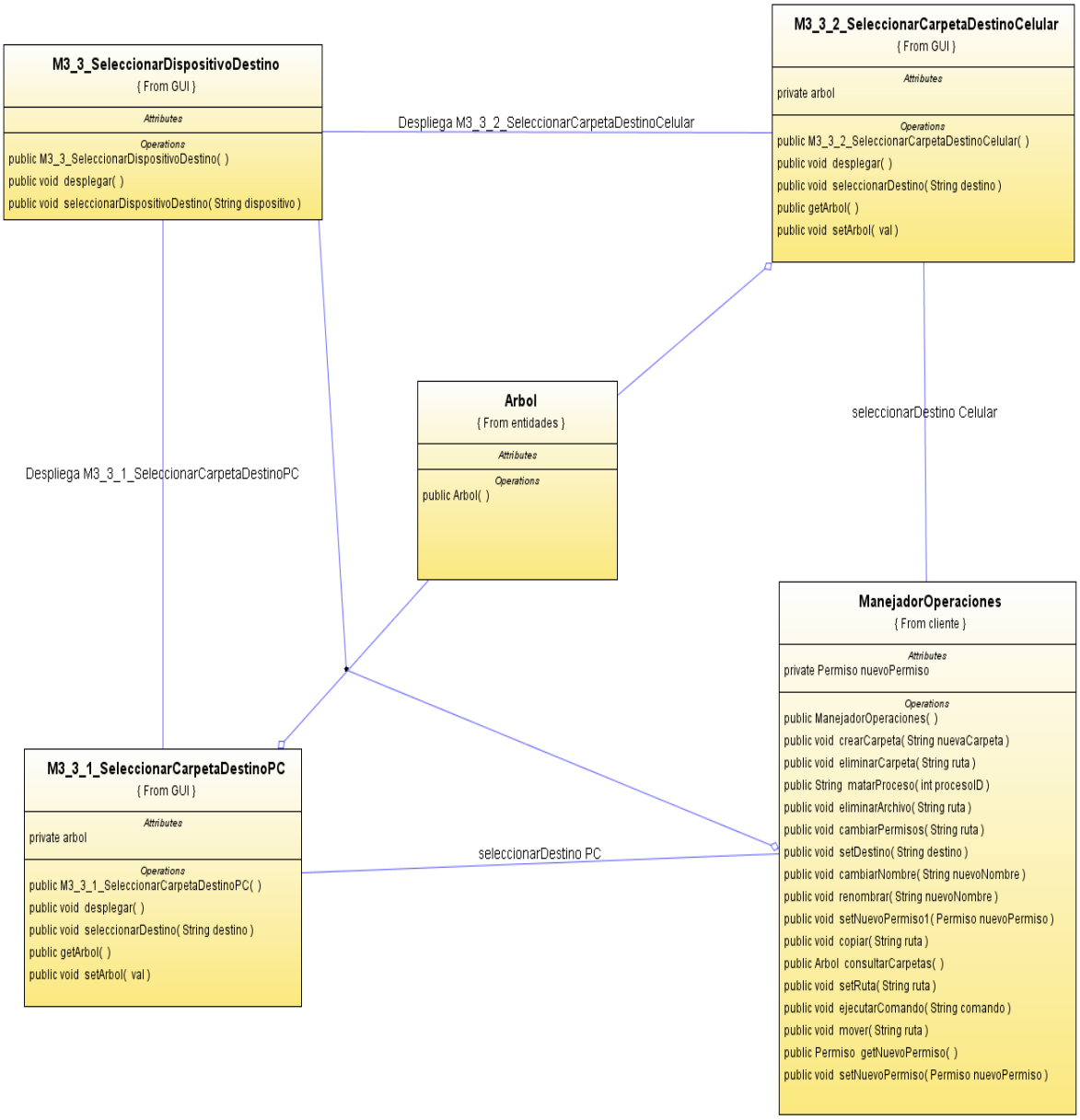


Figura 177. Diagrama de Clases Parte 15

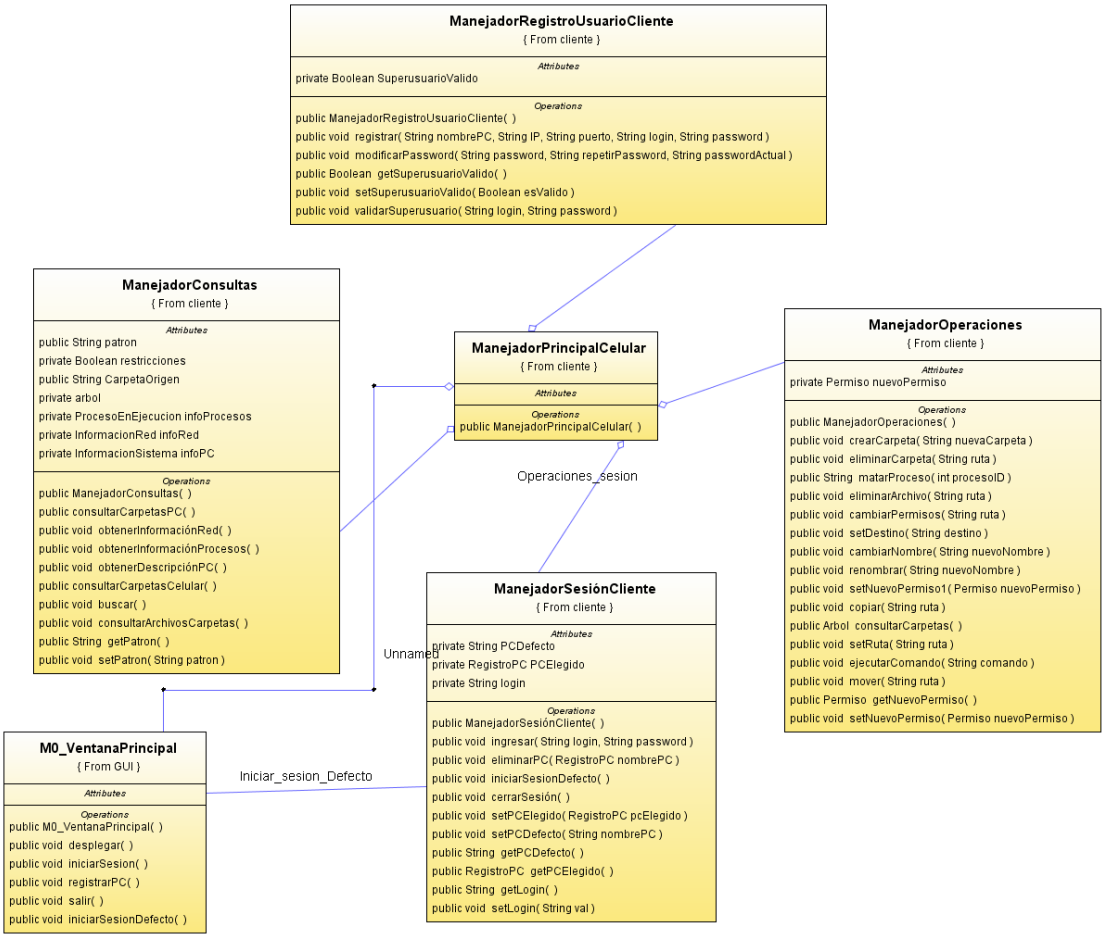


Figura 178. Diagrama de Clases Parte 16

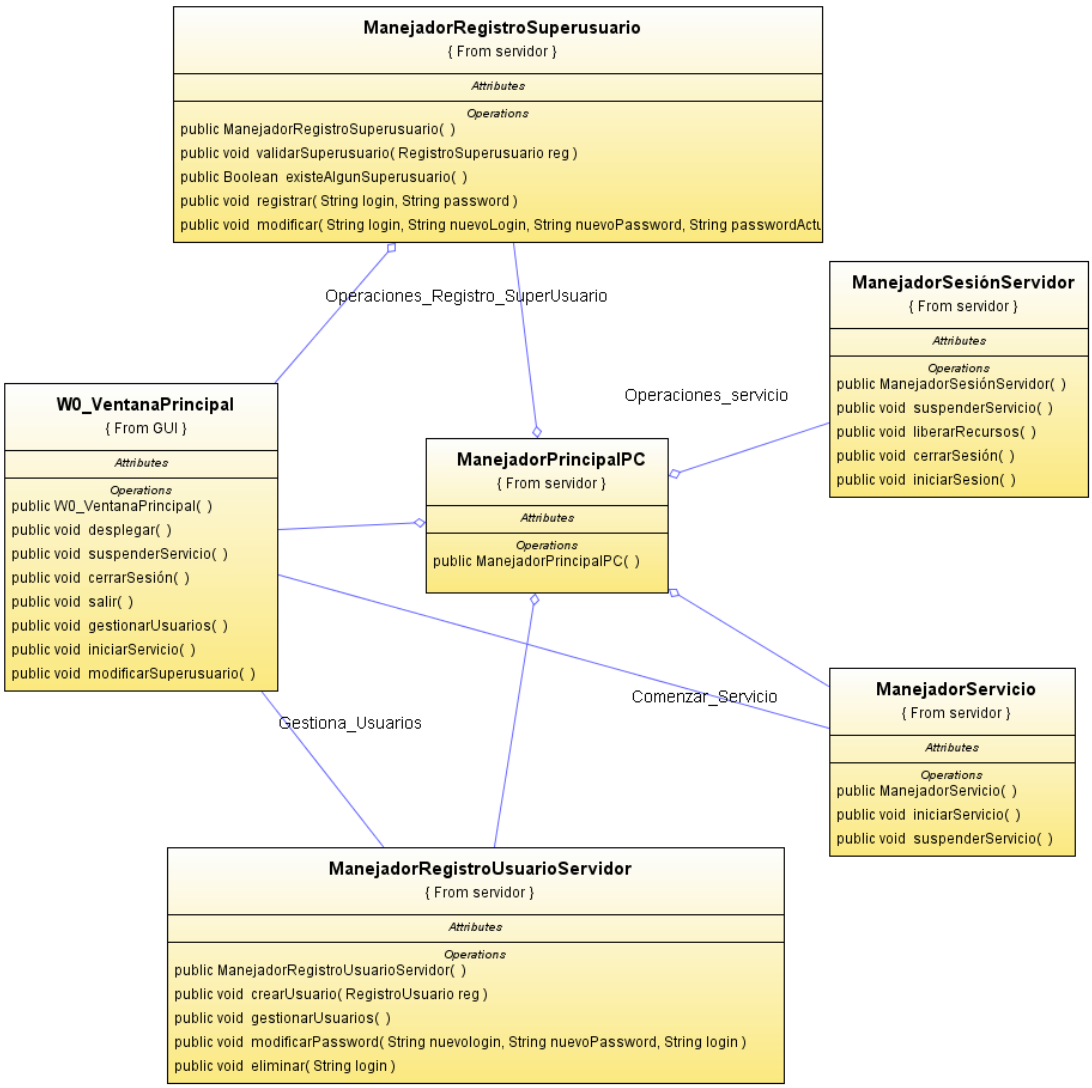


Figura 179. Diagrama de Clases Parte 17

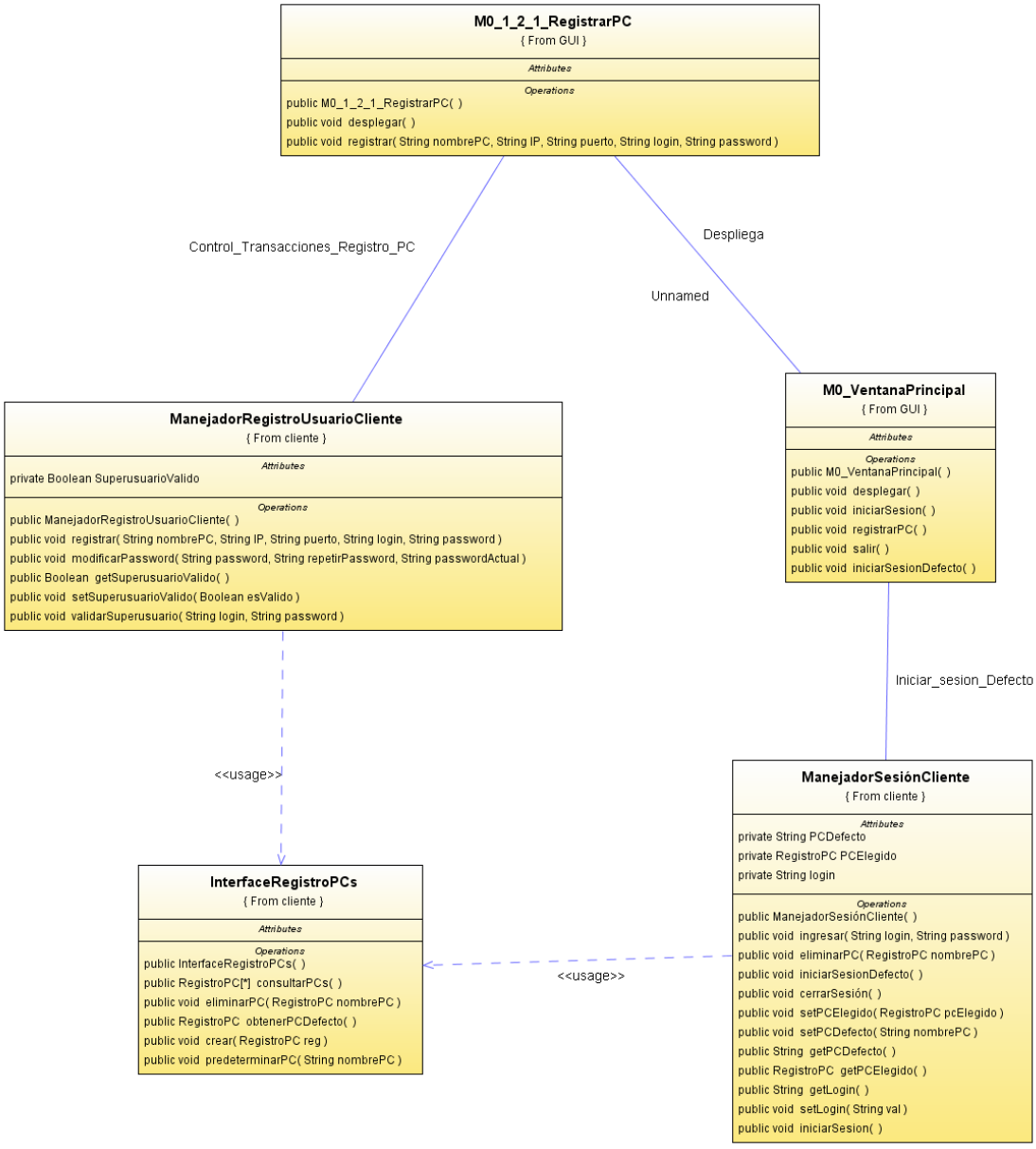


Figura 180. Diagrama de Clases Parte 18

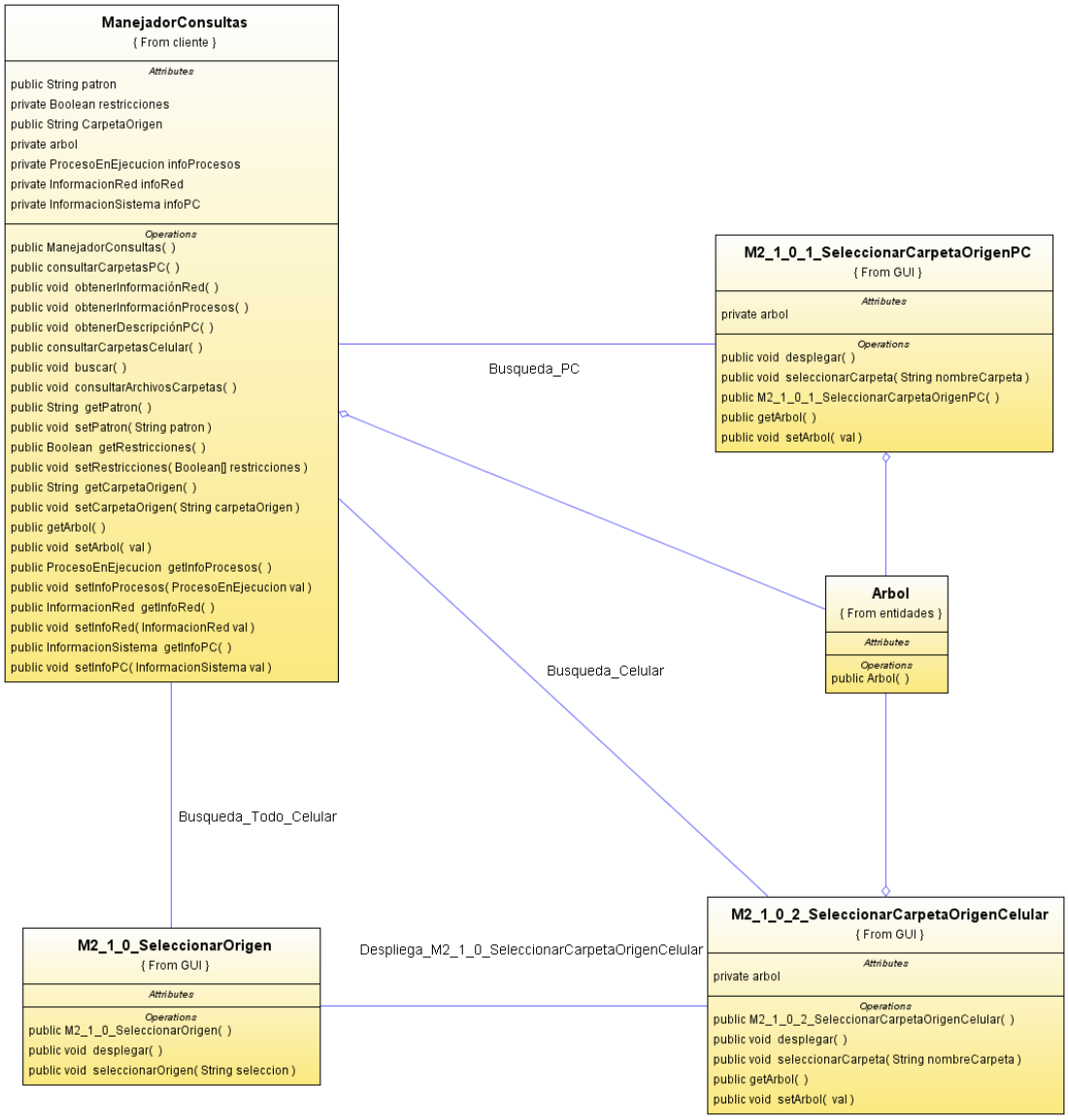


Figura 181. Diagrama de Clases Parte 21

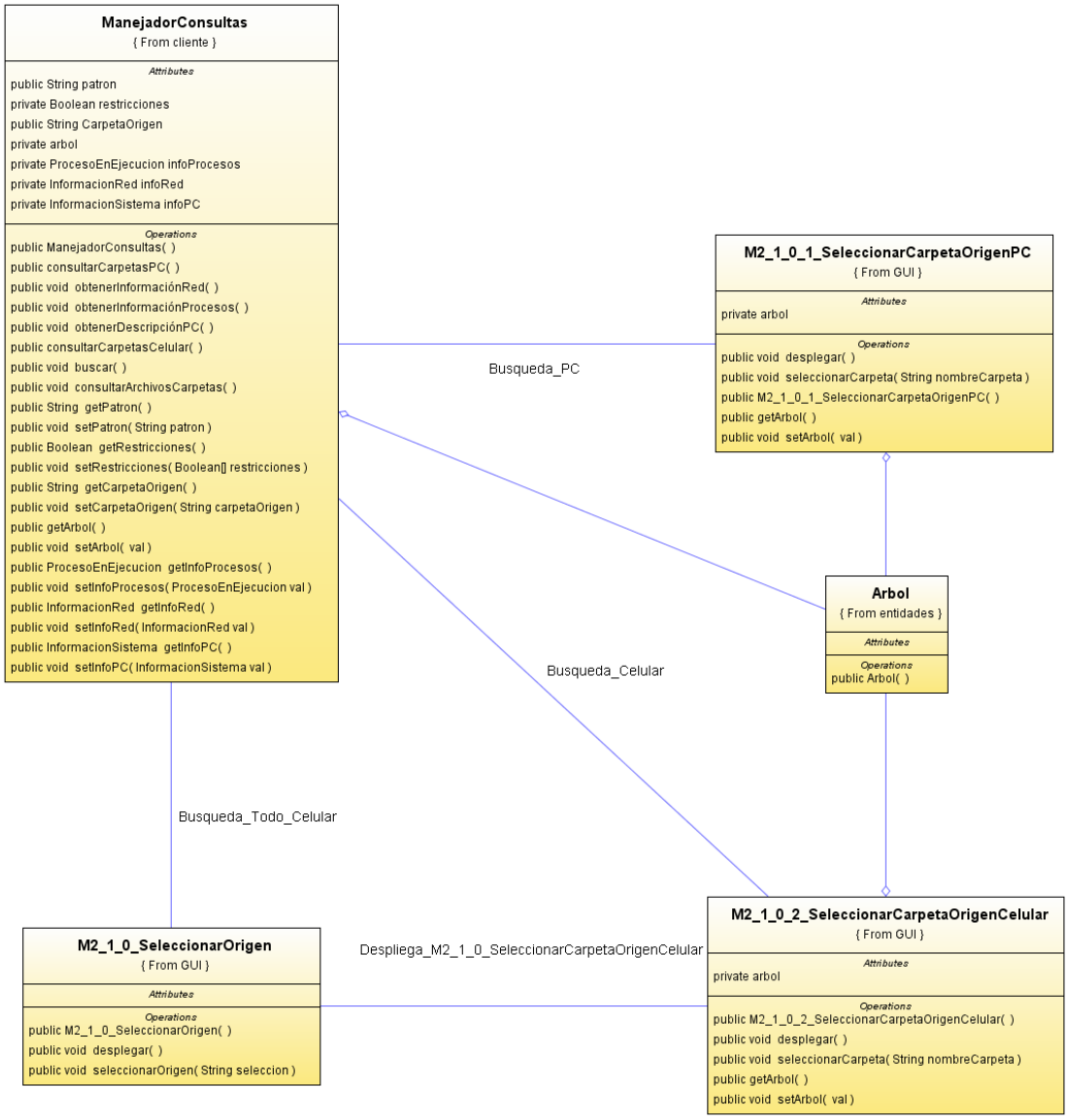


Figura 182. Diagrama de Clases Herencia ExploradorPC

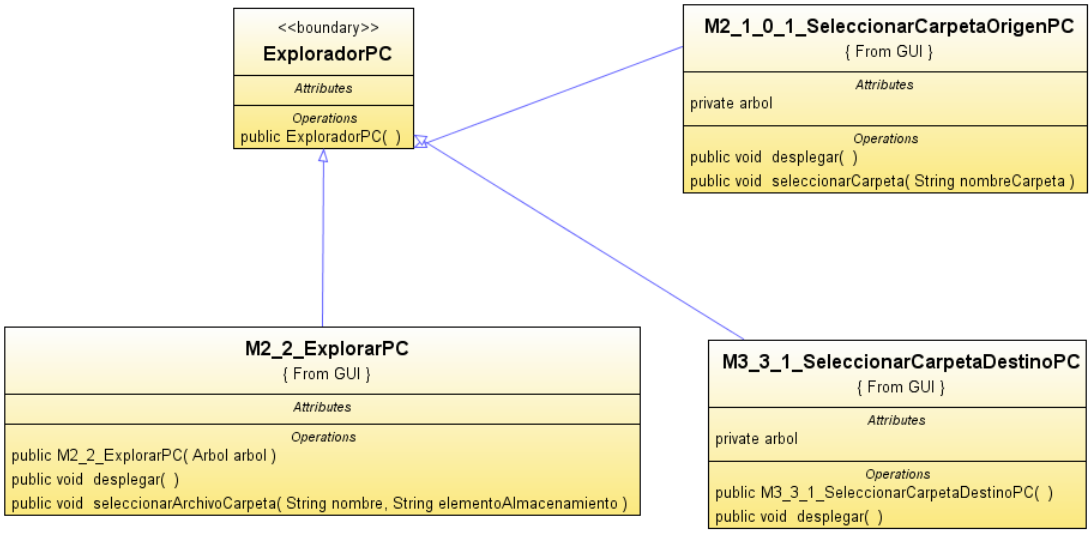


Figura 183. Diagrama de Clases Herencia FileBrowser

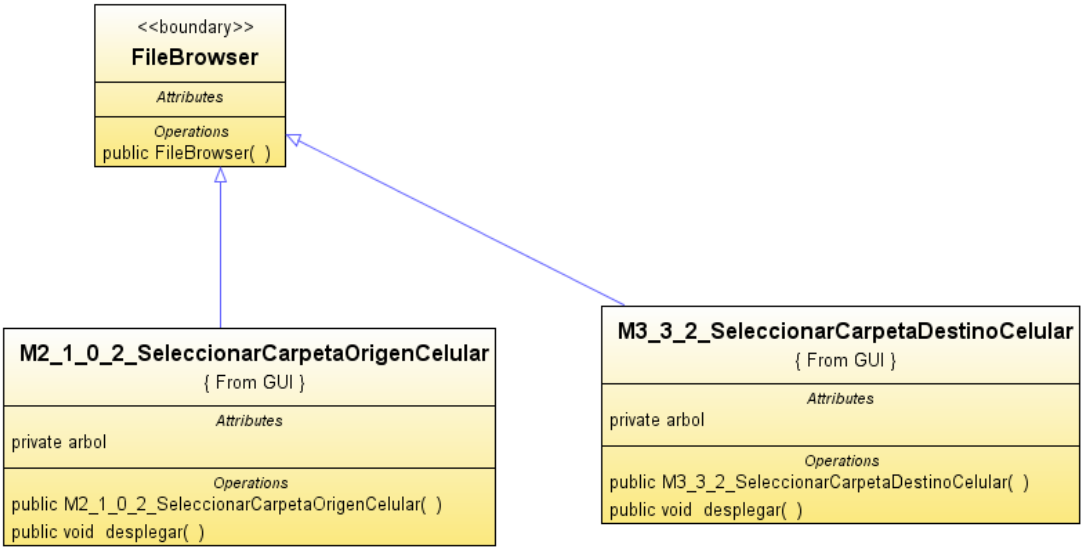


Figura 184. Diagrama de Clases Herencia GameCanvas

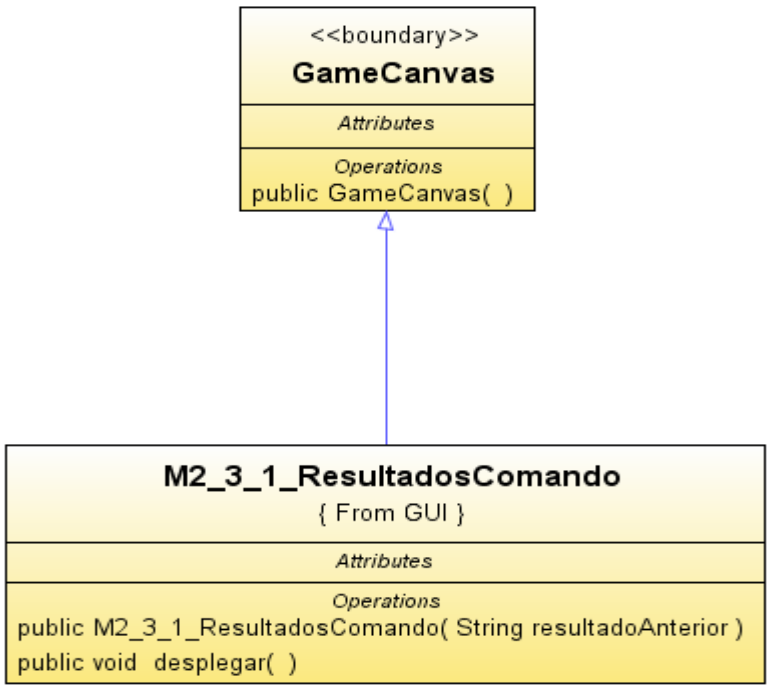


Figura 185. Diagrama de Clases Herencia List

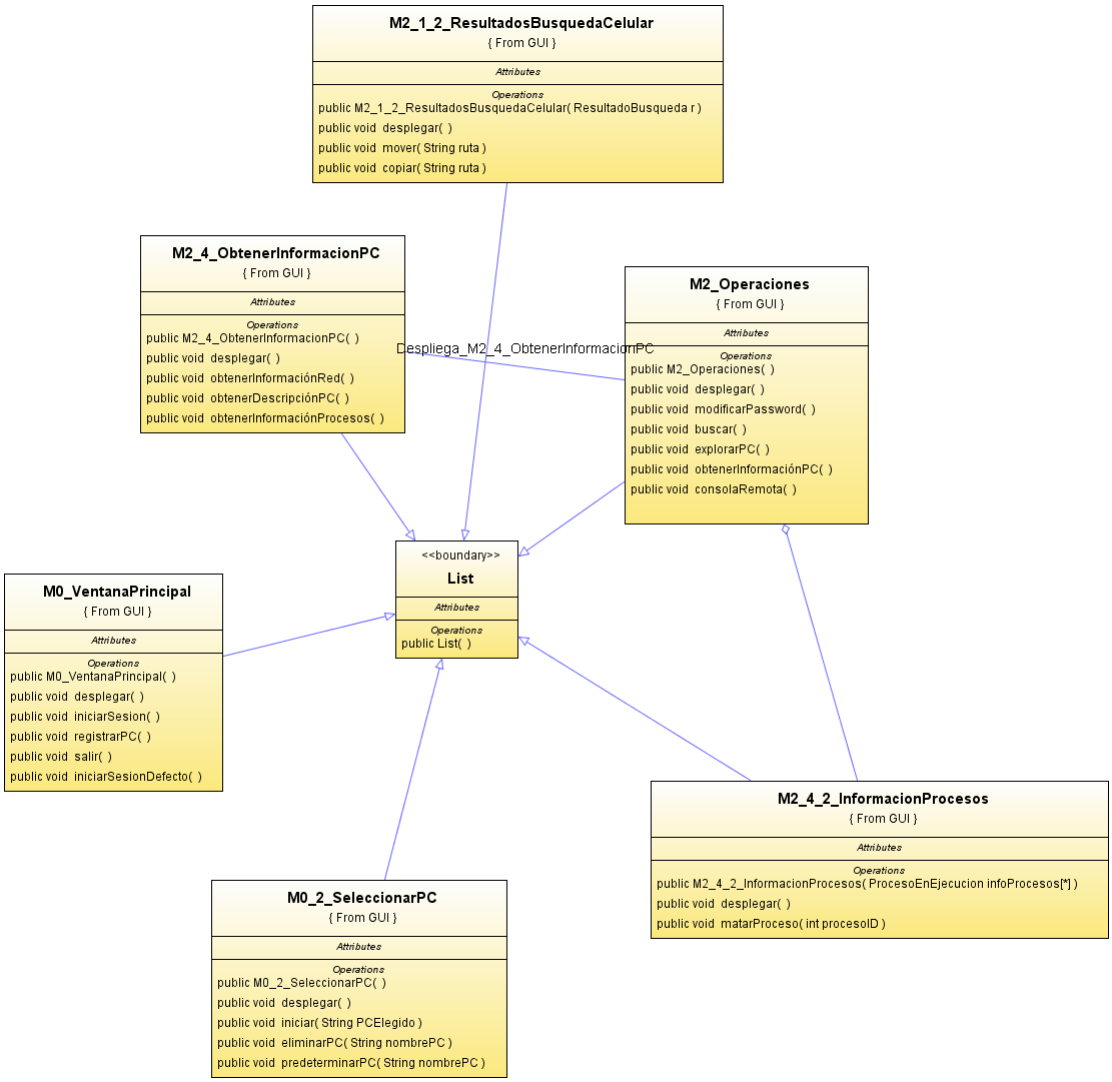


Figura 186. Diagrama de Clases Herencia List2

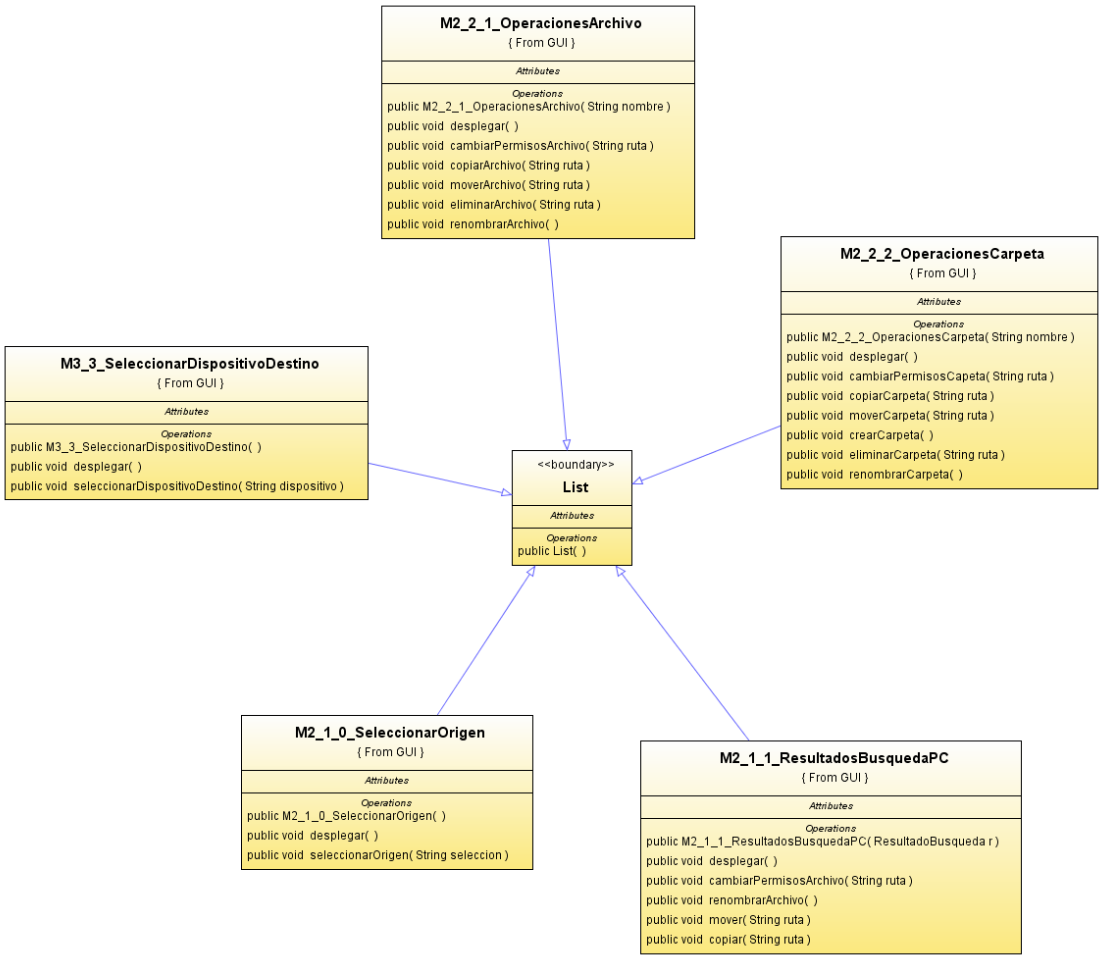


Figura 187. Diagrama de Clases Herencia Servidor

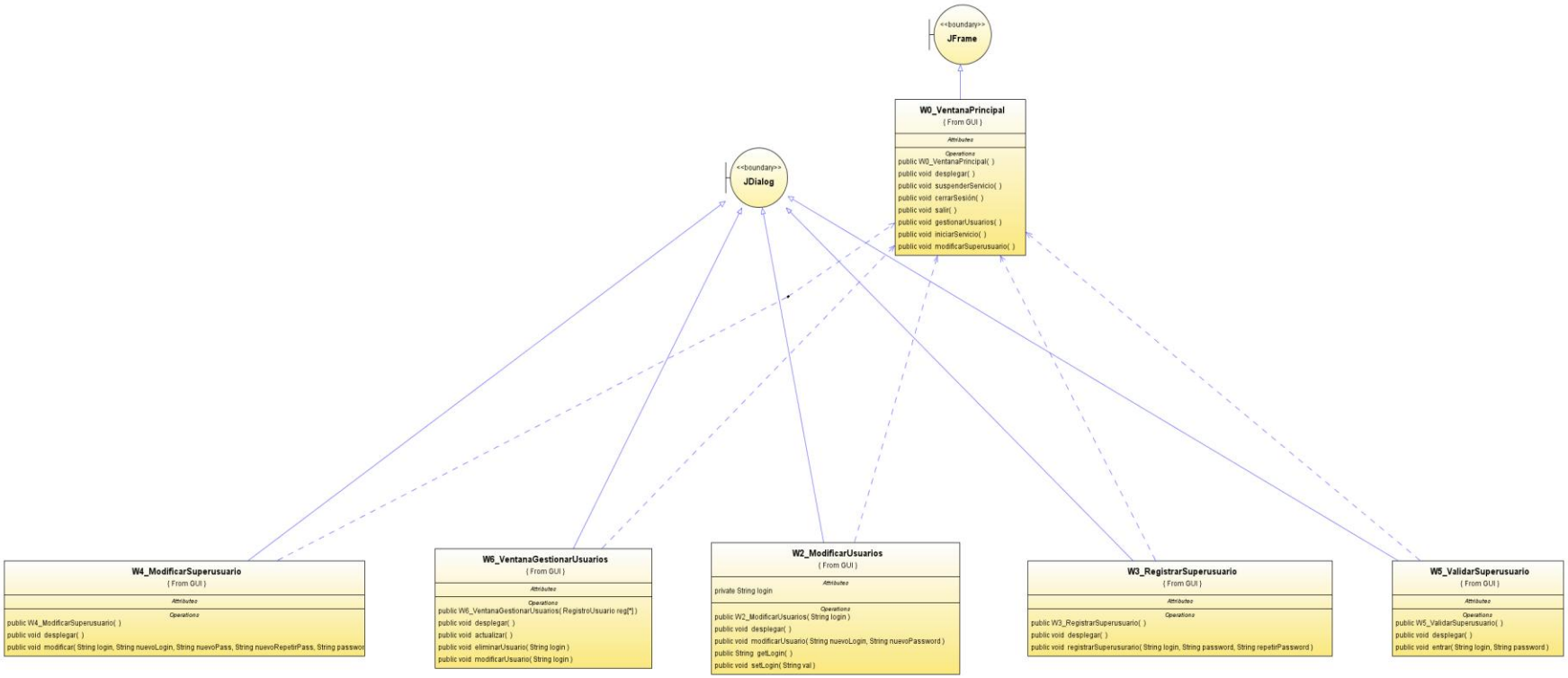


Figura 188. Diagrama de Clases entidades Manejador Consultas

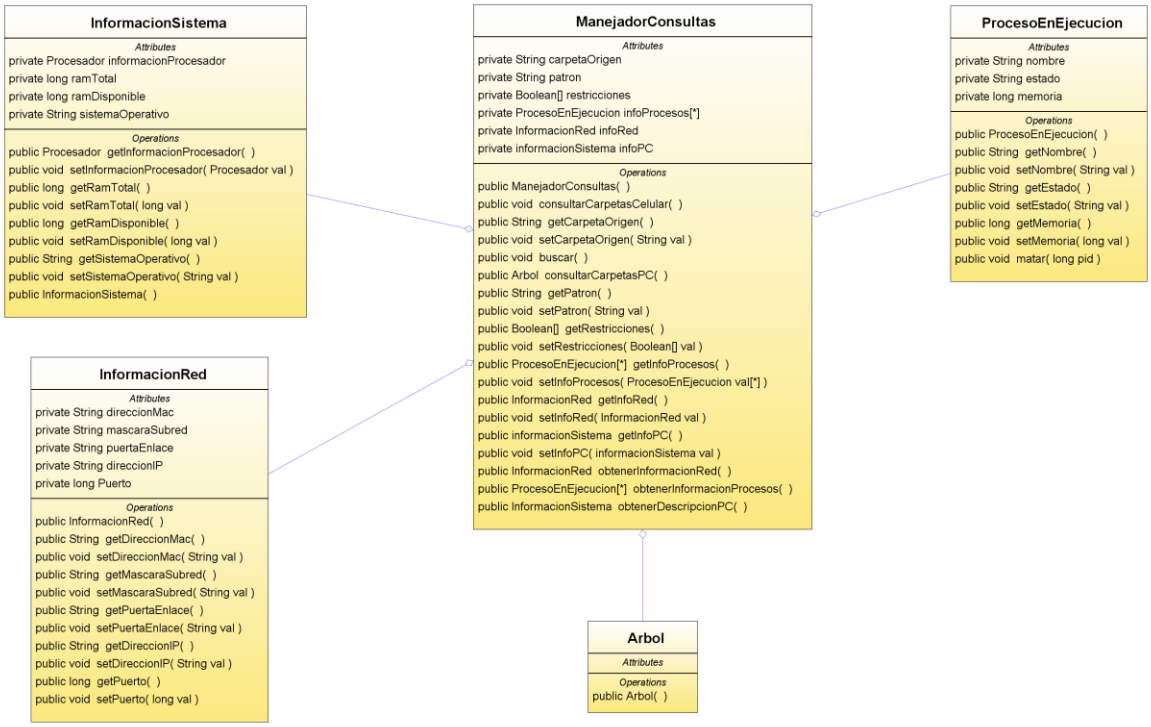


Figura 189. Diagrama de Clases Parte Herencia Form

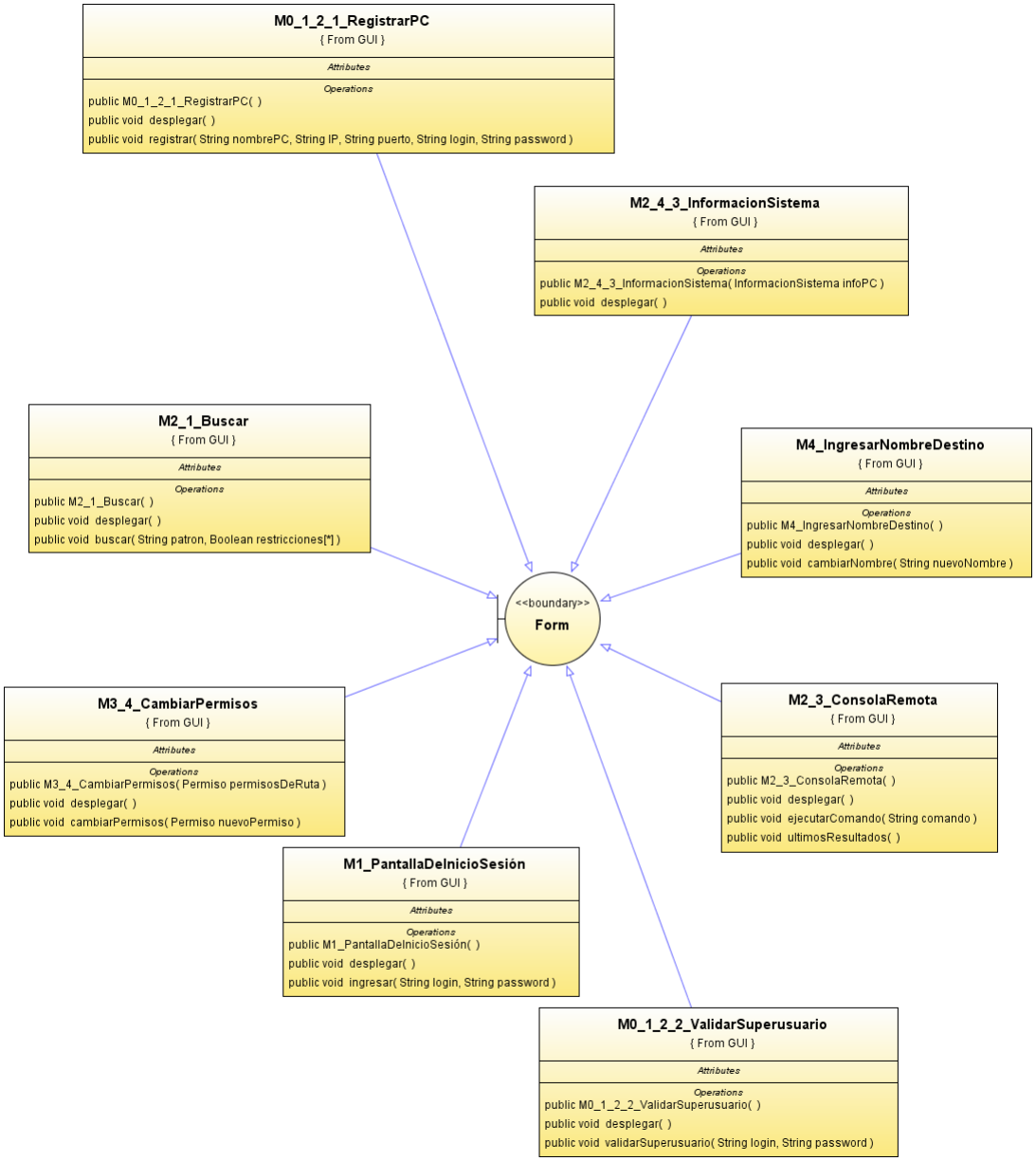
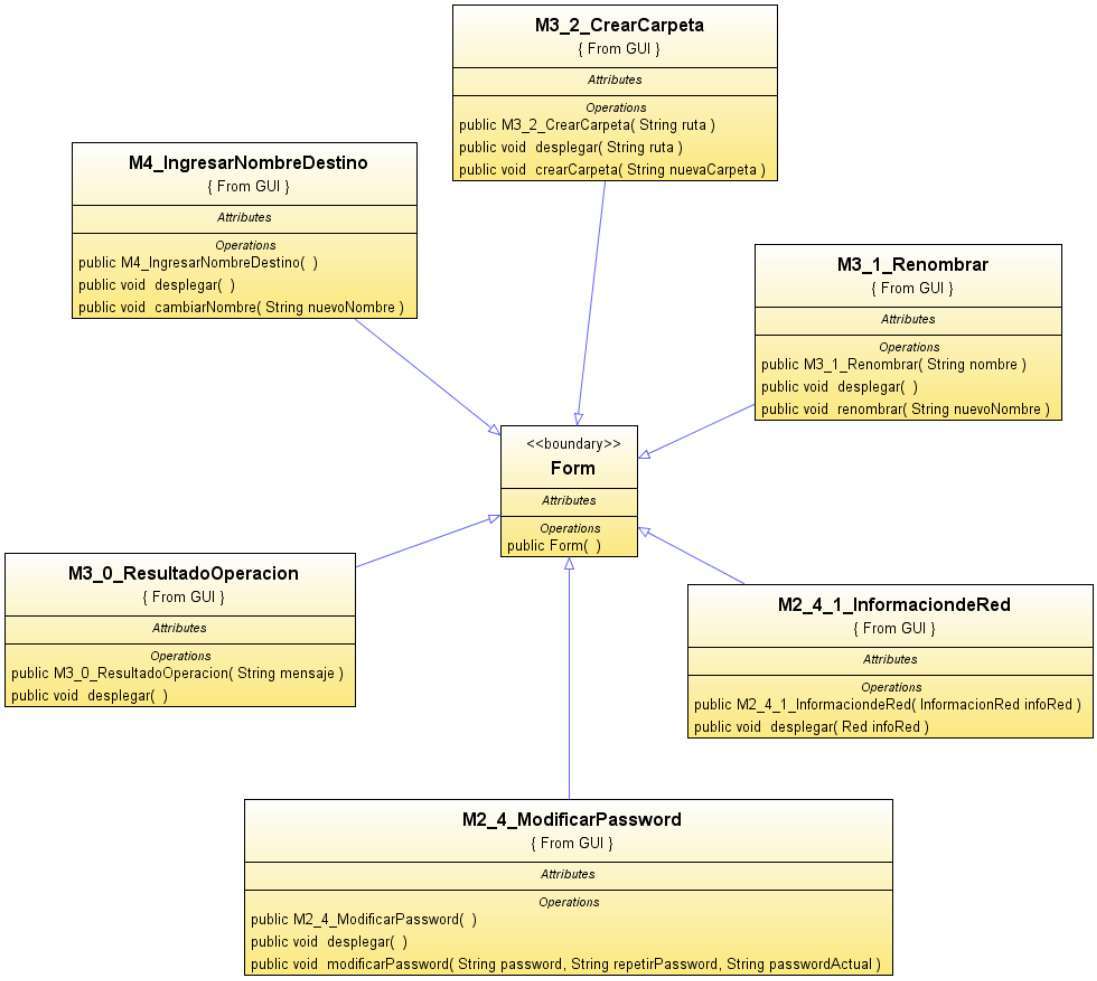


Figura 190. Diagrama de Clases Parte Herencia Form2



8. TARJETAS CRC

Las siguientes son las tarjetas CRC las cuales especifican en detalle las tareas que deben realizar los métodos y además que otras clases están involucradas en la realización de estas tareas.

Tabla 2. CRC M0_1_2_1_RegistrarPC

Clase	**M0_1_2_1_RegistrarPC**
Descripción	Ventana en el celular encargada de recibir los datos necesarios para registrar un PC, como lo son "Nombre del PC", "Dirección IP", "Puerto", "Login Usuario", "Password Usuario", "Repetir Password"
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
M0_1_2_1_RegistrarPC() **Crea una nueva instancia de la ventana M0_1_2_1_RegistrarPC**	
Desplegar() **Despliega la ventana para recibir los datos de registro de un nuevo PC con el cual conectarse.**	
public void registrar(String nombrePC, String IP, String puerto, String login, String password) **Hace el llamado a registrar(String nombrePC, String IP, String puerto, String login, String password) con la información ingresada por el usuario**	ManejadorRegistroUsuarioCliente

Tabla 3. CRC M0_1_2_2_ValidarSuperusuario

Clase	**M0_1_2_2_ValidarSuperusuario**
Descripción	Ventana en el celular encargada de recibir los datos necesarios para verificar

	el permiso de un Superusuario de registrar el PC bajo su administración. Recibe los datos "Login" y "Password" del Superusuario.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M0_1_2_2_ValidarSuperusuario( ) **Crea una nueva instancia de M0_1_2_2_ValidarSuperusuario**	
public void desplegar( ) **Muestra la ventana en la pantalla del celular en donde se recibe la información para validar el Superusuario.**	
public void validarSuperusuario(String login, String password) **Hace un llamado a public void validarSuperusuario(String login, String password)**	ManejadorRegistroUsuarioCliente

Tabla 4. CRC M0_2_SeleccionarPC

Clase	**M0_2_SeleccionarPC**
Descripción	Ventana en el celular encargada de desplegar una lista con los nombres de los PCs registrados en RegistroPC. Permite realizar las siguientes operaciones con el PC seleccionado: iniciar sesión, eliminarlo de RegistroPC y predeterminarlo.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M0_2_SeleccionarPC( ) **Crea una nueva instancia de la clase**	

M0_2_SeleccionarPC	
public void desplegar( ) **Muestra la ventana en el celular**	
public void iniciar(String PCElegido) **El usuario presiona el botón "iniciar" de "M0_2_SeleccionarPC", luego se hace llamado a setPCElegido del ManejadorSesiónCliente enviándole el PC elegido. Luego se crea y se despliega la ventana M1_PantallaDeInicioSesión**	ManejadorSesiónCliente, M1_PantallaDeInicioSesión
public void predeterminarPC(String nombrePC) **El usuario presiona el botón "Predeterminar", luego de PC de la lista. Luego se hace el llamado a predeterminarPC de ManejadorSesiónCliente enviándole dicho PC**	ManejadorSesiónCliente
public void eliminarPC(String nombrePC) **El usuario selecciona un nombre de PC y presiona el botón "eliminar", luego se hace un llamado a** **public void eliminarPC(RegistroPC nombrePC) de ManejadorSesiónCliente**	ManejadorSesiónCliente
public void predeterminarPC(String nombrePC) **Hace un llamado a public void setPCDefecto(String nombrePC) de ManejadorSesiónCliente con el PC elegido por el usuario**	ManejadorSesiónCliente

Tabla 5. CRC M0_VentanaPrincipal

Clase	**M0_VentanaPrincipal**
Descripción	Ventana en el celular encargada de mostrar las opciones para iniciar sesión y registrar un PC
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M0_VentanaPrincipal( ) **Crea una nueva instancia de M0_VentanaPrincipal**	**M0_VentanaPrincipal**
public void desplegar( )	

Despliega la ventana de las opciones del usuario antes de iniciar una sesión	
public void iniciarSesion( ) **El usuario presiona el botón "iniciar Sesión" de "M0_VentanaPrincipal", luego se hace un llamado a iniciarSesion( ) de ManejadorSesiónCliente**	ManejadorSesiónCliente
public void registrarPC( ) **Hace la creación de la M0_1_2_1_RegistrarPC y el despliegue de dicha ventana después de que el usuario hubiera presionado el botón RegistrarPC**	M0_1_2_1_RegistrarPC
public void salir( ) **Hace un llamado a la operación cerrarSesión( ) de ManejadorSesiónCliente**	ManejadorSesiónCliente
public void iniciarSesionDefecto( ) **El usuario presiona el botón "Iniciar Sesión por defecto" de "M0_VentanaPrincipal", luego se hace llamado a iniciarSesionDefecto de ManejadorSesiónCliente**	ManejadorSesiónCliente

Tabla 6. CRC M1_PantallaDeInicioSesión

Clase	**M1_PantallaDeInicioSesión**
Descripción	Ventana en el celular encargada de recibir el "Login" y el "Password" del usuario para permitirle el inicio de la sesión.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M1_PantallaDeInicioSesión(String estado) **Crea una nueva instancia de la clase M1_PantallaDeInicioSesión, si estado es eliminar entonces cambia la etiqueta del botón a iniciar en caso contrario la etiqueta vale validar**	
public void desplegar( ) **Despliega la pantalla en el celular**	

public void validar(String login, String password) **Hace un llamado a continuarEliminacion(String login, String password) de ManejadorSesiónCliente**	ManejadorSesiónCliente
public void ingresar(String login, String password) **El Usuario presiona el botón "Ingresar" de la ventana M1_PantallaDeInicioSesión, luego se hace el llamado a ingresar(String login, String password) del ManejadorSesiónCliente**	ManejadorSesiónCliente

Tabla 7. CRC M2_1_0_1_SeleccionarCarpetaOrigenPC

Clase	**M2_1_0_1_SeleccionarCarpetaOrigenPC**
Descripción	Ventana en el celular encargada de permitirle al usuario navegar por el sistema de archivos del PC y seleccionar una carpeta para luego realizar una búsqueda a partir de ella.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	ExploradorPC
Subclases	
Atributos	Arbol arbol
public M2_1_0_1_SeleccionarCarpetaOrigenPC( ) **Hace el llamado a consultarCarpetasPC( ) de ManejadorConsultas donde recibe el Arbol de carpetas del PC. Crea una nueva instancia de M2_1_0_1_SeleccionarCarpetaOrigenPC**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void seleccionarCarpeta(String nombreCarpeta) **Hace el llamado a setCarpetaOrigen(String carpetaOrigen) de ManejadorConsultas y luego hace el llamado a la operación buscar() de ManejadorConsultas**	ManejadorConsultas
public Arbol getArbol( )	

public void setArbol(Arbol val)	

Tabla 8. CRC M2_1_0_2_SeleccionarCarpetaOrigenCelular

Clase	**M2_1_0_2_SeleccionarCarpetaOrigenCelular**
Descripción	Ventana en el celular encargada de permitirle al usuario navegar por el sistema de archivos del Celular y seleccionar una carpeta para luego realizar una búsqueda a partir de ella.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	FileBrowser
Subclases	
Atributos	Arbol arbol
public M2_1_0_2_SeleccionarCarpetaOrigenCelular( ) **Hace el llamado a consultarCarpetasCelular( ) de ManejadorConsultas. Crea una nueva instancia de M2_1_0_2_SeleccionarCarpetaOrigenCelular**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void seleccionarCarpeta(String nombreCarpeta) **El usuario selecciona una carpeta y su nombre se utiliza para hacer el llamado a setCarpetaOrigen(String carpetaOrigen) de ManejadorConsultas, luego hace el llamado a buscar() de ManejadorConsultas**	ManejadorConsultas
public void setArbol(Arbol val)	
public Arbol getArbol( )	

Tabla 9. CRC M2_1_0_SeleccionarOrigen

Clase	**M2_1_0_SeleccionarOrigen**
Descripción	Ventana en el celular encargada de permitirle al usuario seleccionar el origen de una búsqueda, las opciones disponibles son:

	"Carpeta del PC", "Carpeta del Celular", "Todo el PC", "Todo el Celular".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M2_1_0_SeleccionarOrigen( ) **Crea una nueva instancia de M2_1_0_SeleccionarOrigen**	
public void desplegar( ) **Despliega la ventana en el celular, y recibe el origen de la búsqueda**	
public void seleccionarOrigen(String selección) **Si el usuario selecciona "Carpeta del PC" entonces crea y despliega la ventana M2_1_0_1_SeleccionarCarpetaOrigenPC** **Si el usuario selecciona "Carpeta del celular" entonces crea y despliega la ventana M2_1_0_2_SeleccionarCarpetaOrigenCelular** **Si el usuario selecciona "Todo el PC" o "Todo el Celular" entonces hace el llamado a setCarpetaOrigen(String carpetaOrigen) de ManejadorConsultas y luego hace el llamado a buscar( ) de ManejadorConsultas.**	M2_1_0_1_SeleccionarCarpetaOrigenPC, M2_1_0_2_SeleccionarCarpetaOrigenCelular, ManejadorConsultas

Tabla 10. CRC M2_1_1_ResultadosBusquedaPC

Clase	**M2_1_1_ResultadosBusquedaPC**
Descripción	Ventana en el celular encargada de mostrarle al usuario una lista de archivos y carpetas producto de una búsqueda. Permite realizar una de las siguientes operaciones sobre un archivo o carpeta: "Renombrar", "Mover", "Eliminar", "Cambiar Permisos", "Copiar", "Copiar Ruta".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde

Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M2_1_1_ResultadosBusquedaPC(ResultadoBusqueda r) **Crea una nueva instancia de M2_1_1_ResultadosBusquedaPC**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void cambiarPermisosArchivo(String ruta) **Hace el llamado a la operación obtenerPermisos(String ruta) de ManejadorOperaciones con la ruta elegida por el usuario**	ManejadorOperaciones
public void renombrarArchivo( ) **Hace el llamado a renombrar del ManejadorOperaciones enviándole la ruta del archivo a renombrar el cual fue seleccionado previamente en la misma ventana**	ManejadorOperaciones
public void mover(String ruta) **El usuario presiona el botón "mover", luego de elegir un archivo o carpeta de la lista de resultados. Luego se hace el llamado a mover de ManejadorOperaciones enviándole la ruta.**	ManejadorOperaciones
public void copiar(String ruta) **Hace un llamado a public void copiar(String ruta) de ManejadorOperaciones con la ruta del archivo o carpeta seleccionado por el usuario**	ManejadorOperaciones

Tabla 11. CRC M2_1_2_ResultadosBusquedaCelular

Clase	**M2_1_2_ResultadosBusquedaCelular**
Descripción	Ventana en el celular encargada de mostrarle al usuario una lista de archivos y carpetas producto de una búsqueda en el celular. Permite realizar una de las siguientes operaciones sobre un archivo o carpeta: "Mover" y "Copiar"
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta

Superclases	List
Subclases	
Atributos	
public M2_1_2_ResultadosBusquedaCelular(ResultadoBusqueda r) **Crea una nueva instancia de M2_1_2_ResultadosBusquedaCelular**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void mover(String ruta) **El usuario presiona el botón "mover", luego de elegir un archivo o carpeta de la lista de resultados. Luego se hace el llamado a mover de ManejadorOperaciones enviándole la ruta.**	ManejadorOperaciones
public void copiar(String ruta) **Hace un llamado a la operación copiar (String ruta) de ManejadorOperaciones con la ruta del archivo o carpeta seleccionado por el usuario.**	ManejadorOperaciones

Tabla 12. CRC M2_1_Buscar

Clase	**M2_1_Buscar**
Descripción	Ventana en el celular encargada de recibir los datos necesarios para realizar una búsqueda, como son el nombre del archivo o carpeta con comodines opcionalmente, y para filtrar el resultado de la búsqueda se elige si la salida debe ser solo directorios, con atributo oculto, con atributo sistema, o de solo lectura.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_1_Buscar( ) **Crea una nueva instancia de M2_1_Buscar**	
public void desplegar( ) **Despliega la ventana en el celular**	
public void buscar(String patron, Boolean	ManejadorConsultas,

restricciones[*]) **Hace el llamado a setPatron(String patrón) de ManejadorConsultas con el parametro patrón, hace el llamado a setRestricciones(Boolean restricciones[*]) de ManejadorConsultas con el parametro restricciones, y por último, crea y despliega M2_1_0_SeleccionarOrigen**	M2_1_0_SeleccionarOrigen

Tabla 13. CRC M2_2_1_OperacionesArchivo

Clase	**M2_2_1_OperacionesArchivo**
Descripción	Ventana en el celular encargada de permitir realizar una operación sobre un archivo, las operaciones disponibles son: "Renombrar", "Eliminar", "Mover", "Cambiar Permisos", "Copiar" y "Copiar Ruta".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M2_2_1_OperacionesArchivo(String ruta) **Crea una nueva instancia de la ventana M2_2_1_OperacionesArchivo**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void cambiarPermisosArchivo() **Hace el llamado a la operación obtenerPermisos(String ruta) de ManejadorOperaciones con la ruta elegida por el usuario**	ManejadorOperaciones
public void copiarArchivo() **Hace un llamado a la operación copiar(String ruta) de ManejadorOperaciones con la ruta del archivo seleccionado por el usuario.**	ManejadorOperaciones
public void moverArchivo() **El usuario presiona el botón "mover", luego de elegir un archivo o carpeta de la lista de archivos. Luego se hace el llamado a mover de ManejadorOperaciones enviándole la ruta.**	ManejadorOperaciones
public void eliminarArchivo()	ManejadorOperaciones

Hace un llamado a public void eliminarArchivo(String ruta) de ManejadorOperaciones	
public void renombrarArchivo( ) **Hace el llamado a renombrar del ManejadorOperaciones enviándole la ruta del archivo a renombrar el cual fue seleccionado previamente en la misma ventana**	ManejadorOperaciones

Tabla 14. CRC M2_2_2_OperacionesCarpeta

Clase	**M2_2_2_OperacionesCarpeta**
Descripción	Ventana en el celular encargada de permitir realizar una operación sobre una carpeta, las operaciones disponibles son: "Renombrar", "Crear", "Eliminar", "Mover", "Cambiar Permisos", "Copiar" y "Copiar Ruta".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M2_2_2_OperacionesCarpeta(String ruta) **Crea una nueva instancia de M2_2_2_OperacionesCarpeta**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void cambiarPermisosCapeta() **Hace el llamado a la operación obtenerPermisos(String ruta) de ManejadorOperaciones con la ruta elegida por el usuario**	ManejadorOperaciones
public void copiarCarpeta() **Hace un llamado a la operación copiar(String ruta) de ManejadorOperaciones con la ruta de la carpeta seleccionada por el usuario.**	ManejadorOperaciones
public void moverCarpeta() **El usuario presiona el botón "mover", luego de elegir un archivo o carpeta de la lista de archivos. Luego se hace el llamado a mover de ManejadorOperaciones enviándole la ruta.**	ManejadorOperaciones

public void crearCarpeta( ) **Crea y despliega la ventana M3_2_CrearCarpeta y luego hace un llamado a public void setRuta(String ruta) de ManejadorOperaciones**	ManejadorOperaciones, M3_2_CrearCarpeta
public void eliminarCarpeta() **Hace un llamado a public void eliminarCarpeta(String ruta) de ManejadorOperaciones**	ManejadorOperaciones
public void renombrarCarpeta( ) **Hace el llamado a renombrar del ManejadorOperaciones enviándole la ruta de la carpeta a renombrar el cual fue seleccionado previamente en la misma ventana**	ManejadorOperaciones

Tabla 15. CRC M2_2_ExplorarPC

Clase	**M2_2_ExplorarPC**
Descripción	Ventana en el celular encargada de permitirle al usuario navegar por el sistema de archivos del PC y seleccionar una carpeta o un archivo para luego realizar una operación sobre el objeto seleccionado.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	ExploradorPC
Subclases	
Atributos	
public M2_2_ExplorarPC(Arbol a) **Crea una nueva instancia de la ventana M2_2_ExplorarPC para mostrar el arbol**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void seleccionarArchivoCarpeta(String nombre, String elementoAlmacenamiento) **Si elementoAlmacenamiento == Carpeta entonces crea una nueva instancia de la ventana M2_2_2_OperacionesCarpeta usando el parámetro nombre, luego la despliega en el**	M2_2_2_OperacionesCarpeta, M2_2_2_OperacionesArchivo

celular **Si elementoAlmacenamiento == Archivo entonces crea una nueva instancia de la ventana M2_2_1_OperacionesArchivo usando el parámetro nombre, luego la despliega en el celular**	

Tabla 16. CRC M2_3_1_ResultadosComando

Clase	**M2_3_1_ResultadosComando**
Descripción	Ventana en el celular encargada de mostrarle al usuario el resultado de la ejecución de un comando de consola.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	GameCanvas
Subclases	
Atributos	
public M2_3_1_ResultadosComando(String resultadoAnterior) **Crea una nueva instancia de la ventana M2_3_1_ResultadosComando con el resultado de la ejecución del último comando**	
public void desplegar( ) **Muestra la ventana en el celular**	

Tabla 17. CRC M2_3_ConsolaRemota

Clase	**M2_3_ConsolaRemota**
Descripción	Ventana en el celular encargada de recibir un comando de consola para ser ejecutado en el PC
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_3_ConsolaRemota( )	

Crea una nueva instancia de M2_3_ConsolaRemota	
public void desplegar( ) **Muestra la ventana en el celular**	
public void ejecutarComando(String comando) **Hace un llamado a public void ejecutarComando(String comando) de ManejadorOperaciones**	ManejadorOperaciones
public void ultimosResultados( ) **Despliega la ventana M2_3_1_ResultadosComando**	M2_3_1_ResultadosComando

Tabla 18. CRC M2_4_1_InformaciondeRed

Clase	**M2_4_1_InformaciondeRed**
Descripción	Ventana en el celular encargada de mostrarle al usuario el resultado de la consulta de la información básica de red del PC.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_4_1_InformaciondeRed(InformacionRed infoRed) **Crea una nueva instancia de la ventana M2_4_1_InformaciondeRed para mostrar en el celular el objeto InfoRed**	
public void desplegar(Red infoRed) **Muestra la ventana en el celular**	

Tabla 19. CRC M2_4_2_InformacionProcesos

Clase	**M2_4_2_InformacionProcesos**
Descripción	Ventana en el celular encargada de mostrarle al usuario el resultado de la consulta de la información de los procesos en ejecución con los siguientes atributos:

	nombre del proceso, cantidad de memoria utilizada, estado.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M2_4_2_InformacionProcesos(ProcesoEnEjecucion infoProcesos[*]) **Crea una nueva instancia de la ventana M2_4_2_InformacionProcesos para mostrar infoProcesos en el celular**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void matarProceso(int procesoID) **Hace un llamado a la operación matarProceso(int procesoID) de ManejadorOperaciones, con el parametro procesoID**	ManejadorOperaciones

Tabla 20. CRC M2_4_3_InformacionSistema

Clase	**M2_4_3_InformacionSistema**
Descripción	Ventana en el celular encargada de mostrarle al usuario el resultado de la consulta de la descripción del PC, como puede ser sistema operativo, nombre del procesador, cantidad total de memoria RAM, memoria RAM disponible, etc.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_4_3_InformacionSistema(	

InformacionSistema infoPC) **Crea una nueva instancia de la ventana M2_4_3_InformacionSistema para mostrar el objeto infoPC en el celular**	
public void desplegar( ) **Muestra la ventana en el celular**	

Tabla 21. CRC M2_4_ModificarPassword

Clase	**M2_4_ModificarPassword**
Descripción	Ventana en el celular encargada de recibir el nuevo password del usuario actual, tiene el campo Repetir password para evitar errores de digitación.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_4_ModificarPassword( ) **Crea una nueva instancia de la clase M2_4_ModificarPassword**	
public void desplegar( ) **Despliega la ventana en el celular**	
public void modificarPassword(String password, String repetirPassword, String passwordActual) **El usuario presiona el botón "ok", luego de ingresar la información, nuevo password, repetir nuevo password y password actual. Luego se hace el llamado a modificar password de ManejadorRegistroUsuarioCliente enviándole estos tres parámetros**	ManejadorRegistroUsuarioCliente

Tabla 22. CRC M2_4_ObtenerInformacionPC

Clase	**M2_4_ObtenerInformacionPC**
Descripción	Ventana en el celular encargada de permitirle al usuario realizar las siguientes consultas: "Información de Red", "Información de los Procesos" e "Información del Sistema".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M2_4_ObtenerInformacionPC( ) **Crea una nueva instancia de la ventana M2_4_ObtenerInformacionPC**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void obtenerInformaciónRed( ) **Hace un llamado a la operación obtenerInformaciónRed( ) de ManejadorConsultas**	ManejadorConsultas
public void obtenerDescripciónPC( ) **Hace un llamado a la operación obtenerDescripciónPC( ) de ManejadorConsultas.**	ManejadorConsultas
public void obtenerInformaciónProcesos( ) **Hace un llamado a la operación obtenerInformaciónProcesos( ) de ManejadorConsultas**	ManejadorConsultas

Tabla 23. CRC M2_Operaciones

Clase	M2_Operaciones
Descripción	Ventana en el celular encargada de permitirle al usuario acceder a las siguientes opciones: "Buscar", "Explorar PC", "Consola remota", "Obtener Información PC" y "Modificar Password".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	
public M2_Operaciones( ) **Crea una nueva instancia de la clase M2_Operaciones**	
public void desplegar( ) **Despliega la ventana en el celular, con las principales operaciones**	
public void modificarPassword( ) **El usuario presiona el botón "Modificar Password", luego se crea y se despliega la ventana M2_4_ModificarPassword**	M2_4_ModificarPassword
public void buscar( ) **Crea y despliega la ventana M2_1_Buscar en el celular**	M2_1_Buscar
public void explorarPC( ) **Hace un llamado a la operación consultarArchivosCarpetas( ) de ManejadorConsultas**	ManejadorConsultas
public void obtenerInformaciónPC( ) **Crea y despliega la ventana M2_4_ObtenerInformacionPC en el celular**	M2_4_ObtenerInformacionPC
public void consolaRemota( ) **Crea y despliega la ventana M2_3_ConsolaRemota**	M2_3_ConsolaRemota

Tabla 24. CRC M3_0_ResultadoOperacion

Clase	M3_0_ResultadoOperacion

Descripción	Ventana en el celular encargada de mostrarle al usuario el resultado de la ejecución de una operación, "Renombrar", "Crear", "Eliminar", "Mover", "Cambiar Permisos" y "Copiar".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M3_0_ResultadoOperacion(String mensaje) **Crea una nueva instancia de M3_0_ResultadoOperacion con el fin de mostrar el parametro mensaje en la pantalla del celular**	
public void desplegar( ) **Muestra la ventana en el celular**	

Tabla 25. CRC M3_1_Renombrar

Clase	**M3_1_Renombrar**
Descripción	Ventana en el celular encargada de recibir el nuevo nombre del archivo o carpeta a renombrar
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M3_1_Renombrar(String nombre) **Crea una nueva instancia de M3_1_Renombrar donde muestra el parámetro nombre, que es el nombre actual del archivo o carpeta a renombrar**	

public void desplegar( ) **Muestra la ventana en el celular**	
public void renombrar(String nuevoNombre) **El usuario ingresa la información del nuevo nombre del archivo o carpeta y presiona el botón "Renombrar" entonces se hace un llamado a renombrar(String nuevoNombre) del ManejadorOperaciones enviándole el nuevo nombre**	ManejadorOperaciones

Tabla 26. CRC M2_4_2_InformacionProcesos

Clase	**M2_4_2_InformacionProcesos**
Descripción	Ventana en el celular encargada de mostrarle al usuario el resultado de la consulta de la información de los procesos en ejecución con los siguientes atributos: nombre del proceso, cantidad de memoria utilizada, estado.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_4_2_InformacionProcesos(ProcesoEnEjecucion infoProcesos[*]) **Crea una nueva instancia de la ventana M2_4_2_InformacionProcesos para mostrar infoProcesos en el celular**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void matarProceso(int procesoID) **Hace un llamado a la operación matarProceso(int procesoID) de ManejadorOperaciones, con el parametro procesoID**	ManejadorOperaciones

Tabla 27. CRC M2_4_3_InformacionSistema

Clase	M2_4_3_InformacionSistema
Descripción	Ventana en el celular encargada de mostrarle al usuario el resultado de la consulta de la descripción del PC, como puede ser sistema operativo, nombre del procesador, cantidad total de memoria RAM, memoria RAM disponible, etc.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_4_3_InformacionSistema(InformacionSistema infoPC) **Crea una nueva instancia de la ventana M2_4_3_InformacionSistema para mostrar el objeto infoPC en el celular**	

Tabla 28. CRC M2_4_ModificarPassword

Clase	M2_4_ModificarPassword
Descripción	Ventana en el celular encargada de recibir el nuevo password del usuario actual, tiene el campo Repetir password para evitar errores de digitación.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	

Atributos	
public M2_4_ModificarPassword( ) **Crea una nueva instancia de la clase M2_4_ModificarPassword**	
public void desplegar( ) **Despliega la ventana en el celular**	
public void modificarPassword(String password, String repetirPassword, String passwordActual) **El usuario presiona el botón "ok", luego de ingresar la información, nuevo password, repetir nuevo password y password actual. Luego se hace el llamado a modificarPassword(String password, String repetirPassword, String passwordActual) de ManejadorRegistroUsuarioCliente enviándole los mismos parámetros**	ManejadorRegistroUsuarioCliente

Tabla 29. CRC M2_4_ObtenerInformacionPC

Clase	**M2_4_ObtenerInformacionPC**
Descripción	Ventana en el celular encargada de permitirle al usuario realizar las siguientes consultas: "Información de Red", "Información de los Procesos" e "Información del Sistema".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_4_ObtenerInformacionPC( ) **Crea una nueva instancia de la ventana M2_4_ObtenerInformacionPC**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void obtenerInformaciónRed( ) **Hace un llamado a la operación**	

obtenerInformaciónRed( ) de ManejadorConsultas	
public void obtenerDescripciónPC( ) **Hace un llamado a la operación obtenerDescripciónPC( ) de ManejadorConsultas.** **public void obtenerInformaciónProcesos( )** **Hace un llamado a la operación obtenerInformaciónProcesos( ) de ManejadorConsultas**	

Tabla 30. CRC M2_Operaciones

Clase	**M2_Operaciones**
Descripción	Ventana en el celular encargada de permitirle al usuario acceder a las siguientes opciones: "Buscar", "Explorar PC", "Consola remota", "Obtener Información PC" y "Modificar Password".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M2_Operaciones( ) **Crea una nueva instancia de la clase M2_Operaciones**	
public void desplegar( ) **Despliega la ventana en el celular, con las principales operaciones**	
public void modificarPassword( ) **El usuario presiona el botón "Modificar Password", luego se crea y se despliega la ventana M2_4_ModificarPassword**	M2_4_ModificarPassword
public void buscar( ) **Crea y despliega la ventana M2_1_Buscar en el celular**	
public void explorarPC( ) **Hace un llamado a la operación consultarArchivosCarpetas( ) de ManejadorConsultas y recibe un arbol de**	ManejadorConsultas, M2_2_ExplorarPC

carpetas para crear una nueva instancia de la ventana M2_2_ExplorarPC y luego la despliega en el celular	
public void obtenerInformaciónPC( ) **Crea y despliega la ventana M2_4_ObtenerInformacionPC en el celular**	M2_4_ObtenerInformacionPC
public void consolaRemota( ) **Crea y despliega la ventana M2_3_ConsolaRemota en el celular**	M2_3_ConsolaRemota

Tabla 31. CRC M3_0_ResultadoOperacion

Clase	**M3_0_ResultadoOperacion**
Descripción	Ventana en el celular encargada de mostrarle al usuario el resultado de la ejecución de una operación, "Renombrar", "Crear", "Eliminar", "Mover", "Cambiar Permisos" y "Copiar".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M3_0_ResultadoOperacion(String mensaje) **Crea una nueva instancia de M3_0_ResultadoOperacion con el fin de mostrar el parametro mensaje en la pantalla del celular**	
public void desplegar( ) **Muestra la ventana en el celular**	

Tabla 32. CRC M3_1_Renombrar

Clase	**M3_1_Renombrar**
Descripción	Ventana en el celular encargada de recibir el nuevo nombre del archivo o carpeta a renombrar.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta

Superclases	Form
Subclases	
Atributos	
public M3_1_Renombrar(String nombre)	
public void desplegar( )	
public void renombrarArchivo(String nombre, String nuevoNombre)	
public void renombrarCarpeta(String nuevoNombre)	
public void renombrar(String nuevoNombre) **El usuario ingresa la información del nuevo nombre y presiona el botón "Renombrar. Luego se hace un llamado a renombrar1 del ManejadorOperaciones enviándole el nuevo nombre**	

Tabla 33. CRC M3_2_CrearCarpeta

Clase	**M3_2_CrearCarpeta**
Descripción	Ventana en el celular encargada de recibir el nombre de la subcarpeta a crear.
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M3_2_CrearCarpeta( ) **Crea una nueva instancia de M3_2_CrearCarpeta**	
public void desplegar() **Muestra la ventana en el celular**	
public void crearCarpeta(String nuevaCarpeta) **Hace un llamado a la operación crearCarpeta(String nuevaCarpeta) de ManejadorOperaciones**	ManejadorOperaciones

Tabla 34. CRC M3_3_1_SeleccionarCarpetaDestinoPC

Clase	**M3_3_1_SeleccionarCarpetaDestinoPC**

Descripción	Ventana en el celular encargada de permitirle al usuario navegar por el sistema de archivos del PC y seleccionar una carpeta como destino de una operación "Copiar" o "Mover"
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	ExploradorPC
Subclases	
Atributos	Arbol arbol
public M3_3_1_SeleccionarCarpetaDestinoPC(Arbol consulta) **Crea una nueva instancia de la clase M3_3_1_SeleccionarCarpetaDestinoPC.**	
public void desplegar( ) **Despliega la ventana en el celular**	
public void seleccionarDestino(String destino) **El usuario elije entre una de las carpetas que se listaban en la ventana y presiona OK luego se hace el llamado a setDestino en ManejadorOperaciones enviándole el nombre de la carpeta seleccionada. Luego se hace un llamado a copiarFase3( ) de ManejadorOperaciones.**	ManejadorOperaciones
public Arbol getArbol( )	
public void setArbol(Arbol val)	

Tabla 35. CRC M3_3_2_SeleccionarCarpetaDestinoCelular

Clase	**M3_3_2_SeleccionarCarpetaDestinoCelular**
Descripción	Ventana en el celular encargada de permitirle al usuario navegar por el sistema de archivos del celular y seleccionar una carpeta como destino de una operación "Copiar" o "Mover"
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	FileBrowser
Subclases	

Atributos	Arbol arbol
public M3_3_2_SeleccionarCarpetaDestinoCelular( ) **Crea una nueva instancia de la clase M3_3_2_SeleccionarCarpetaDestinoCelular. Luego hace el llamado a la función consultarCarpetas de ManejadorOperaciones de la cual recibe un objeto tipo Arbol**	
public void desplegar( ) **Despliega la ventana en el celular**	
public void seleccionarDestino(String destino) **El usuario elije entre una de las carpetas que se listaban en la ventana y presiona OK luego se hace el llamado a setDestino en ManejadorOperaciones enviándole el nombre de la carpeta seleccionada**	
public void setArbol(Arbol val)	
public Arbol getArbol( )	

Tabla 36. CRC M3_3_SeleccionarDispositivoDestino

Clase	**M3_3_SeleccionarDispositivoDestino**
Descripción	Ventana en el celular encargada de solicitar el dispositivo destino de la operación actual. Los dispositivos son: "PC" o "Celular".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	List
Subclases	
Atributos	

public M3_3_SeleccionarDispositivoDestino( ) **Crea una nueva instancia de la ventana M3_3_SeleccionarDispositivoDestino**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void seleccionarDispositivoDestino(String dispositivo) **Hace un llamado a setDispositivoDestino(String dispositivo) de ManejadorOperaciones y luego hace un llamado a copiarFase2( ) de ManejadorOperaciones**	ManejadorOperaciones

Tabla 37. CRC M3_4_CambiarPermisos

Clase	**M3_4_CambiarPermisos**
Descripción	Ventana en el celular encargada de recibir los nuevos permisos de un archivo o carpeta, se pueden cambiar los permisos: "Solo Lectura", "Oculto", "Sistema" y "Almacenamiento".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M3_4_CambiarPermisos(Permiso permisosDeRuta) **Crea una nueva ventana M3_4_CambiarPermisos y muestra permisosDeRuta en el celular**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void cambiarPermisos(Permiso nuevoPermiso) **Hace el llamado a la operación setNuevoPermiso(Permiso nuevoPermiso) de**	ManejadorOperaciones

ManejadorOperaciones	

Tabla 38. CRC M4_IngresarNombreDestino

Clase	**M4_IngresarNombreDestino**
Descripción	Ventana en el celular encargada de recibir el Nombre que tendrá archivo o carpeta cuando llegue al destino en una operación "Copiar" o "Mover".
Módulo	Interfaces/cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public M4_IngresarNombreDestino( ) **Crea una nueva instancia de la ventana M4_IngresarNombreDestino**	
public void desplegar( ) **Muestra la ventana en el celular**	
public void cambiarNombre(String nuevoNombre) **Hace un llamado a la operación setNuevoNombre(String nuevoNombre) de ManejadorOperaciones. Luego hace un llamado a copiarFase4( ) de ManejadorOperaciones**	ManejadorOperaciones

Tabla 39. CRC InterfaceCliente

Clase	**InterfaceCliente**
Descripción	Clase encargada de realizar solicitudes al PC para realizar las operaciones requeridas por el usuario.
Módulo	Interfaces/cliente
Estereotipo	Borde
Propiedades	Concreta
Superclases	Form
Subclases	
Atributos	
public InterfaceCliente( )	

public Boolean validarUsuario(RegistroUsuario r) **Envía un mensaje validarUsuario al Servidor enviándole el objeto de tipo RegistroUsuario**	InterfaceServidor
public void cerrarSesión( ) **Envía el mensaje cerrarSesión( ) al Servidor**	InterfaceServidor
public ResultadoBusqueda buscar(String patrón, Boolean restricciones[*]) **Envía el mensaje buscar(String patrón, Boolean restricciones[*]) al Servidor en el PC solicitando un objeto de tipo ResultadoBusqueda el cual contiene una lista de archivos y carpetas, cuyos nombres concuerda con el parametro patrón y cumple con las restricciones entregadas.**	InterfaceServidor
public Arbol consultarArchivosCarpetas( ) **Enviar un mensaje consultarArchivosCarpetas( ) al Servidor**	InterfaceServidor
public Permiso obtenerPermisos(String ruta) **Envía el mensaje obtenerPermisos(String ruta) al Servidor y recibe los permisos del archivo o carpeta ubicado en ruta**	InterfaceServidor
public String cambiarPermisos(String ruta, Permiso nuevoPermiso) **Envía el mensaje cambiarPermisos(String ruta, Permiso nuevoPermiso) al Servidor y recibe un String con un mensaje informando del éxito o fracaso de la operación**	InterfaceServidor
public Arbol consultarCarpetas( ) **Envía el mensaje consultarCarpetas( ) al Servidor en el PC, solicitando el arbol de carpetas del PC**	InterfaceServidor
public void copiarArchivo(byte origen[*], String destino) Envía un mensaje copiarArchivo(byte origen[*], String destino) al Servidor	InterfaceServidor
public String matarProceso(int procesoID) **Envía un mensaje matarProceso(int procesoID) al Servidor y recibe un String con el éxito o fracaso de la operación**	InterfaceServidor
public byte[*] obtenerArchivo(String origen) **Envía el mensaje obtenerArchivo(String origen) al Servidor y recibe un arreglo de bytes con el contenido del archivo con nombre "origen"**	InterfaceServidor

public Boolean validarSuperusuario(RegistroSuperusuario reg) **Manda un mensaje de validarSuperusuario al servidor solicitándole un Boolean que informe acerca de si el Superusuario es valido**	InterfaceServidor
public void crearUsuario(RegistroUsuario reg) **Envía el mensaje crearUsuario al servidor enviándole el objeto de tipo RegistroUsuario con la información necesaria para crear un nuevo usuario**	InterfaceServidor
public InformacionSistema obtenerDescripcionPC( ) **Envía el mensaje obtenerDescripcionPC ( ) al Servidor. Recibe y retorna un objeto de tipo InformacionSistema con el resultado de la consulta**	InterfaceServidor
public InformacionRed obtenerInformaciónRed( ) **Envía el mensaje obtenerInformacionRed ( ) al Servidor. Recibe y retorna un objeto de tipo InformacionRed con el resultado de la consulta**	InterfaceServidor
public ProcesoEnEjecucion[*] obtenerInformacionProcesos( ) **Envía el mensaje obtenerInformacionProcesos( ) al Servidor. Recibe y retorna un arreglo con el listado de procesos en ejecución en el PC**	InterfaceServidor
public String crearCarpeta(String ruta, String nuevaCarpeta) **Envía el mensaje crearCarpeta(String ruta, String nuevaCarpeta) al Servidor. Recibe y retorna un String con un mensaje informando del éxito o fracaso de la operación**	InterfaceServidor
public String ejecutarComando(String comando) **Envía el mensaje ejecutarComando(String comando) al Servidor. Recibe y retorna un String con el resultado de la ejecución del comando en el PC.**	InterfaceServidor
public String eliminarArchivo(String ruta) **Envía el mensaje eliminarArchivo(String ruta) al Servidor. Recibe y retorna un String con un mensaje informando del éxito o fracaso de la operación**	InterfaceServidor

public String eliminarCarpeta(String ruta) **Envía el mensaje eliminarCarpeta (String ruta) al Servidor. Recibe y retorna un String con un mensaje informando del éxito o fracaso de la operación**	InterfaceServidor
public void iniciarSesion( ) **Manda el mensaje iniciarSesion al servidor**	InterfaceServidor
public void modificarPassword(String login, String passwordActual, String password) **Manda un mensaje modificarPassword al Servidor enviándole un los parámetros passwordActual y NuevoPassword**	InterfaceServidor
public void eliminarOrigen(String ruta) **Manda el mensaje eliminarOrigen(String ruta) al Servidor**	InterfaceServidor
public ResultadoBusqueda buscar(String patrón, Boolean restricciones[*], String rutaBusqueda) **envía el mensaje buscar(String patrón, Boolean restricciones[*], String rutaBusqueda) al Servidor en el PC y recibe el resultado de la búsqueda**	InterfaceServidor
public void crearCarpeta(String nuevaCarpeta) **Envía el mensaje crearCarpeta(String nuevaCarpeta) al Servidor**	InterfaceServidor
public String[*] obtenerListaArchivos(String ruta) **Envía el mensaje String[*] obtenerListaArchivos(String ruta) al Servidor. Recibe y retorna un arreglo de String con las rutas de los archivos que están dentro de la carpeta especificada por ruta**	InterfaceServidor
public void copiar(String origen, String destino) **Envía el mensaje copiar(String origen, String destino) al Servidor**	InterfaceServidor
String[*] obtenerCarpetasVacias(String ruta) **Envía el mensaje obtenerCarpetasVacias(String ruta) al Servidor. Recibe y retorna un arreglo de String con las rutas relativas de las carpetas vacías que están dentro de la**	InterfaceServidor

carpeta ruta.	
public String renombrar(String ruta, String nuevoNombre) **Envía el mensaje renombrar(String ruta, String nuevoNombre) a InterfaceServidor y recibe un String con un mensaje que indica si la operación tuvo éxito o no.**	InterfaceServidor

Tabla 40. CRC InterfaceRecursosCelular

Clase	**InterfaceRecursosCelular**
Descripción	Es una clase que actúa como interfaz entre los recursos del celular y el resto de la aplicación
Módulo	Interfaces/Cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public InterfaceRecursosCelular( )	
public String obtenerIMEI( ) **Solicita a "InterfaceRecursosCelular" la información del IMEI del celular.**	
public ResultadoBusqueda buscar(String patrón, Boolean restricciones[*], String rutaBusqueda) **Solicita a Recursos Celular un objeto de tipo ResultadoBusqueda el cual contiene una lista de archivos y carpetas, cuyos nombres concuerda con el parametro patrón y cumple con las restricciones entregadas y además está entre la carpeta con la ruta rutaBusqueda.**	
public void copiar(String origen, String destino) **Solicita a "Recursos Celular" tome la información existente de origen y la copie en destino del celular.**	
public Arbol consultarCarpetas( ) **Solicita a "Recursos Celular" la lista de archivos y**	Arbol

carpetas del celular, este lo retorna en forma de un objeto tipo Arbol.	
public void crearArchivo (byte contenido[*], String nombre) Solicita a "Recursos Celular" cree un archivo con la información de contenido y cuyo nombre sea "nombre" en el celular.	
public void eliminarOrigen(String ruta) Solicita a "Recursos Celular" eliminar el archivo ruta de el sistema de archivos del celular	
public String[*] obtenerListaArchivos(String ruta) Solicita a "Recursos Celular" una lista que representa al sistema de archivos del celular.	
public ResultadoBusqueda buscar(String patrón, Boolean restricciones) Solicita a "Recursos Celular" un objeto de tipo ResultadoBusqueda el cual contiene una lista de archivos y carpetas, cuyos nombres concuerda con el parametro patrón y cumple con las restricciones entregadas.	
public byte[*] obtenerArchivo(String origen) Solicita a "Recursos Celular" un arreglo de bytes con el contenido del archivo con nombre igual al valor del parámetro origen. Por último retorna el arreglo de bytes	
public String[*] obtenerCarpetasVacias(String ruta) Solicita a "Recursos Celular" un arreglo listando los nombres de las carpetas vacías dentro de la ruta.	

Tabla 41. CRC InterfaceRegistroPCs

Clase	**InterfaceRegistroPCs**
Descripción	Es una clase que actúa como interfaz entre los PCs registrados y el resto de la aplicación
Módulo	Interfaces/Cliente/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	
Subclases	

Atributos	
public InterfaceRegistroPCs( )	
public RegistroPC[*] consultarPCs( )	
public void eliminarPC(RegistroPC nombrePC)	
public RegistroPC obtenerPCDefecto( ) **Solicita a "InterfaceRecursosPCs" el Registro del PC que está definido como el por defecto al cual el usuario se conectará más fácilmente.**	
public void crear(RegistroPC reg) **Solicita a "Registro PCs" almacenar el registro de PC reg.**	
public void predeterminarPC(String nombrePC) **Solicita a "Registro PCs" que busque el nombre del PC entregado y lo defina como predeterminado, de esta manera el PC que este actualmente predeterminado perderá este estado.**	

Tabla 42. CRC W0_VentanaPrincipal

Clase	W0_VentanaPrincipal
Descripción	Ventana en el PC encargada de mostrarle al Superusuario las opciones "Gestionar Usuarios", "Modificar Superusuario", "Cerrar Conexión", "Iniciar servicio", "Ayuda", "Salir", además muestra el historial de las últimas acciones realizadas por los usuarios que se han conectado.
Módulo	Interfaces/Servidor/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	JFrame
Subclases	
Atributos	

public W0_VentanaPrincipal( ) **Crea una nueva instancia de la ventana W0_VentanaPrincipal**	
public void desplegar( ) **Despliega la ventana en el PC**	
public void suspenderServicio( ) **El usuario presiona el botón "Suspender Servicio". Luego se hace el llamado a suspenderServicio de ManejadorServicio**	ManejadorServicio
public void cerrarSesión( ) **Hace un llamado a la operación liberarRecursos( ) de ManejadorSesiónServidor**	ManejadorSesiónServidor
public void salir( ) **Hace un llamado a la operación suspenderServicio( ) de ManejadorSesiónServidor, luego cierra la aplicación.**	ManejadorSesiónServidor
public void gestionarUsuarios( ) **El Superusuario presiona el botón "Gestionar Usuarios" de la ventana principal, luego se hace el llamado a gestionarUsuarios del ManejadorRegistroUsuarioServidor**	ManejadorRegistroUsuarioServidor
public void iniciarServicio( ) **El Superusuario presiona el botón iniciar servicio de la ventana principal, luego se hace el llamado a iniciarServicio del ManejadorServicio**	ManejadorServicio
public void modificarSuperusuario( ) **El Superusuario presiona el botón "Modificar Superusuario", luego se crea y se despliega la ventana W4_ModificarSuperusuario**	W4_ModificarSuperusuario

Tabla 43. CRC W2_ModificarUsuarios

Clase	**W2_ModificarUsuarios**
Descripción	Ventana en el PC encargada de recibir los datos necesarios para modificar un usuario, permite modificar el login y el password de un usuario.

Módulo	Interfaces/Servidor/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	JFrame
Subclases	
Atributos	String login
public W2_ModificarUsuarios(String login) **Crea una nueva instancia de la clase W2_ModificarUsuarios**	
public void desplegar( ) **Despliega la ventana en el PC**	
public void modificarUsuario(String nuevoLogin, String nuevoPassword, String repetirNuevoPassword) **Compara ambos nuevoPassword y repetirNuevoPassword si son iguales:** **Hace el llamado a modificarUsuario del InterfaceRegistroUsuarios enviándole el RegistroUsuario actual, la información del nuevo login y la información de nuevo password.**	InterfaceRegistroUsuarios
public String getLogin( )	
public void setLogin(String val)	

Tabla 44. CRC W3_RegistrarSuperusuario

Clase	**W3_RegistrarSuperusuario**
Descripción	Ventana en el PC encargada de recibir los datos necesarios para registrar el Superusuario por primera vez: "Login", "Password" y "Repetir Password"
Módulo	Interfaces/Servidor/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	JFrame
Subclases	
Atributos	
public W3_RegistrarSuperusuario( ) **Crea una nueva instancia de la clase W3_RegistrarSuperusuario**	
public void desplegar( )	

Despliega la ventana en el PC	
public void registrarSuperusurario(String login, String password, String repetirPassword) **El Superusuario rellena los datos login, password y repetirPassword los cuales serán utilizados como información de registro del Superusuario.** **Se comparan ambos password y repetirPassword si son iguales se hace el llamado a registrar enviándole el login y el password**	ManejadorRegistroSuperusuario

Tabla 45. CRC W4_ModificarSuperusuario

Clase	**W4_ModificarSuperusuario**
Descripción	
Módulo	Interfaces/Servidor/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	JFrame
Subclases	
Atributos	
public W4_ModificarSuperusuario( ) **Crea una nueva instancia de la clase W4_ModificarSuperusuario**	
public void desplegar( ) **Despliega la ventana en el PC**	
public void modificar (String login, String nuevoLogin, String nuevoPass, String nuevoRepetirPass, String passwordActual) **Verifica que nuevoPass y nuevoRepetirPass sean iguales, si es así hace el llamado a modificar del ManejadorRegistroSuperusuario enviándole la nueva información y el password y el login actual ingresado por el Superusuario, si el usuario selecciono no cambiar al login o al password se envía dicha información en forma nula.**	ManejadorRegistroSuperusuario

Tabla 46. CRC W5_ValidarSuperusuario

Clase	W5_ValidarSuperusuario
Descripción	Ventana en el PC encargada de recibir los datos necesarios para autenticar el Superusuario: "Login" y "Password", permite realizar 5 intentos fallidos antes cerrar la aplicación.
Módulo	Interfaces/Servidor/GUI
Estereotipo	Borde
Propiedades	Concreta
Superclases	JFrame
Subclases	
Atributos	
public W5_ValidarSuperusuario( ) **Crea una nueva instancia de la clase W5_ValidarSuperusuario**	
public void desplegar( ) **Despliega la ventana en el PC**	
public void entrar(String login, String password) **El Superusuario presiona el botón "Entrar", pero antes rellena los datos login y password los cuales serán utilizados como información de validación para ingresar a la aplicación servidor. Luego se hace la creación de un objeto entidad de tipo RegistroSuperusuario a través de la información anteriormente ingresada y se hace llamado validarSuperusuario de ManejadorRegistroSuperusuario enviándole el objeto.**	RegistroSuperusuario, ManejadorRegistroSuperusuario

Tabla 47. CRC W6_VentanaGestionarUsuarios

Clase	W6_VentanaGestionarUsuarios
Descripción	Ventana en el PC encargada de mostrar una lista de los usuarios actuales del PC, incluye los atributos "Login" e "Imei" de RegistroUsuario en las columnas de la tabla. Permite realizar las siguientes operaciones sobre un registro de usuario: "Eliminar" y "Modificar".
Módulo	Interfaces/Servidor/GUI

Estereotipo	Borde
Propiedades	Concreta
Superclases	JFrame
Subclases	
Atributos	
public W6_VentanaGestionarUsuarios(RegistroUsuario reg[*]) **Crea una nueva instancia de la clase W6_VentanaGestionarUsuarios, luego rellena la tabla con la información de los usuarios registrados en el servidor**	
public void desplegar( ) **Despliega la ventana en el PC**	
public void actualizar(String nuevoLogin, String nuevoPassword) **Actualiza la tabla con la lista de usuarios para presentar al Superusuario. Por medio de la nueva información del Usuario, login y password.**	
public void eliminarUsuario(String login) **EL SuperUsuario selecciona uno de los usuarios registrados y presiona el botón eliminar, luego hace el llamado a eliminar de ManejadorRegistroUsuarioServidor**	ManejadorRegistroUsuarioServidor
public void modificarUsuario(RegistroUsuario registroUsuario) **Crea y despliega la ventana W2_ModificarUsuarios**	W2_ModificarUsuarios

Tabla 48. CRC InterfaceRecursosPC

Clase	**InterfaceRecursosPC**
Descripción	Es una clase que actúa como interfaz entre los recursos del PC y el resto de la aplicación
Módulo	Interfaces/Servidor
Estereotipo	Borde
Propiedades	Concreta
Superclases	
Subclases	
Atributos	

public InterfaceRecursosPC( )	
public String ejecutarComando(String comando) Solicita ejecutar en el servidor el comando	
public void crearCarpeta(String nombre, String subcarpeta) **Crea en la carpeta "nombre" la subcarpeta "subcarpeta "en el PC**	
public String eliminarCarpeta(String nombre) **Solicita a "Recursos PC" eliminar la carpeta "nombre".**	
public Permiso obtenerPermisos(String ruta) **Solicita a "Recursos PC" los permisos de acceso del archivo o carpeta ubicado en ruta y crea y retorna un objeto de tipo Permiso con el resultado**	
public String cambiarPermisos(String ruta, Permiso nuevoPermiso) **Solicita a "Recursos PC" cambiar los permisos de acceso del archivo o carpeta ubicado en ruta. Construye y retorna un String con un mensaje informando el éxito o fracaso de la operación.**	
public String matarProceso(int procesoID) **Solicita a "Recursos PC" finalizar un proceso en ejecución, crea un String con un mensaje informando el éxito o fracaso de la operación.**	
public InformacionSistema obtenerDescripciónPC( ) Solicita a "Recursos PC" un resumen de la información del PC, con el resultado crea una instancia de la clase InformacionSistema y la retorna	
public Arbol consultarCarpetas( ) Solicita a "Recursos PC" una representación del sistema de archivos en forma de un objeto tipo Arbol.	
public ResultadoBusqueda buscar(String patrón, Boolean restricciones[*]) Solicita a "Recursos PC" buscar los archivos que cumplan con el patrón y las restricciones entregadas	
public void copiarArchivo(byte origen[*], String destino) Solicita a "Recursos PC" copiar en destino el contenido de origen.	

public byte[*] obtenerArchivo(String origen) **Solicita a "Recursos PC" buscar los archivos que cumplan con el patrón y las restricciones entregadas**	
public InformacionRed obtenerInformacionRed( ) **Solicita a "Recursos PC" un resumen de la información de red del PC, construye un objeto de tipo InformacionRed con el resultado y lo retorna.**	
public ProcesoEnEjecucion[*] obtenerInformacionProcesos( ) **Solicita a RecursosPC un listado con los procesos actualmente en ejecución en el PC, entonces crea un arreglo de objetos ProcesoEnEjecucion y lo retorna.**	
public String renombrar(String ruta, String nuevoNombre) **Crea un comando de renombrar para la línea de comandos de Windows y luego la ejecuta con los parámetros de la ruta y el nuevo nombre del archivo**	
public void eliminarOrigen(String ruta) **Solicita a "Recursos PC" eliminar el archivo ruta de el sistema de archivos del celular**	
public ResultadoBusqueda buscar(String patrón, Boolean restricciones[*], String rutaBusqueda) **Solicita realizar una búsqueda de archivos y carpetas con los parámetros patrón y las restricciones, en la ruta rutaBusqueda en Recursos PC.**	
public void crearCarpeta(String nuevaCarpeta) Crea la carpeta nuevaCarpeta en el sistema de archivos del PC siendo nuevaCarpeta la ruta completa de la carpeta a crear.	
public String[*] obtenerListaArchivos(String ruta) Solicita a "Recursos PC" retornar el conjunto de archivos encontrados dentro de la carpeta ruta incluyendo las de las subcarpetas.	
public void copiar(String origen, String destino) Solicita a "Recursos PC" copiar la información de origen en destino del PC	

public String[*] obtenerCarpetasVacias(String ruta) **Solicita a "Recursos PC" un listado con las subcarpetas vacías de la carpeta con nombre igual al valor del parámetro ruta. Luego retorna el listado**	

Tabla 49. CRC InterfaceRegistroSuperusuario

Clase	**InterfaceRegistroSuperusuario**
Descripción	Es una clase que actúa como interfaz entre la información del Superusuario y el resto de la aplicación
Módulo	Interfaces/Servidor
Estereotipo	Borde
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public InterfaceRegistroSuperusuario( )	
public Boolean validarSuperusuario(RegistroSuperusuario reg) **Solicita a "Registro Superusuario" por medio de una consulta SQL validar los datos ingresados por un posible SuperUsuario**	
public Boolean existeAlgunSuperusuario( ) **Solicita a "Registro Superusuario" por medio de una consulta SQL verificar si existe un Superusuario registrado.**	
public void registrar(RegistroSuperusuario reg) **Solicita a "Registro Superusuario" por medio de una consulta SQL registrar el Superusuario.**	
public void actualizar(RegistroSuperusuario nuevoReg) Solicita a "Registro Superusuario" por medio de una consulta SQL modificar el registro existente del Superusuario.	

Tabla 50. CRC InterfaceRegistroUsuarios

Clase	**InterfaceRegistroUsuarios**

Descripción	Es una clase que actúa como interfaz entre la información del usuario y el resto de la aplicación
Módulo	Interfaces/Servidor
Estereotipo	Borde
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public InterfaceRegistroUsuarios( )	
public Boolean validarUsuario(RegistroUsuario r) Verifica que el registro r corresponda con un usuario existente.	
public void modificarUsuario (String nuevoLogin, String nuevoPassword, RegistroUsuario registroUsuario) **Verifica que el nuevo login no exista ya en RegistroUsuarios , luego busca en el registro que posea el login actual y procede a actualizar su información.**	RegistroUsuarios
public RegistroUsuario[*] consultarUsuarios( ) **Solicita a RegistroUsuarios la información de todos los usuarios registrados en el servidor por medio de una consulta SQL**	RegistroUsuarios
public void eliminar(String login) Solicita a RegistroUsuarios eliminar el registro del login registrado	RegistroUsuarios
public void crearUsuario(RegistroUsuario reg) **Solicita a RegistroUsuarios que cree una nueva entrada en la lista de usuarios registrados por medio del objeto entregado de tipo RegistroUsuario.**	RegistroUsuarios
public void validarUsuario(String login, String passwordActual) **Solicita a RegistroUsuarios que verifique si el registro entregado es válido dentro de los posibles clientes que pueden conectarse al servicio realizando para ello una consulta SQL.**	RegistroUsuarios

Tabla 51. CRC InterfaceServidor

Clase	**InterfaceServidor**
Descripción	Clase encargada de recibir las solicitudes del celular para realizar las operaciones requeridas por el usuario.
Módulo	Interfaces/Servidor
Estereotipo	Borde
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public InterfaceServidor( )	
public Boolean validarUsuario(RegistroUsuario r) **Hace el llamado a ejecutarConsulta( ) recibiendo de él un objeto RegistroUsuario esperando de ello un Boolean que informe acerca de la validez del usuario**	RegistroUsuario,
public void cerrarSesión( ) **Hace un llamado a la operación cerrarSesión( ) de ManejadorSesiónServidor**	ManejadorSesiónServidor
public ResultadoBusqueda buscar(String patrón, Boolean restricciones[*]) **Hace el llamado a la operación buscar(String patrón, Boolean restricciones[*]) de InterfaceRecursosPC**	InterfaceRecursosPC
public Arbol consultarArchivosCarpetas( ) **Hace el llamado a consultarArchivosCarpetas de la InterfaceRecursosPC solicitando un objeto tipo árbol.**	InterfaceRecursosPC,
public Permiso obtenerPermisos(String ruta) **Hace el llamado a la operaciónobtenerPermisos(String ruta) de InterfaceRecursosPC y recibe un objeto de tipo Permiso con los permisos del archivo o carpeta ubicado en ruta.**	InterfaceRecursosPC
public String cambiarPermisos(String ruta, Permiso nuevoPermiso) **Hace el llamado a la operacióncambiarPermisos(String ruta, Permiso nuevoPermiso) de InterfaceRecursosPC y recibe un objeto de tipo Permiso con los permisos del archivo o carpeta**	InterfaceRecursosPC

ubicado en ruta.	
public Arbol consultarCarpetas( ) **Hace el llamado al método consultarCarpetas( ) de InterfaceRecursosPC y recibe el arbol de carpetas del PC.**	InterfaceRecursosPC
public void copiarArchivo(byte origen[*], String destino) Hace el llamado a CopiarArchivo de InterfaceRecursosPC	InterfaceRecursosPC
public Boolean validarSuperusuario(RegistroSuperusuario reg) **Envía el mensaje validarSuperusuario(RegistroSuperusuario reg) del ManejadorRegistroSuperusuario**	ManejadorRegistroSuperusuario
public void crearUsuario(RegistroUsuario reg) **Manda un mensaje de crearUsuario a** **ManejadorRegistroUsuarioServidor enviándole un objeto de tipo RegistroUsuario.**	ManejadorRegistroUsuarioServidor
public InformacionSistema obtenerDescripcionPC( ) Hace un llamado a la operación obtenerDescripciónPC( ) de InterfaceRecursosPC. **Recibe y retorna un objeto de tipo InformaciónRed como resultado de la consulta.**	InterfaceRecursosPC
public InformacionRed obtenerInformacionRed( ) **Hace un llamado a la operación obtenerInformacionRed( ) de InterfaceRecursosPC. Recibe un objeto de tipo InformaciónRed con el resultado de la consulta.**	InterfaceRecursosPC
public ProcesoEnEjecucion[*] obtenerInformacionProcesos( ) **Hace un llamado a la operación obtenerInformacionProcesos( ) de InterfaceRecursosPC**	InterfaceRecursosPC
public String matarProceso(int procesoID) **Hace un llamado a la operación matarProceso(int procesoID) de InterfaceRecursosPC y recibe un String con el éxito o fracaso de la operación.**	InterfaceRecursosPC
public String crearCarpeta(String ruta, String nuevaCarpeta) Hace el llamado a crearCarpeta con los parámetros de la ruta y el nombre de la nueva carpeta a crear.	InterfaceRecursosPC
public String ejecutarComando(String comando)	InterfaceRecursosPC

Hace el llamado a ejecutarComando enviándole la información del comando a ejecutar.	
public String eliminarArchivo(String ruta) **Hace el llamado a eliminarArchivo enviándole la información de la ruta del archivo a eliminar.**	InterfaceRecursosPC
public String eliminarCarpeta(String ruta) **Hace el llamado a eliminarCarpeta enviándole la información de la ruta de la carpeta a eliminar.**	InterfaceRecursosPC
public void iniciarSesion( ) **Hace el llamado a iniciarSesion del ManejadorSesiónServidor**	ManejadorSesiónServidor
public void modificarPassword(String login, String passwordActual, String password) **Hace el llamado a modificarPassword del ManejadorRegistroUsuarioServidor**	ManejadorRegistroUsuarioServidor
public String renombrar(String ruta, String nuevoNombre) **Hace el llamado a renombrar de la InterfaceRecursosPC enviándole la ruta del archivo y el nuevo nombre.**	InterfaceRecursosPC
public void eliminarOrigen(String ruta) **Hace el llamado a eliminarOrigen enviándole la información de la ruta del archivo a eliminar.**	InterfaceRecursosPC
public ResultadoBusqueda buscar(String patrón, Boolean restricciones[*], String rutaBusqueda) **Hace el llamado a buscar(String patrón, Boolean restricciones[*], String rutaBusqueda) de InterfaceRecursosPC y recibe el resultado de la búsqueda**	InterfaceRecursosPC
public void crearCarpeta(String nuevaCarpeta)	InterfaceRecursosPC
public String[*] obtenerListaArchivos(String ruta)	InterfaceRecursosPC
public String[*] obtenerCarpetasVacias(String ruta) **Hace el llamado a obtenerCarpetasVacias de InterfaceRecursosPC con la ruta de la carpeta donde se buscarán las subcarpetas vacías**	InterfaceRecursosPC

Tabla 52. ManejadorConsultas

Clase	ManejadorConsultas
Descripción	Controla todas las consultas que requiera el usuario, como búsquedas, obtener información del PC, obtener información de red, obtener información de los procesos, además consultar archivos y carpetas
Módulo	Manejadores\cliente
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	public String patrón, private Boolean restricciones, public String CarpetaOrigen, Arbol arbol, InformacionRed infoRed, InformacionSistema infoPC
public ManejadorConsultas( )	
public Arbol consultarCarpetasPC( ) **Hace el llamado a consultarArchivosCarpetas( ) de InterfaceCliente donde recibe el arbol de carpetas del PC**	InterfaceCliente
public void obtenerInformaciónRed( ) **Hace un llamado a la operación obtenerInformaciónRed( ) de InterfaceCliente y recibe un objeto de tipo InformaciónRed con el resultado de la consulta y con este objeto crea y despliega la ventana M2_4_1_InformaciondeRed**	M2_4_1_InformaciondeRed
public void obtenerInformaciónProcesos( ) **Hace un llamado a la operación obtenerInformacionProcesos( ) de InterfaceCliente y recibe un arreglo de objetos ProcesoEnEjecucion con el listado de los procesos que se están ejecutando en el PC, luego crea y despliega la ventana M2_4_2_InformacionProcesos con dicho arreglo.**	M2_4_2_InformacionProcesos
public void obtenerDescripciónPC( ) **Hace un llamado a la operación**	M2_4_3_InformacionSistema

obtenerDescripcionPC( ) de InterfaceCliente. **Recibe un objeto de tipo InformacionSistema como resultado de la consulta. Con este objeto crea y despliega la ventana M2_4_3_InformacionSistema**	
public void buscar( ) **Si el atributo carpetaOrigen indica como dispositivo el PC entonces hace el llamado a buscar(String patrón, Boolean restricciones[*]) y con el resultado de la búsqueda crea y despliega la ventana M2_1_1_ResultadosBusquedaPC** **Si el atributo carpetaOrigen indica como dispositivo el Celular entonces hace el llamado a buscar(String patrón, Boolean restricciones[*]) y con el resultado de la búsqueda crea y despliega la ventana M2_1_2_ResultadosBusquedaCelular**	M2_1_1_ResultadosBusquedaPC
public String getPatron( )	
public void setPatron(String patron)	
public Boolean getRestricciones( )	
public void setRestricciones(Boolean[] restricciones)	
public String getCarpetaOrigen( )	
public void setCarpetaOrigen(String carpetaOrigen)	
public Arbol getArbol( )	
public void setArbol(Arbol val)	
public InformacionRed getInfoRed( )	
public void setInfoRed(InformacionRed val)	
public InformacionSistema getInfoPC( )	
public void setInfoPC(InformacionSistema val)	
public ProcesoEnEjecucion getInfoProcesos( )	
public void setInfoProcesos(ProcesoEnEjecucion val)	
public void consultarArchivosCarpetas( ) **Hace el llamado a consultarArchivosCarpetas de la InterfaceCliente de lo cual recibe un objeto**	M2_2_ExplorarPC, InterfaceCliente

tipo Arbol el cual utiliza para crear y desplegar la ventana M2_2_ExplorarPC	

Tabla 53. ManejadorOperaciones

Clase	**ManejadorOperaciones**
Descripción	Controla las operaciones sobre archivos y carpetas, ejecución de comandos de consola y matar procesos.
Módulo	Manejadores\cliente
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	private Permiso nuevoPermiso, String ruta, String destino, String dispositivoDestino, String nuevoNombre
public ManejadorOperaciones( )	
public String getNuevoNombre()	
public void setNuevoNombre(String aNuevoNombre)	
public String getDispositivoDestino()	
public void setDispositivoDestino(String aDispositivoDestino)	
public void eliminarCarpeta(String ruta) Hace el llamado a eliminarCarpeta(String ruta) de la InterfaceCliente, luego de lo cual Crea y despliega la ventana M3_0_ResultadoOperacion.	M3_0_ResultadoOperacion, InterfaceCliente
public void matarProceso(int procesoID) Hace un llamado a la operación matarProceso(int procesoID) de InterfaceCliente y recibe un String con un mensaje indicando el resultado de la operación y con este mensaje entonces crea y despliega la ventana M3_0_ResultadoOperacion.	InterfaceCliente, M3_4_CambiarPermisos, M3_0_ResultadoOperacion
public void eliminarArchivo(String ruta) Hace el llamado a eliminarArchivo(String ruta) de la InterfaceCliente, luego de lo cual Crea y despliega la ventana M3_0_ResultadoOperacion.	M3_0_ResultadoOperacion, InterfaceCliente
public void obtenerPermisos(String ruta) Hace el llamado a la operación obtenerPermisos(String ruta) de	InterfaceCliente, M3_4_CambiarPermisos, M3_0_ResultadoOperacion

InterfaceCliente y recibe los permisos actuales del archivo o carpeta ubicado en ruta en un objeto de tipo Permiso. Con este objeto crea una nueva instancia de la ventana M3_4_CambiarPermisos y luego la despliega.	
public void cambiarPermisos(Permiso nuevoPermiso) **Hace el llamado a cambiarPermisos(String ruta, Permiso nuevoPermiso) de InterfaceCliente, con el atributo ruta y el parámetro nuevoPermiso. Finalmente con el String retornado se crea y despliega la ventana M3_0_ResultadoOperacion.**	InterfaceCliente, M3_0_ResultadoOperacion
public void setDestino(String destino) **Hace el llamado a consultarCarpetas de la InterfaceRecursosCelular recibiendo un objeto de tipo Arbol**	InterfaceRecursosCelular
public void renombrar(String nuevoNombre) **Hace el llamado a renombrar de la InterfaceCliente enviándole la ruta del archivo y el nuevo nombre. Luego crea y despliega la ventana M3_0_ResultadoOperacion**	InterfaceCliente, M3_0_ResultadoOperacion
public void setNuevoPermiso(Permiso nuevoPermiso)	
public void copiarFase1(String ruta) **Crea y despliega la ventana M3_3_SeleccionarDispositivoDestino en el celular**	M3_3_SeleccionarDispositivoDestino
public void copiarFase2( ) **Si dispositivoDestino == Celular entonces hace un llamado a consultarCarpetas( ) de InterfaceRecursosCelular y recibe un árbol de carpetas, para luego crear y desplegar la ventana M3_3_2_SeleccionarCarpetaDestinoCelular en el celular** **Si dispositivo == PC entonces hace un llamado a consultarCarpetas( ) de InterfaceCliente y recibe un árbol de carpetas, para crear y desplegar la ventana M3_3_1_SeleccionarCarpetaDestinoPC**	InterfaceRecursosCelular, M3_3_1_SeleccionarCarpetaDestinoPC, M3_3_2_SeleccionarCarpetaDestinoCelular
public void copiarFase3( ) **Crea y despliega la ventana M4_IngresarNombreDestino en el celular**	M4_IngresarNombreDestino

public void copiarFase4( ) **Si dispositivoDestino == celular y DispositivoOrigen == Celular entonces se hace un llamado a copiar(String origen, String destino) de InterfaceRecursosCelular** **Si dispositivoDestino == Celular y DispositivoOrigen == PC y elementoAlmacenamiento == Carpeta entonces se hace un llamado a obtenerListaArchivos(String ruta) de InterfaceCliente y se recibe un arreglo de String con las rutas relativas de los archivos que están dentro de la carpeta ruta y sus subcarpetas.** **Para cada archivo del arreglo se realiza la copia individual. Después se hace un llamado a la operación obtenerCarpetasVacias(String ruta) de InterfaceCliente y se recibe un arreglo de String con las rutas relativas de las subcarpetas vacías dentro de la carpeta ruta.** **Para cada subcarpeta vacía se hace un llamado a crearCarpeta(String nuevaCarpeta) de InterfaceCliente con el nombre de la carpeta compuesto de ruta relativa** **Si dispositivoDestino == Celular y DispositivoOrigen == PC y elementoAlmacenamiento == Archivo entonces se hace un llamado a obtenerArchivo(String origen) de InterfaceCliente y recibe un arreglo de bytes con el contenido del archivo de nombre origen. Luego hace un llamado a crearArchivo(byte contenido[*], String nombre) de InterfaceRecursosCelular** **Si dispositivoDestino == PC y DispositivoOrigen == PC entonces hace un llamado a copiar(String origen, String destino) de InterfaceCliente** **Si dispositivoDestino == PC y DispositivoOrigen == Celular y**	InterfaceRecursosCelular, M3_0_ResultadoOperacion, InterfaceCliente

elementoAlmacenamieto == Archivo entonces hace un llamado a obtenerArchivo(String origen) de InterfaceRecursosCelular **Y recibe un arreglo de bytes con el contenido del archivo, para luego hacer un llamado a copiarArchivo(byte origen[*], String destino) de InterfaceCliente** **Si dispositivoDestino == PC y DispositivoOrigen == Celular y elementoAlmacenamiento == Carpeta entonces hace un llamado a obtenerListaArchivos(String ruta) de InterfaceRecursosCelular donde recibe una lista de String con las rutas relativas de los archivos que están dentro de la carpeta ruta y sus subcarpetas. Después hace la copia individual de cada archivo en el PC. Luego hace un llamado a obtenerCarpetasVacias(String ruta) de InterfaceRecursosCelular para obtener un listado con las subcarpetas vacías dentro de la carpeta ruta. Para cada subcarpeta vacía se hace un llamado a crearCarpeta(String nuevaCarpeta) de InterfaceCliente** **Finalmente se crea y despliega la ventana M3_0_ResultadoOperacion en el celular, con un mensaje que indica el éxito o fracaso de la operación copiar.**	
public void setRuta(String ruta)	
public void ejecutarComando(String comando) **Hace un llamado a ejecutarComando(String comando) de InterfaceCliente y recibe un String con el resultado de la operación. Luego crea y despliega la ventana M2_3_1_ResultadosComando en el celular**	InterfaceCliente, M2_3_1_ResultadosComando
public void mover(String ruta) **Crea y despliega la ventana M3_3_SeleccionarDispositivoDestino, a partir de lo cual se genera el proceso de copia de un archivo, luego de ello si el archivo o carpeta a mover estaba como origen en el celular entonces se hace el llamado a eliminarOrigen de InterfaceRecursosCelular enviándole la ruta. Si**	M3_3_SeleccionarDispositivoDestino, InterfaceRecursosCelular, InterfaceCliente

el archivo o carpeta a mover estaba como origen en el PC entonces se hace el llamado a eliminarOrigen de InterfaceCliente enviándole la ruta	
public Permiso getNuevoPermiso( )	
public void setNuevoPermiso(Permiso nuevoPermiso)	
public void crearCarpeta(String nuevaCarpeta) **Hace un llamado a crearCarpeta(String ruta, String nuevaCarpeta) de InterfaceCliente con el atributo ruta y el parámetro nuevaCarpeta y recibe un String con el resultado de la creación. Luego crea y despliega la ventana M3_0_ResultadoOperacion**	InterfaceCliente , M3_0_ResultadoOperacion

Tabla 54. CRC ManejadorPrincipalCelular

Clase	**ManejadorPrincipalCelular**
Descripción	Crea y solicita desplegar la ventana principal en el celular.
Módulo	Manejadores\cliente
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public ManejadorPrincipalCelular( )	

Tabla 55. CRC ManejadorRegistroUsuarioCliente

Clase	**ManejadorRegistroUsuarioCliente**
Descripción	Controla las operaciones de modificar y registrar un nuevo usuario en el cliente.
Módulo	Manejadores\cliente
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	

Atributos	private Boolean SuperusuarioValido, RegistroUsuario registroUsuario, RegistroPC registroPC
public ManejadorRegistroUsuarioCliente( )	
public void registrar(String nombrePC, String IP, String puerto, String login, String password) **Crea una instancia de RegistroPC con nombrePC, IP y puerto y la asigna al atributo registroPC.** **Crea una instancia de RegistroUsuario con login, password e IMEI nulo y la asigna al atributo registroUsuario** **Crea y despliega la ventana M0_1_2_2_ValidarSuperusuario en el celular.**	M0_1_2_2_ValidarSuperusuario, InterfaceCliente
public void modificarPassword(String password, String repetirPassword, String passwordActual) **Compara los parámetros nuevoPassword y RepetirNuevoPassword si son iguales, obtiene el login del usuario almacenado en el ManejadorSesiónCliente luego hace llamado a modificarPassword de la InterfaceCliente enviando los parámetros login, Password y NuevoPassword.**	ManejadorSesiónCliente, InterfaceCliente
public Boolean getSuperusuarioValido( )	
public void setSuperusuarioValido(Boolean esValido)	
public void registrar2(String login, String password) **Hace un llamado a validarSuperusuario(RegistroSuperusuario reg) de InterfaceCliente y recibe un Boolean que indica si el Superusuario es válido. En caso de serlo entonces :** *** Hace el llamado a crear(RegistroPC reg) de InterfaceRegistroPCs** *** Finalmente, hace el llamado a crearUsuario(RegistroUsuario reg) de InterfaceCliente**	InterfaceCliente, InterfaceRegistroPCs

Tabla 56. CRC ManejadorSesiónCliente

Clase	**ManejadorSesiónCliente**
Descripción	Controla las operaciones iniciar y cerrar

	sesión en el celular.
Módulo	Manejadores\cliente
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	private String PCDefecto, private RegistroPC PCElegido, private String login
public ManejadorSesiónCliente( )	
public void ingresar(String login, String password) **Hace el llamado a ValidarUsuario de la InterfaceCliente de lo cual obtiene un Boolean que informa de la validez de la información de registro de un usuario, si este es válido e hace llamado a IniciarSesion de la InterfaceCliente, luego se crea y despliega la ventana M2Operaciones**	InterfaceCliente, M2Operaciones
public void iniciarSesionDefecto( ) **Hace llamado a obtenerPCDefecto( ) de InterfaceRegistroPCs y recibe el RegistroPC del PC por defecto. Luego crea y despliega la ventana M1_PantallaDeInicioSesión**	M1_PantallaDeInicioSesión, InterfaceRegistroPCs
public void continuarEliminacion(String login, String password) **Hace un llamado a ValidarUsuario de la InterfaceCliente luego si el usuario es válido entonces hace un llamado a eliminarPC(RegistroPC registroPC) de InterfaceRegistroPCs. Luego crea y despliega la ventana M3_0_ResultadoOperacion en el celular con el resultado de la eliminación.**	InterfaceRegistroPCs, InterfaceCliente, M3_0_ResultadoOperacion
Public void eliminarPC(RegistroPC nombrePC) **Crea y despliega la ventana M1_PantallaDeInicioSesión en el celular**	M1_PantallaDeInicioSesión
public void iniciarSesion( ) **Hace un llamado a consultarPCs( ) de InterfaceRegistroPCs y recibe un arreglo de RegistroPC para luego crear y desplegar la ventana M0_2_SeleccionarPC**	M0_2_SeleccionarPC, InterfaceRegistroPCs

public void cerrarSesión( ) **Hace un llamado a cerrarSesión( ) de InterfaceCliente**	InterfaceCliente
public void setPCElegido(RegistroPC pcElegido)	
public void setPCDefecto(String nombrePC) **Hace llamado a predeterminarPC de ManejadorSesiónCliente Enviándole el nombre del PC que será el por defecto.**	
public String getPCDefecto( )	
public void setPCElegido(RegistroPC PCElegido) **Establece en el ManejadorSesiónCliente el PC elegido por el usuario al cual se desea conectar**	
public String getLogin( )	
public void setLogin(String val)	
public void iniciarSesion( ) **Hace el llamado a consultarPCs de InterfaceRegistroPCs solicitándole un arreglo de objetos de tipo RegistroPC. Luego despliega la ventana M0_2_SeleccionarPC, la cual muestra en una lista la información obtenida anteriormente Crea una nueva instancia de la clase M0_2_SeleccionarPC**	

Tabla 57. CRC ManejadorPrincipalPC

Clase	**ManejadorPrincipalPC**
Descripción	Gestiona la utilización de las diferentes ventanas, la gestión del Superusuario y sus respectivos estados.
Módulo	Manejadores\Servidor
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public ManejadorPrincipalPC( ) **Hace el llamado a existeAlgunSuperusuario del ManejadorRegistroSuperusuario preguntando por la existencia de un Superusuario registrado. Si aún nadie se había registrado antes crea y despliega**	W3_RegistrarSuperusuario, W5_ValidarSuperusuario, ManejadorRegistroSuperusuario

W3_RegistrarSuperusuario. En caso contrario se crea y despliega W5_ValidarSuperusuario	
public void desplegarVentanaPrincipal( ) **Crea y despliega la ventana W0_VentanaPrincipal**	W0_VentanaPrincipal

Tabla 58. CRC ManejadorRegistroSuperusuario

Clase	**ManejadorRegistroSuperusuario**
Descripción	Controla las operaciones de registrar, modificar y validar el Superusuario.
Módulo	Manejadores\Servidor
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public ManejadorRegistroSuperusuario( )	
public void validarSuperusuario(RegistroSuperusuario reg) **Hace el llamado a validarSuperusuario de InterfaceRegistroSuperusuario enviándole un objeto RegistroSuperusuario solicitándole un Boolean que informa si la persona que quiere ingresar es o no el Superusuario. Si es el Superusuario se hace el llamado a desplegarVentanaPrincipal de ManejadorPrincipalPC**	InterfaceRegistroSuperusuario, W0_VentanaPrincipal, ManejadorPrincipalPC
public Boolean existeAlgunSuperusuario( ) **Hace un llamado a existeAlgunSuperusuario( ) de InterfaceRegistroSuperusuario**	InterfaceRegistroSuperusuario
public void registrar(String login, String password) **Crea a partir de los datos login y password un objeto de tipo RegistroSuperusuario, luego hace el llamado a registrar de InterfaceRegistroSuperusuario enviándole dicho objeto. Finalmente hace llamado a desplegarVentanaPrincipal de el manejadorPrincipalPC**	InterfaceRegistroSuperusuario, ManejadorPrincipalPC

public void modificar (String login, String nuevoLogin, String nuevoPassword, String passwordActual) Hace llamado a ValidarSuperusuario de la InterfaceRegistroSuperusuario de lo cual obtiene un Boolean que comprueba los datos login y password del Superusuario, Si estos datos son validos se hace llamado a actualizar de la misma clase.	InterfaceRegistroSuperusuario

Tabla 59. CRC ManejadorRegistroUsuarioServidor

Clase	**ManejadorRegistroUsuarioServidor**
Descripción	Controla las operaciones de modificar y registrar un nuevo usuario en el Servidor.
Módulo	Manejadores\Servidor
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public ManejadorRegistroUsuarioServidor( )	
public void crearUsuario(RegistroUsuario reg) **Hace el llamado a crearUsuario(RegistroUsuario reg) de InterfaceRegistroUsuarios**	InterfaceRegistroSuperusuario
public void gestionarUsuarios( ) **Hace el llamado a consultarUsuarios de InterfaceRegistroUsuario solicitándole un arreglo de objetos de tipo RegistroUsuario. Luego crea y despliega la ventana W6_VentanaGestionarUsuarios, la cual muestra en una lista la información obtenida anteriormente**	W6_VentanaGestionarUsuarios, InterfaceRegistroSuperusuario
public void eliminar (String login) **Hace un llamado a eliminar de la InterfaceRegistroUsuarios**	InterfaceRegistroUsuarios
Public Boolean validarusuario(RegistroUsuario r) **Hace un llamado a validarUsuario de la InterfaceRegistroUsuarios**	InterfaceRegistroUsuarios

Tabla 60. ManejadorServicio

Clase	ManejadorServicio
Descripción	Controla el inicio y la suspensión del servicio en el PC.
Módulo	Manejadores\Servidor
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public ManejadorServicio( )	
public void iniciarServicio( ) **Permite el ingreso de una conexión entrante permitiéndole acceder a los recursos del PC.**	
public void suspenderServicio( ) **Cambia el valor del atributo estadoSesion a "suspendida".**	

Tabla 61. CRC ManejadorSesiónServidor

Clase	ManejadorSesiónServidor
Descripción	Controla las operaciones iniciar y cerrar sesión en el PC.
Módulo	Manejadores\Servidor
Estereotipo	Control
Propiedades	Concreta
Superclases	
Subclases	
Atributos	
public ManejadorSesiónServidor( )	
public void suspenderServicio( ) **Primero busca si hay alguna sesión abierta y entonces la cierra, luego envía una orden al servidor Glassfish y suspende el servicio**	
public void liberarRecursos( ) Elimina las conexiones existentes entre una sesión y los recursos del PC	

public void cerrarSesión( ) Cierra la sesión actual	
public void iniciarSesion( ) **Establece la conexión de un nuevo Usuario con el servicio entregado por el servidor**	

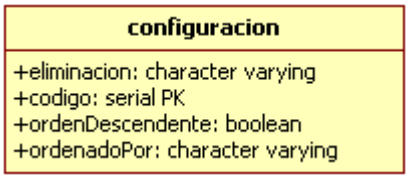
9. TABLAS DE LA BASE DE DATOS

Debido a la naturaleza del proyecto no se requirieron complejas configuraciones de bases de datos, por ello solo se requirieron las siguientes tablas:

9.1. TABLAS DEL SERVIDOR

- Tabla configuración: Almacena la configuración de historial, la columna eliminación es la encargada de almacenar la frecuencia de eliminación del historial, la cual puede tener los valores diario, semanal, mensual o no eliminar. ordenDescendente indica si la tabla estará ordenada descendentemente o no, además ordenadoPor indica por cual columna se ordenará si por Operación, ejecución, Usuario o Fecha, de la tabla Historial.

Figura 191. Tabla del Servidor Configuración



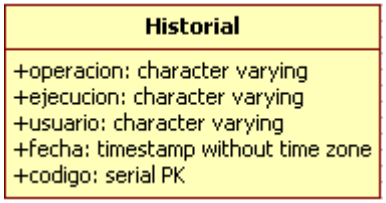
La siguiente es la instrucción SQL para la creación de la tabla configuración.

```
CREATE TABLE configuracion
(
 eliminacion character varying,
 "ordenDescendente" boolean,
 "ordenadoPor" character varying,
 codigo serial NOT NULL,
```

```
 CONSTRAINT configuracion_pkey PRIMARY KEY (codigo)
)
WITHOUT OIDS;
ALTER TABLE configuracion OWNER TO "admin";
```

- Tabla Historial: Almacena el historial de las operaciones hechas por los usuarios del servicio ; la columna operación es una descripción de la solicitud hecha, ejecución indica si el método se ejecuto correctamente o lanzo excepción , usuario indica el nombre de la cuenta de usuario desde donde se realizo la petición, fecha es el momento en que se hizo la solicitud.

Figura 192. Tabla del Servidor historial



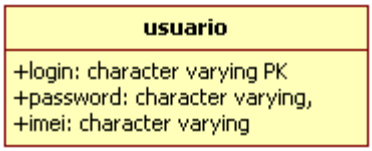
La siguiente es la instrucción SQL para la creación de la tabla Historial.

```
CREATE TABLE historial
(
 operacion character varying,
 ejecucion character varying,
 usuario character varying,
 fecha timestamp without time zone,
 codigo serial NOT NULL,
 CONSTRAINT historial_pkey PRIMARY KEY (codigo)
)
WITHOUT OIDS;
```

ALTER TABLE historial OWNER TO "admin";

- Tabla usuario: Almacena las cuentas de usuario que pueden acceder al servicio de ese PC. Consta de las columnas nombre, password e IMEI.

Figura 193. Tabla del Servidor Usuario



La siguiente es la instrucción SQL para la creación de la tabla usuario.

```
CREATE TABLE usuario
(
 "login" character varying NOT NULL,
 "password" character varying,
 imei character varying,
 CONSTRAINT usuario_pkey PRIMARY KEY ("login")
)
WITHOUT OIDS;
ALTER TABLE usuario OWNER TO "admin";
```

9.2. *TABLAS DEL CLIENTE*[5]

[5] Información obtenida de *JAVA A TOPE: J2ME (JAVA 2 MICRO EDITION). EDICIÓN ELECTRÓNICA*

La tecnología J2ME provee un mecanismo a los MIDlets que le permite almacenar datos de forma persistente para su futura recuperación. Este mecanismo está implementado sobre una base de datos basada en registros que se le ha llamado Record Management System o RMS(Sistema de gestión de Registros).

Esta información será guardada en el dispositivo en una zona de memoria dedicada para este propósito. La cantidad de memoria y la zona asignada para ello dependerán de cada dispositivo.

Los *MIDlets* son los encargados de crear los *Record Stores* para comunicarse con ellos.

Un *Record Store* tal como su nombre indica es un almacén de registros. Estos registros son la unidad básica de información que utiliza la clase RecordStore para almacenar datos.

Cada uno de estos registros está formado por dos unidades:

Un número identificador de registro (*Record* ID) que es un valor entero que realiza la función de clave primaria en la base de datos.

Un arreglo de bytes que es utilizado para almacenar la información deseada.

Tabla 62. Representación grafica de un RecordStore

RecordStore	
Record ID	**Datos**
1	byte [] arrayDatos
2	byte [] arrayDatos
……	……

Los principales métodos de esta clase son:

openRecordStore – Abre el almacén de registros

closeRecordStore – Cierra el almacén de registros

deleteRecordStore – Borra el almacén de registros

getName – Recupera el nombre del almacén de registros

getNumRecords – Recuperar el número de registros del almacén

addRecord – Añade un registro al almacén de registros

getRecord – Recupera un registro del almacén de registros

deleteRecord – Borra un registro del almacén de registros

enumerateRecord – Obtiene un *enumeration* del almacén de registros

Como es visto cada registro de este tipo de tablas se debe almacenar en forma de bytes.

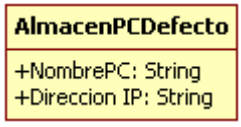
Para el cliente solo se crearon dos Record Store (Almacenes de registros)

RecordStore RegistroPC: Este almacena el nombre de un PC servidor y su dirección IP.

Figura 194. Tabla del Cliente RegistroPC

RecordStoreAlmacenPCDefecto: Este almacena el PC por defecto y solo posee un PC a la vez pues gracia a él se recupera quien es el actual predeterminado.

Figura 195. Tabla del Cliente AlmacenPCDefecto



ANEXO B.

1. XML (EXTENSIBLE MARKUP LANGUAGE)

Lenguaje de etiquetado extensible, es un simple y muy flexible formato de texto derivado de SGML (ISO 8879). Originalmente diseñado para satisfacer los retos de publicaciones electrónicas de gran escala, XML está teniendo un papel cada vez más importante en el intercambio de una gran cantidad de datos en la red.

El lenguaje de etiquetado extensible es un conjunto de reglas para definir etiquetas semánticas que dividen un documento en partes e identifica las diferentes partes del documento. Es un metalenguaje de etiquetado que define una sintaxis en el cual se pueden escribir otros lenguajes de etiquetado.

XML no es otro lenguaje de etiquetado como HTML, TeX o troff. Estos lenguajes no son extensibles, por lo tanto definen un conjunto fijo de etiquetas que describen un número fijo de elementos y solo se pueden usar las etiquetas definidas en el lenguaje.

XML es un lenguaje que permite hacer etiquetas personalizadas. Estas etiquetas deben estar organizadas de acuerdo a ciertos principios generales, pero son muy flexibles en su representación.

1.1. VENTAJAS

XML es ideal para documentos largos y complejos porque los datos están estructurados. Es posible especificar un vocabulario que defina los elementos en el documento y puede especificar las relaciones entre elementos.

Permite desarrollar lenguajes de etiquetado para dominios específicos (por ejemplo, música, química, recursos humanos) sin necesidad de soporte especial o complicados plugins por parte de los fabricantes del navegador

Los datos son auto descriptivos y son fáciles de leer y escribir por programas de computador y personas.

Los datos son razonablemente resistentes a la corrupción de datos, si se pierden algunos bytes o incluso largas secuencias de bytes, esto no corrompe notablemente el texto restante, en contraste con otros formatos como datos comprimidos u objetos serializados de Java en el cual la corrupción de uno o menos de un solo byte puede hacer el resto del archivo irrecuperable.

XML también provee un mecanismo del lado del cliente que integra datos de múltiples fuentes y ser mostrados como un solo documento. Los datos pueden ser reorganizados al instante. Las partes del documento pueden ser mostradas u ocultadas dependiendo de las acciones del usuario. Esto puede ser muy útil con grandes repositorios de información como bases de datos relacionales.

XML no tiene propietario y es fácil de leer y escribir, es un formato excelente para el intercambio de datos entre diferentes aplicaciones. XML no tiene limitaciones con copyright, patentes, secreto de fabricación o algún otro tipo de restricciones de propiedad intelectual.

1.2. ESPACIOS DE NOMBRES

Los espacios de nombres permiten eliminar las ambigüedades y solucionar los problemas de homonimia que se producen en los documentos, ya que en un mismo documento existen palabras con el mismo nombre (ejemplo "capital"), pero con diferentes significados y espacios semánticos (término geográfico/término económico-financiero)

Tabla 63 Comparación de dos documentos XML con problemas de homonimia

Capital	**Capital**
<país nombre="España"> **<capital>Madrid</capital>** **</país>**	<inversión> <capital>2000€</capital> </inversión>

Ante la ambigüedad, surgió la pregunta de cómo combinar en un mismo documento varios vocabularios.

<país nombre="España">

<capital>Madrid</capital>

<capital>2000€</capital>

</país>

Una posible solución era asignar un nombre único a cada etiqueta creando una autoridad mundial que asignara dicho nombre, pero una solución mucho más fácil era utilizar un mecanismo ya existente, por eso se pensó en los URIs. Un URI es un identificador global único. Aquí no se trata de utilizar un URI como enlace, ni tiene por qué tener contenido, los URI sólo se utilizan para que el nombre sea único. Ejemplo: **http://www.hipertexto.info**

Cuando los espacios de nombre no estaban predefinidos un conflicto de nombre ocurría cuando dos diferentes documentos usan los mismos nombres para sus elementos

Documento XML con información de frutas

```
<table>
  <tr>
  <td>Apples</td>
  <td>Bananas</td>
  </tr>
</table>
```

Documento XML con información de muebles

```
<table>
  <name>African Coffee Table</name>
  <width> 80 </width>
  <length> 120 </length>
</table>
```

Si estos dos documentos XML fueran adicionados juntos habría un conflicto de nombres porque ambos elementos contienen un elemento <table> con diferente contenido y definición

✓ Resolviendo conflictos usando prefijos

```
<h:table>
  <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Ahora no habrá conflicto porque los dos documentos usan un diferente nombre para el elemento table

(<h:table> and <f:table>).

Al usar un prefijo se han creado dos tipos de elementos <table>

✓ Resolviendo conflictos usando espacios de nombres (NameSpaces)

```
<h:table xmlns: h = "http://www.w3.org/TR/html4/" >
  <h:tr>
  <h:td> Apples </h:td>
  <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns: f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

En lugar de usar los prefijos se ha adicionado el atributo xmlns a <table> para dar al prefijo un nombre calificado asociado con un espacio de nombre

Un espacio de nombre es identificado por medio de un URI.

Semántica para la declaración de espacios de nombres

- ✓ **Declaración de espacios de nombres**

[1]	NSAttName	::=	PrefixedAttName | DefaultAttName
[2]	PrefixedAttName	::=	'xmlns:' NCName
[3]	DefaultAttName	::=	'xmlns'
[4]	NCName	::=	NCNameStartChar NCNameChar*
[5]	NCNameChar	::=	NameChar - ':'
[6]	NCNameStartChar	::=	Letter | '_'

En este caso NCName sería un prefijo usado para asociar los elementos y atributos con el espacio de nombres, alcanzando solo aquellos elementos que se encuentran dentro del alcance del elemento al cual se le añadió el espacio de nombres

- ✓ **Nombres calificados**

[7]	QName	::=	PrefixedName | UnprefixedName
[8]	PrefixedName	::=	Prefix ':' LocalPart
[9]	UnprefixedName	::=	LocalPart
[10]	Prefix	::=	NCName

[11] LocalPart ::= NCName

Los Tags en un documento XML deben tener un formato como el que sigue

[12] STag ::= '<' QName (S Attribute)* S? '>' [NSC: Prefix Declared]

[13] ETag ::= '</' QName S? '>' [NSC: Prefix Declared]

[14] EmptyElemTag ::= '<' QName (S Attribute)* S? '/>' [NSC: Prefix Declared]

Donde los atributos se definen también en función de nombres calificados:

[15] Attribute ::= NSAttName Eq AttValue

| QName Eq AttValue [NSC: Prefix Declared]

Un ejemplo es el siguiente:

```
<edi:price xmlns:edi='http://ecommerce.example.org/schema'
units='Euro'>32.18</edi:price>
```

Ejemplo con atributos

```
<x xmlns:edi='http://ecommerce.example.org/schema'>
<!—El atributo ‘taxClass’ corresponde al espacio de nombres
http://ecommerce.example.org/schema -->
        <lineItem edi:taxClass="exempt">Baby food</lineItem>
</x>
```

El ámbito de un espacio de nombres se extiende desde el inicio del primer tag hasta su finalización excluyendo el ámbito de su propia declaración.

El ámbito de un tag vacio es el mismo.

Múltiples espacios de nombres pueden ser declarados en un mismo tag.

```
<?xml version="1.0"?>

<!-- Comentarios -->

<bk : book xmlns : bk= 'urn:loc.gov:books'  xmlns:isbn='urn:ISBN:0-395-36341-6'>

        <bk:title>Cheaper by the Dozen</bk:title>

<isbn:number>1568491379</isbn:number>


</bk:book>
```

- ✓ **Espacios de nombre por defecto**

Un espacio de nombre por defecto aplica a todos los elementos que no están prefijados dentro del ámbito de un espacio de nombres. Los espacios de nombres por defecto no afectan a los atributos.

- ✓ **Unicidad de atributos**

En documentos XML un tag no puede contener dos atributos con:

nombres idénticos

nombres calificados con la misma parte local y con prefijo el cual haya sido enlazado a un espacio de nombres que sea idéntico.

ANEXO C.

1. XML SCHEMA[6]

XML Schema es un lenguaje de esquema escrito en XML, basado en la gramática y pensado para proporcionar una mayor potencia expresiva que la DTD, más limitadas en la descripción de los documentos a nivel formal. Su extensión usual es .xsd. Gracias al XML Schema es posible definir la estructura de un documento XML.

Los Schemas se construyen a partir de diferentes tipos de componentes:

- Elemento (*element*)
- Atributo (*attribute*)
- Tipo simple (*simple type*)
- Tipo complejo (*complex type*)
- Entre otros

Un XML Schema define las siguientes restricciones:

- Si un elemento puede o no ser vacio
- Valores por defecto para elementos y atributos
- Los hijos de un elemento
- El número de hijos de un elemento
- El orden de los hijos
- Atributos que pueden aparecer en un documento

[6] http://www.w3schools.com/Schema/default.asp

- Tipos de datos y atributos

Los XML Schemas son los sucesores de los DTDs porque:

XML Schema son extensibles para futuras adiciones

- Son escritos en XML
- Soportan tipos de datos
- Soportan espacios de nombres (NameSpaces)
- Debido a que soporta tipos de datos:
- Es más fácil describir el contenido de los documentos admisibles
- Es más fácil validar correctamente un dato
- Es más fácil trabajar con datos de una base de datos
- Es más fácil definir restricciones sobre los datos
- Es más fácil definir patrones en los datos (Formatos)
- Es más fácil convertir datos entre diferentes tipos de datos

Este es un documento escrito en XML:

```
<?xml version="1.0"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Una posible Schema para el anterior documento puede ser:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
 <xs:complexType>
   <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

1.1. ENCABEZADO DE UN DOCUMENTO SCHEMA

La raíz de un documento XMLSchema es <schema>

```
<?xml version="1.0"?>

<xs:schema>

</xs:schema>
```

Se definen para esto los siguientes atributos

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

</xs:schema>
```

El siguiente atributo: xmlns:xs="http://www.w3.org/2001/XMLSchema" Indica que los elementos y los tipos de datos usados en el Schema vienen del espacio de nombres "http://www.w3.org/2001/XMLSchema".

El atributo targetNamespace="http://www.w3schools.com" indica que el elemento definido por el Schema (note, to , from, heading, body) vienen de el espacio de nombres "http://www.w3schools.com"

El fragmento: xmlns="http://www.w3schools.com" indica que el espacio de nombres por defecto será "http://www.w3schools.com".

El atributo **elementFormDefault** sirve para exigir que al usar el esquema desde un documento los elementos se asocien con el targetNamespace del esquema. Es posible también cambiar en el Schema los valores por defecto para todas las declaraciones de elementos locales a través del atributo *elementFormDefault*, como se muestra:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://ejemplo.org/empleado/"
  elementFormDefault="qualified">
  …
</xsd:schema>
```

Ahora, por defecto, todos los elementos locales deberán ser calificados en la instancia como en esta instancia válida:

```
<tns:empleado xmlns:tns="http://ejemplo.org/empleado/">
  <tns:nombre>Monica</tns:nombre>
  <tns:fechaingreso>1997-12-02</tns:fechaingreso>
  <tns:salario>42000.00</tns:salario>
</tns:empleado>
```

Ya que en este caso todos los elementos están calificados, podríamos escoger usar una declaración de espacio de nombres por defecto, y la instancia se mantendría válida:

```
<empleado xmlns:tns="http://ejemplo.org/empleado/">
```

<nombre>Monica<nombre>

<fechaingreso>1997-12-02<fechaingreso>

<salario>42000.00<salario>

</Empleado>

1.2. *REFERENCIA A UN SCHEMA EN UN DOCUMENTO XML*

<?xml version="1.0"?>

<note xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">

<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>

</note>

El atributo xmlns="http://www.w3schools.com" especifica el espacio de nombres por defecto.

El atributo xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" define que se ha instanciado un documento de tipo Schema.

Una vez se ha instanciado se puede usar el atributo schemaLocation con 2 valores. El primer valor es el espacio de nombres a usar. El segundo valor es la localización de documento Schema de ese documento

xsi:schemaLocation="http://www.w3schools.com note.xsd"

1.3. *ELEMENTOS SIMPLES XSD*

Es un elemento que solo contiene texto. Aunque el texto puede ser de diferentes tipos de datos (boolean, String, date, etc) o puede ser de un tipo definido por el usuario del documento.

Sintaxis de un elemento simple <xs:element name="xxx" type="yyy"/>

Los más comunes tipos de datos son

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Valores por defecto

```
<xs: element name="color" type="xs:string" default="red"/>
```

Ejemplo:

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Definición Schema

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

1.4. ATRIBUTOS

Un elemento simple no puede tener un atributo. Si un elemento tiene atributos se considera ser un elemento complejo pero el atributo mismo se declara como un elemento simple.

Sintaxis

<xs:attribute name="xxx" type="yyy"/>

xxx es el nombre del atributo yyy el tipo de dato del atributo

ejemplo

<lastname lang="EN">Smith</lastname>

<xs:attribute name="lang" type="xs:string"/>

Valor por defecto

<xs:attribute name="lang" type="xs:string" default="EN"/>

Valor fijo y no puede cambiarse

<xs:attribute name="lang" type="xs:string" fixed="EN"/>

Atributos requeridos y opcionales

<xs:attribute name="lang" type="xs:string" use="required"/>

1.5. RESTRICCIONES PARA TIPOS DE DATOS (FACETS)

Se utiliza para definir restricciones mediante la palabra restriction.

<xs:restriction base="xs:string">

…

</xs:restriction>

Restricción	Descripción
Enumeration	Define una lista de valores aceptables
fractionDigits	Especifica el máximo número de lugares decimales permitidos. Debe ser igual o más grande de cero.
Length	Especifica el número exacto de caracteres o lista de items permitidos. Debe ser

	igual o más grande que cero.
maxLength	Especifica el número máximo de caracteres o lista de items permitidos. Debe ser igual o más grande que cero.
minExclusive	Especifica el número mínimo de caracteres o lista de items permitidos. Debe ser igual o más grande que cero.
Pattern	Define la exacta secuencia de los caracteres permitidos
totalDigits	Especifica el número de exacto de dígitos permitidos.
whiteSpace	Especifica cómo se mostrarán los espacios en blanco, si se mostraran o no.

Ejemplo:

```
<xs:element name="letter">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:pattern value="([a-z][A-Z])+"/>
                        <xs:whiteSpace value="preserve"/>
                        <xs:minLength value="5"/>
                        <xs:maxLength value="8"/>
                        <xs:length value="8"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
```

1.6. TIPOS COMPLEJOS

Un elemento complejo puede tener otros elementos y atributos.

Hay diversos tipos de elementos complejos:

- Elementos vacios
- Elementos que contienen solo otros elementos
- Elementos que solo contienen texto
- Elementos que contienen texto y otros elementos.

Ejemplo

```
<employee>
        <firstname>John</firstname>
        <lastname>Smith</lastname>
</employee>
```

El anterior tipo complejo puede representarse en Shema así:

```
<xs:element name="employee">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="firstname" type="xs:string"/>
                        <xs:element name="lastname" type="xs:string"/> </xs:sequence>
        </xs:complexType>
</xs:element>
```

ANEXO C.

1. SIMPLE OBJECT ACCESS PROTOCOL (SOAP) 1.1

SOAP es un protocolo liviano para el intercambio de información estructurada en un entorno distribuido. Es un protocolo basado en XML que consiste de tres partes: un "envelope" que define una estructura para describir que es un mensaje y como procesarlo, un conjunto de reglas de codificación para expresar instancias de tipos de datos personalizados, y una convención para la representación de llamadas a procedimientos remotos y respuestas. SOAP puede ser usado con varios protocolos, sin embargo la especificación solo describe el uso de SOAP en combinación con HTTP y HTTP Extensión Framework.

El protocolo SOAP consiste de tres partes. La primera es el envelope, usado para describir el contenido del mensaje y algunas claves sobre como procesarlo. La segunda consiste de las reglas para codificar tipos de datos personalizados. Esta es una de las más importantes partes de SOAP: su extensibilidad. La última parte describe la aplicación del envelope y las reglas de codificación para representar llamadas y respuestas RPC, incluyendo el uso de HTTP como protocolo de transporte.

El uso más común de SOAP hoy es hacer llamadas RPC en computación distribuida. Cualquier protocolo de transporte puede usarse para enviar mensajes SOAP, aunque en la especificación SOAP solo se describe HTTP, se discute también su uso con protocolos como SMTP, ftp, BEEP, JXTA, y otros.

1.1. CONVENCIONES DE NOTACIÓN

Los prefijos de espacios de nombres "SOAP-ENV" y "SOAP-ENC" usados en este documento están asociados con los espacios de nombres SOAP
http://schemas.xmlsoap.org/soap/envelope/ y
"http://schemas.xmlsoap.org/soap/encoding/" respectivamente.

A lo largo de este documento, el prefijo de espacio de nombres “xis” está asociado con el URI “http://www.w3.org/1999/XMLSchema-instance”. Igualmente, el prefijo de espacio de nombres

“xsd” está asociado con el URI “http://www.w3.org/1999/XMLSchema”. El prefijo de espacio de nombres “tns” es usado para indicar el espacio de nombres “target namespace” del documento actual. Todos los demás prefijos son solo ejemplos.

1.2. RELACIÓN CON XML

Todos los mensajes SOAP son codificados usando XML. Una aplicación SOAP debería incluir el espacio de nombres propio sobre todos los elementos y atributos definidos en los mensajes SOAP que esta genere. Una aplicación SOAP debe ser capaz de procesar los espacios de nombres SOAP en los mensajes que recibe.

SOAP define 2 espacios de nombres:

- El SOAP envelope tiene el identificador de espacio de nombres http://schemas.xmlsoap.org/soap/envelope/
- El SOAP encoding tiene el identificador de espacio de nombres "http://schemas.xmlsoap.org/soap/encoding/"

Un mensaje SOAP no debe contener un DTD (Document Type Declaration) y tampoco debe tener instrucciones de procesamiento.

SOAP usa el atributo local y no calificado “id” de tipo “ID” para especificar el único identificador de un elemento codificado. SOAP usa el atributo local y no calificado “href” de tipo “uri-reference” para especificar una referencia a ese valor, de acuerdo a la especificación XML, XML Schema Specification, and XML Linking Language Specification.

1.3. SOAP ENVELOPE

Un mensaje SOAP es un documento XML que consiste de un elemento envelope obligatorio. El envelope está conformado por una cabecera opcional, y un cuerpo obligatorio. El espacio de nombres identificador de los elementos y atributos definidos en esta sección es "http://schemas.xmlsoap.org/soap/envelope/". Un mensaje SOAP contiene lo siguiente:

- El Envelope es el elemento superior del documento XML y representa el mensaje SOAP.

- La cabecera es un mecanismo genérico para adicionar características a un mensaje SOAP en una forma descentralizada sin previo acuerdo entre las partes involucradas en la comunicación.
- El cuerpo es un contenedor de información obligatoria esperada por el destino final del mensaje.

Las reglas gramaticales son las siguientes:

- ✓ **Envelope**
  - El nombre del elemento es "Envelope"
  - El elemento debe estar presente en un mensaje SOAP
  - El elemento puede contener declaraciones de espacio de nombres o atributos adicionales. Si están presentes, estos atributos deben estar en un espacio de nombres calificado. De igual forma, el elemento puede contener sub-elementos adicionales. Si están presentes, estos subelementos deben estar en un espacio de nombres calificado y deben seguir al elemento SOAP Body.

- ✓ **Cabecera**
  - El nombre del elemento es "Header".
  - El elemento puede estar presente en un mensaje SOAP. Si está presente, el elemento debe ser el primer hijo inmediato del elemento SOAP Envelope.
  - El elemento puede tener un conjunto de entradas de cabecera, cada una es hijo inmediato del elemento "Header". Todos los elementos hijo inmediatos del elemento SOAP Header debe estar en un espacio de nombres calificado. Las entradas de cabecera se usan para codificar elementos para procesamiento de transacciones, autenticación, u otra información relacionada con el procesamiento o enrutamiento del mensaje.

- ✓ **Cuerpo**
  - El nombre del elemento es "Body".

- El elemento debe estar presente en un mensaje SOAP y debe ser un hijo inmediato del elemento SOAP Envelope. El elemento Body debe seguir directamente al elemento SOAP Header, si está presente. De lo contrario, el cuerpo debe ser el primer elemento hijo inmediato del elemento SOAP Envelope.
- El elemento puede contener un conjunto de entradas, cada una hija inmediata del elemento SOAP Body. Los elementos hijos inmediatos del elemento SOAP Body pueden estar en un espacio de nombres calificado. SOAP define el elemento Fault, el cual es usado para indicar mensajes de error.

1.4. ATRIBUTO SOAP ENCODINGSTYLE

El atributo global SOAP encodingStyle es usado para indicar las reglas de serialización usadas en un mensaje SOAP. Este atributo puede aparecer en cualquier elemento, y tiene alcance sobre el elemento que lo contiene y todos los elementos hijos que no tengan este atributo. No hay una codificación por defecto para un mensaje SOAP.

El valor del atributo está en una lista ordenada de una o más URIs identificando las reglas de serialización que pueden ser usadas para des-serializar el mensaje SOAP indicado en el orden del más específico al menos específico. Ejemplos de valores son:

"http://schemas.xmlsoap.org/soap/encoding/"
"http://my.host/encoding/restricted http://my.host/encoding/"

Las reglas de serialización definidas por SOAP están identificadas por el URI "http://schemas.xmlsoap.org/soap/encoding/". Los mensajes usando esta serialización particular deberían indicarlo usando el atributo SOAP encodingStyle. En adición, todas las URIs sintácticamente empiezan con http://schemas.xmlsoap.org/soap/encoding/.

Un valor de longitud cero del URI (“”) indica explícitamente que no se hacen reclamos por el encoding estile de los elementos contenidos. Esto puede ser usado para desactivar cualquier reclamo de elementos contenidos.

El atributo encodingStyle está en un espacio de nombres calificado usando el identificador SOAP-ENV. En el siguiente ejemplo se especifica el atributo encodingStyle como parte del elemento Envelope:

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

...

...

</SOAP-ENV:Envelope>

1.5. CABECERA SOAP (SOAP HEADER)

SOAP provee un mecanismo para extender un mensaje en una forma descentralizada y modular sin previo conocimiento entre las partes en comunicación. Ejemplos típicos de extensiones que pueden ser implementados como entradas de cabecera son autenticación, administración de transacciones, payment etc.

El elemento Header está codificado como el primer hijo inmediato del elemento SOAP Envelope. Todos los hijos inmediatos del elemento Header son llamados entradas de cabecera.

Las reglas de codificación para entradas de cabecera son las siguientes:

- Una entrada de cabecera se identifica por su nombre completo de elemento calificado, el cual consiste del espacio de nombres URI y el nombre local. Todos los hijos inmediatos del elemento SOAP Header deben estar en un espacio de nombres calificado.
- El atributo SOAP EncodingStyle puede ser usado para indicar el estilo de codificación usado por las entradas de cabecera.
- El atributo SOAP mustUnderstand y el atributo SOAP actor se pueden usar para indicar como procesar las entradas y por quién.

Aquí hay un ejemplo de una cabecera SOAP que contiene un hijo inmediato llamado username, el cual es un elemento que identifica el usuario que está haciendo la petición.

<SOAP-ENV:Header>

<ns1:username xmlns:ns1=”MiAplicacion”>

Carolina

</ns1:username>

</SOAP-ENV:Header>

1.6. USO DE ATRIBUTOS DE CABECERA

Los atributos de cabecera definidos en esta sección determinan cómo un destino de un mensaje SOAP debería procesar el mensaje. Una aplicación SOAP al generar un mensaje SOAP debería usar los atributos de cabecera solo sobre hijos inmediatos del elemento SOAP Header. El destino de un mensaje SOAP debe ignorar todos los atributos de la cabecera que no pertenezcan a un elemento hijo inmediato del elemento SOAP Header.

✓ El atributo SOAP actor

Un mensaje SOAP viaja desde el origen hasta el destino final, y en su camino posiblemente pasa a través de un conjunto de intermediarios SOAP. Un intermediario SOAP es una aplicación que es capaz de recibir y reenviar mensajes SOAP. Tanto los intermediarios como el destino final están identificados por una URI.

Es probable que no todas las partes de un mensaje SOAP estén dirigidas al destino final, y pueden estar dirigidas a uno o más intermediarios en la ruta del mensaje. El rol de un destino de un elemento cabecera es similar al de aceptar un contrato en que no puede ser extendido más allá de ese destino. Así, cuando un destino recibe un elemento cabecera no debe reenviarlo a la siguiente aplicación en la ruta del mensaje SOAP. Cada intermediario puede insertar un elemento cabecera similar pero en ese caso, el contrato es entre esa aplicación y el destino del nuevo elemento de cabecera.

El atributo global actor se puede usar para indicar el receptor de un elemento cabecera. El valor del atributo actor es una URI. La URI especial "http://schemas.xmlsoap.org/soap/actor/next" indica que el elemento cabecera está dirigido a la primera aplicación SOAP que procese el mensaje.

Si un atributo actor no aparece sobre un elemento de cabecera, se asume que el elemento está dirigido al destino final del mensaje. En esencia, esto es equivalente a incluir el atributo actor con el URI del destino final.

En el siguiente ejemplo el mensaje es enviado a un servidor intermedio localizado en Http://www.mindstrm.com/AppServer. Se quiere que la aplicación servidor registre el nombre del usuario que hizo la petición, y entonces pase la petición al servidor destino final. Para hacer esto, se le da el valor http://www.mindstrm.com//AppServer al atributo actor del elemento username:

<SOAP-ENV:Header>

<ns1:username xmlns:ns1=”JavaSoapBook” SOAP-ENV:actor=http://www.mindstrm.com/AppServer>

Jessica

</ns1:username>

</SOAP-ENV:Header>

El intermediario remueve el elemento username de la cabecera antes de pasar el mensaje al destino final. El atributo actor está en un espacio de nombres calificado por el identificador SOAP-ENV. Esto es porque el atributo actor está definido por SOAP y está asociado al espacio de nombres http://schemas.xmlsoap.org/soap/envelope.

✓ El atributo SOAP mustUnderstand

El atributo global mustUnderstand puede ser usado para indicar si el procesamiento de una entrada de cabecera es obligatorio u opcional. El destino de una entrada de cabecera está definido por el atributo SOAP actor. El valor del atributo mustUnderstand puede ser “1” o “0”. La ausencia del atributo SOAP mustUnderstand es semánticamente equivalente a su presencia con el valor “0”.

Si un elemento cabecera es etiquetado con un atributo SOAP mustUnderstand con valor de “1”, el destino de esa entrada de cabecera debe obedecer a la semántica (como está definida por el nombre completo calificado del elemento) y procesarlo correctamente de acuerdo a dicha semántica. Si el receptor de este mensaje no entiende el elemento username, se requiere que responda con una falla SOAP.

Esto puede ser útil por ejemplo, si el emisor está actualizado con una nueva versión y la nueva versión usa alguna información nueva que tiene que ser procesada por el servidor para un resultado eficaz. Una versión antigua del servidor no entiende los nuevos elementos de cabecera SOAP que provienen de una aplicación cliente recientemente actualizada.

Modificando el ejemplo anterior para indicar que el receptor debe entender el elemento username:

<SOAP-ENV:Header>

<ns1:username xmlns:ns1="JavaSoapBook" SOAP-ENV:actor=http://www.mindstrm.com/AppServer SOAP-ENV:mustUnderstand="1">

Jessica

</ns1:username>

</SOAP-ENV:Header>

1.7. EL CUERPO SOAP (SOAP BODY)

El elemento SOAP Body provee un mecanismo simple para intercambiar información obligatoria dirigida al destino final del mensaje. Típicamente el uso del elemento Body incluye llamadas RPC y reporte de errores.

El elemento Body está codificado como un elemento hijo inmediato del elemento SOAP Envelope. Si un elemento cabecera está presente entonces el elemento Body debe seguir inmediatamente al elemento cabecera, de lo contrario el cuerpo debe ser el primer elemento hijo inmediato del elemento Envelope.

Todos los hijos inmediatos del elemento Body son llamados entradas y cada entrada del cuerpo está codificada como un elemento independiente dentro del elemento SOAP Body.

Las reglas de codificación para las entradas del cuerpo son las siguientes:

- Una entrada del cuerpo está identificada por su nombre calificado completo, lo cual consiste del espacio de nombres URI y el nombre local. Los hijos inmediatos del elemento SOAP Body pueden estar en un espacio de nombres calificado.

- El atributo SOAP encodingStyle puede ser usado para indicar el estilo de codificación usado para las entradas del cuerpo.

SOAP define una entrada de cuerpo, la cual es la entrada Fault usada para reportar errores.

1.8. SOAP FAULT

El elemento Fault es el único elemento definido por SOAP para el cuerpo y se usa para llevar información de error hacia el origen de un mensaje SOAP. El elemento Fault debe aparecer como un subelemento inmediato del elemento Body, y no puede aparecer más de una vez.

SOAP define cuatro subelementos del elemento Fault. No todos son requeridos, y algunos son apropiados solo bajo ciertas condiciones.

✓ El elemento faultcode

El elemento faultcode provee un indicio de la falla que es identificable por un proceso software, proporcionando una oportunidad para el sistema que recibe la falla para actuar apropiadamente. El código no es necesariamente útil para humanos – ese es el propósito del elemento faultstring. El elemento faultcode es obligatorio, y debe aparecer como un subelemento del elemento Fault. SOAP define un número de códigos de falla para uso en el elemento faultcode. Estos códigos de falla están asociados con el espacio de nombres http://schemas.xmlsoap.org/soap/envelope. Aquí hay un resumen de las descripciones de los códigos de falla definidos por SOAP:

- *VersionMismatch*

Indica que un espacio de nombres inválido fue asociado con el SOAP Envelope

- *MustUnderstand*

Significa que hubo un elemento de cabecera que contenía un elemento mustUnderstand con un valor de 1, y el atributo no fue entendido o la semántica asociada con el atributo no podría ser seguida.

- *Client*

Indica que hubo algún error en el formato del mensaje SOAP o que el mensaje no contiene la información apropiada o necesaria. El cliente debería asumir que el mensaje no es adecuado para ser reenviado sin hacer los cambios apropiados.

- *Server*

Indica que el mensaje no pudo ser procesado debido a razones ajenas al formato o el contenido del mensaje recibido. Esto se puede interpretar para decir que el mensaje podría ser reenviado sin modificaciones y posiblemente ser procesado después. Por ejemplo, un servidor de bases de datos necesario para completar la acción requerida puede estar inactivo, pero puede estar activo después.

Estos códigos de falla son extensibles; esto significa que pueden ser extendidos usando una notación con punto, donde el nombre sigue cada punto sirve para proveer detalle más específico. Por ejemplo: Server.WeatherStationFailure.

Mientras esto se puede considerar como un error de servidor, provee información más específica indicando que hubo algún problema con la estación del Tiempo, equipo necesario para llenar la petición.

✓ El elemento faultstring

Este elemento también es obligatorio, y debe aparecer como subelemento del elemento Fault. Su propósito es suministrar una descripción entendible por el ser humano de la falla; no está diseñada para ser procesada por él.

El elemento SOAP Fault se usa para transportar información de error y/o estado dentro de un mensaje SOAP. Si está presente, el elemento SOAP Fault debe aparecer como una entrada de cuerpo y no debe aparecer más de una vez en el elemento Body.

✓ El elemento Faultactor

El elemento faultactor está dirigido a proveer información acerca de quién causó la falla en la ruta del mensaje. Esto es similar al atributo SOAP actor pero en lugar de indicar el destino de la entrada cabecera, indica la fuente de la falla. El valor del atributo faultactor es una URI identificando la fuente. Las aplicaciones que no sean el destino final del mensaje SOAP deben incluir el elemento faultactor en un elemento SOAP Fault. El último destino de un mensaje puede usar el elemento faultactor para indicar explícitamente que él generó la falla.

✓ El elemento Detail

El elemento detail está destinado a transportar la información específica del error relacionada con el elemento Body. Debe estar presente, si el contenido del elemento Body no pudo ser procesado exitosamente. No debe ser usado para llevar información de los errores pertenecientes a las entradas de cabecera. La información detallada de error perteneciente a las entradas de cabecera debe ser transportada dentro de las entradas de cabecera.

La ausencia del elemento detail en el elemento Fault indica que las fallas no están relacionadas con el procesamiento del elemento Body. Esto puede ser usado para distinguir si el elemento Body fue procesado o no en caso de falla.

Todos los elementos hijos inmediatos del elemento detail son llamados entradas detail y cada entrada detail está codificada como un elemento independiente dentro del elemento detail.

Las reglas de codificación de las entradas de detalle son las siguientes:

- Una entrada detail está identificada por su nombre completo calificado, el cual consiste del espacio de nombres URI y el nombre local. Los elementos hijos inmediatos del elemento detail pueden estar en un espacio de nombres calificado.
- El atributo SOAP encodingStyle puede ser usado para indicar el estilo de codificación usado por las entradas de detalle.

Otros subelementos Fault pueden estar presentes, ellos pertenecen a un espacio de nombres calificado.

1.9. CÓDIGOS DE FALLA SOAP

Los valores de faultcode definidos en esta sección deben ser usados en el elemento faultcode cuando describan las fallas descritas en la especificación SOAP. El identificador de espacio de nombres para estos valores de códigos de falla es "http://schemas.xmlsoap.org/soap/envelope/". El uso de este espacio es recomendado (pero no requerido) en la especificación de métodos definidos fuera de la especificación SOAP.

El valor por defecto de faultcode se define en una forma extensible que permite para nuevos valores de códigos de falla SOAP, ser definidos mientras mantienen compatibilidad hacia atrás con valores existentes de códigos de falla. El mecanismo usado es muy similar al 1xx, 2xx, 3xx, etc. estado básico de las clases definidas en HTTP. Sin embargo, en lugar de enteros, ellos están definidos como nombres calificados. El carácter punto "." es usado como separador de valores de códigos de falla indicando que a la izquierda del punto está el valor código de falla más genérico que el valor a la derecha. Ejemplo: Client.Authentication.

El conjunto de códigos de falla definido en la especificación SOAP es:

Tabla 64 Códigos de falla definidos en SOAP

Nombre	Significado
VersionMismatch	El mensaje estaba en un espacio de nombres inválido para el elemento SOAP Envelope
MustUnderstand	Un elemento hijo inmediato del elemento SOAP Header con un atributo mustUnderstand de valor de "1"que no fue interpretada u obedecida por la parte que la procesó.
Client	El tipo de errores de la clase cliente indica que el mensaje no fue bien redactado o contiene información incorrecta. Por ejemplo, la ausencia de información de autenticación. Generalmente indica que el mensaje no puede ser reenviado sin hacerle cambios.
Server	El tipo de errores Servidor indican que el mensaje no puede ser procesado por razones ajenas al contenido del mensaje mismo. Por ejemplo, el procesamiento podría incluir comunicarse con un servidor, el cual no responde. El mensaje puede tener éxito después.

ANEXO D.

1. WEB SERVICES DESCRIPTION LANGUAGE (WSDL) 1.1

1.1. RESUMEN

WSDL es un formato XML para la descripción de servicios Web como un conjunto de operaciones de punto final operando sobre mensajes que contienen documentos o procedimientos orientados a la información. Las operaciones y los mensajes son descritos abstractamente, y enlazados a un protocolo concreto de red y el formato de los mensajes define un punto final (endpoint). Los puntos finales relatados son combinados dentro de puntos finales abstractos (Servicios). WSDL es extensible para permitir la descripción de puntos finales independientemente de los formatos de mensajes o protocolos de red usados para comunicarse.

1.2. INTRODUCCIÓN

Como los protocolos de comunicación y formatos de mensajes están estandarizados en la comunidad web, cada vez es más necesario e importante describir las comunicaciones de una manera más estructurada. WSDL direcciona esta necesidad para definir gramáticas XML que describan servicios web como colecciones de puntos finales de comunicación capaces de intercambiar mensajes.

La definición de servicios con WSDL proporcionar documentación para sistemas distribuidos y sirven como fórmula para automatizar los detalles involucrados en las aplicaciones de comunicación.

Un documento WSDL define servicios como colecciones de puntos finales de red, o puertos (port). En WSDL, la definición abstracta de puntos finales y de mensajes se separa de su concreta red de despliegue o formato de datos. Esto permite la reutilización de definiciones abstractas: mensajes (message), que son descripciones abstractas de los datos que se

intercambiarán, y tipos de puertos (port type) que son colecciones abstractas de las operaciones (operation). El protocolo

concreto y la especificación del formato de datos para un tipo de puerto constituye un elemento reutilizable (binding). Un puerto se define asociando

a cada dirección de red un binding, y una colección de puertos definen un servicio. Por lo tanto, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

Types : Un contenedor de datos de definiciones de tipo que usa algún tipo de sistema (como XSD).

Message: Un resumen, definición escrita de los datos que se comuniquen.

Operation: Un resumen de la descripción de una acción soportada por el servicio.

Port Type: Un resumen del conjunto de operaciones de apoyo de uno o varios puntos finales.

Binding: Un concreto protocolo y especificación de formato de datos de especificación de formato para un tipo de puerto(Port Type).

Port: Un único punto final definido como una combinación de un binding y una dirección de red.

Service: Una colección de puntos finales relacionados.

Es importante observar que WSDL no introduce una nueva definición de tipo de lenguaje. WSDL reconoce la necesidad de que los sistemas deben ser muy expresivos para describir formatos de mensajes, y apoya la especificación de esquemas XML (XSD) como su sistema de tipo canónico. Sin embargo, dado que no es razonable esperar que un solo tipo de gramática sea usado para describir todos los formatos de mensajes presentes y futuros, WSDL permite el uso extensible de otros lenguajes.

Además, WSDL define un mecanismo binding. Este es usado para adjuntar un protocolo específico o formato de datos o la estructura de un conjunto de mensajes, operaciones, o puntos finales. Permitiendo la reutilización de definiciones abstractas.
Además de la definición de servicios esenciales, esta especificación introduce extensiones al binding para los siguientes protocolos y formatos de mensajes:

SOAP 1.1

HTTP GET / POST

MIME

1.3. DEFINICIÓN DEL SERVICIO

Esta sección describe los elementos núcleo del lenguaje WSDL.

✓ Estructura de un documento WSDL

Un documento WSDL es simplemente un conjunto de definiciones (**definitions**).Hay un elemento **definitions** en la raíz, con definiciones dentro. La gramática es la siguiente:

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>

  <import namespace="uri" location="uri"/>*

  <wsdl:documentation .... /> ?

  <wsdl:types> ?
    <wsdl:documentation .... />?
    <xsd:schema .... />*
    <-- extensibility element --> *
  </wsdl:types>

  <wsdl:message name="nmtoken"> *
    <wsdl:documentation .... />?
    <part name="nmtoken" element="qname"? type="qname"?/> *
  </wsdl:message>

  <wsdl:portType name="nmtoken">*
    <wsdl:documentation .... />?
```

```
  <wsdl:operation name="nmtoken">*
    <wsdl:documentation .... /> ?
    <wsdl:input name="nmtoken"? message="qname">?
      <wsdl:documentation .... /> ?
    </wsdl:input>
    <wsdl:output name="nmtoken"? message="qname">?
      <wsdl:documentation .... /> ?
    </wsdl:output>
    <wsdl:fault name="nmtoken" message="qname"> *
      <wsdl:documentation .... /> ?
    </wsdl:fault>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="nmtoken" type="qname">*
  <wsdl:documentation .... />?
  <-- extensibility element --> *
  <wsdl:operation name="nmtoken">*
    <wsdl:documentation .... /> ?
    <-- extensibility element --> *
    <wsdl:input> ?
      <wsdl:documentation .... /> ?
      <-- extensibility element -->
    </wsdl:input>
    <wsdl:output> ?
```

```
        <wsdl:documentation .... /> ?
        <-- extensibility element --> *
      </wsdl:output>
      <wsdl:fault name="nmtoken"> *
        <wsdl:documentation .... /> ?
        <-- extensibility element --> *
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="nmtoken"> *
    <wsdl:documentation .... />?
    <wsdl:port name="nmtoken" binding="qname"> *
      <wsdl:documentation .... /> ?
      <-- extensibility element -->
    </wsdl:port>
    <-- extensibility element -->
  </wsdl:service>

  <-- extensibility element --> *

</wsdl:definitions>
```

Los servicios son definidos usando los seis elementos:

types , el cual provee definiciones de tipos de datos usados para describir el mensaje intercambiado.

message, el cual representa una definición abstracta de los datos a ser transmitidos.

portType, el cual es un conjunto de operaciones abstractas. Cada operación se refiere a un mensaje de entrada y mensajes de salida**.**

binding, el cual especifica un protocolo concreto y especificación del formato de datos para las operaciones y mensajes definidos por un particular portType.

port, el cual especifica una dirección para un binding, y así definir un único punto final de comunicación.

service, el cual es usado para agregar un conjunto de puertos relacionados.

✓ Types

El elemento **types** rodea la definición de los tipos de datos que son relevantes para el intercambio de mensajes. Para una máxima interoperabilidad y neutralidad de la plataforma, WSDL prefiere el uso de XSD como un canónico tipo de sistema, y lo trata como un tipo del sistema intrínseco.

```
<definitions .... >
    <types>
        <xsd:schema .... />*
    </types>
</definitions>
```

El sistema de tipos XSD puede ser usado para definir los tipos en el mensaje independiente de que el formato de transmisión sea XML o si el resultante Schema valida el formato particular de transmisión.

✓ Mensaje

Los mensajes consisten de uno o más partes lógicas. Cada parte está asociada con un tipo de algún tipo de sistema usando un atributo del tipo de mensaje. El conjunto de atributos de tipo de mensaje es extensible. WSDL define como tal muchos tipos de mensaje para el uso con XSD.

element: se refiere a un elemento XSD usando un QName

type. Se refiere a un tipo simple o un tipo complejo de XSD usando un QName

Otros atributos de tipo de mensaje pueden ser definidos siempre y cuando ellos usen un espacio de nombres diferente al de WSDL.

La sintaxis para un mensaje es el siguiente. Los atributos del tipo del mensaje son mostrados en negrita.

<definitions .... >

  <message name="nmtoken"> *

    <part name="nmtoken" **element="qname"? type="qname"?**/> *

  </message>

</definitions>

El atributo **name** provee un nombre único entre todos los mensajes dentro del documento WSDL.

✓ Port Types

Un port type es un nombrado conjunto de operaciones y mensajes abstractos involucrados.

<wsdl:definitions .... >

  <wsdl:portType name="nmtoken">

    <wsdl:operation name="nmtoken" .... /> *

  </wsdl:portType>

</wsdl: definitions>

El atributo **name** del port type provee un nombre único entre todos los port types definidos dentro del documento WSDL.

Una operación es nombrada por medio de atributo **name.**

WSDL tiene cuatro primitivas transiciones que un punto final puede soportar:

One-way. El punto final recibe un mensaje

Request-response. El punto final recibe un mensaje, y envía un mensaje de acuse

Solicit-response. El punto final envía un mensaje, y recibe un acuse de recibo.

Notification. El punto final envía un mensaje

WSDL hace referencia a estas operaciones como primitivas. Aunque la petición/respuesta o solicitud/respuesta pueden ser modelada abstracta utilizando dos mensajes en un solo sentido, es útil modelar estas primitivas de operación debido a que:

Son muy comunes.

La secuencia puede ser descrita sin tener que introducir más flujo complejo de información.

Algunos puntos finales sólo pueden recibir mensajes si ellos son el resultado de una petición de respuesta sincrónica.

Un simple algoritmo de flujo puede ser derivado de estas primitivas en el momento en que el flujo definición se desea.

Aunque petición/respuesta o solicitud/respuesta se correlacionan lógicamente en el documento WSDL, un determinado binding describe la correlación concreta de la información. Por ejemplo, los mensajes de petición y de respuesta pueden ser cambiados como parte de una o dos comunicaciones de la red real.

Aunque la base de la estructura WSDL apoya la vinculación de estas cuatro primitivas de transmisión, WSDL sólo define el compromiso de las primitivas one-way y Request-response. Se espera que las especificaciones que definen los protocolos para **Solicit-response** o **notification** también incluyeran extensiones WSDL para el binding que permitan el uso de estas primitivas.
Las operaciones se refirieren a los mensajes que participan usando el atributo **message** de tipo QName.

- ✓ **Bindings**

Un binding define el formato del mensaje y los detalles del protocolo para las operaciones y mensajes definidos por un portType. Puede haber cualquier número de bindings para un determinado portType. La gramática de un binding es la siguiente:

<wsdl:definitions .... >

<wsdl:binding name="nmtoken" type="qname"> *

```
    <-- extensibility element (1) --> *
    <wsdl:operation name="nmtoken"> *
      <-- extensibility element (2) --> *
      <wsdl:input name="nmtoken"? > ?
        <-- extensibility element (3) -->
      </wsdl:input>
      <wsdl:output name="nmtoken"? > ?
        <-- extensibility element (4) --> *
      </wsdl:output>
      <wsdl:fault name="nmtoken"> *          <-- extensibility element (5) --> *
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

El atributo name proporciona un nombre único entre todos los binding definidos dentro de WSDL en el que remite el documento.
Un binding referencia a un portType usando el atributo type de tipo **Qname**.

La extensibilidad de los elementos en un binding se utiliza para especificar la gramática concreta para la entrada (3), la salida (4), y los mensajes de error (5). – la información antes de la operación (2), así como la información antes del binding (1) también se puede especificar.

Una operación dentro de un elemento binding especifica información vinculante para el funcionamiento con el mismo nombre dentro de la unión del portType.

Para unificar el nombre de una operación se utiliza el espacio de nombres wsdl:input y wsdl:output

Un binding debe especificar exactamente un protocolo.

Un binding no debe especificar información de direccionamiento.

✓ Puertos (Ports)

Un puerto define el punto final de un individuo mediante la especificación de una dirección única para un binding.

```
<wsdl:definitions .... >
  <wsdl:service .... > *
    <wsdl:port name="nmtoken" binding="qname"> *
      <-- extensibility element (1) -->
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

El atributo name proporciona un nombre único entre todos los puertos definidos dentro de documento WSDL.

La atributo binding(de tipo QName) hace referencia a la utilización de reglas de enlace definidas por WSDL.

(1) se utiliza para especificar la información de dirección para el puerto.

Un puerto no deberá especificar más de una dirección.
Un puerto no deberá especificar cualquier otra información que no sea la información de las direcciones.

✓ Servicios

Un grupo de servicios es un conjunto de los puertos juntos:

```
<wsdl:definitions .... >
  <wsdl:service name="nmtoken"> *
    <wsdl:port .... />*
  </wsdl:service>
</wsdl:definitions>
```

El atributo name proporciona un nombre único entre todos los servicios que se definen dentro de un documento WSDL.

Los puertos dentro de un servicio tienen la siguiente relación:

Ninguno de los puertos se comunican entre sí (por ejemplo la salida de un puerto no es la entrada de otro).

Si un servicio tiene varios puertos que comparten un portype, pero emplean distintos binding o direcciones, los puertos son alternativos. Cada puerto ofrece semánticamente un comportamiento equivalente (dentro del transporte y las imposiciones impuestas por cada binding). Esto permite a un consumidor de un documento WSDL poder elegir puerto (s) para comunicarse con la base de algunos criterios (protocolo, distancia, etc.)

Mediante el examen de sus puertos, podemos determinar el portType de un servicio. Esto permite a un consumidor de un documento WSDL determinar si desea comunicarse a un determinado servicio basado o no compatible con varios tipos de puerto. Esto es útil si existe alguna relación implícita entre las operaciones de los puertos de los tipos, y que todo el conjunto de puertos de los tipos debe estar presente con el fin de realizar una tarea concreta.

ANEXO E.

1. JSR 172 - JAVA SPECIFICATION REQUEST 172

Este capítulo está basado en la especificación de servicios Web para Java 2 Micro Edition JSR 172.

1.1. OBJETIVO GENERAL

Proveer dos capacidades nuevas a la plataforma J2ME:

- ✓ Acceso a servicios web remotos basados en SOAP/XML
- ✓ Interpretación de datos XML

Hay un gran interés y movimiento en la comunidad java en el uso de estándares de servicios web e infraestructuras para proveer el modelo de programación para la siguiente generación de servicios empresariales. Hay mucho interés en la comunidad de desarrolladores en extender los servicios empresariales a clientes J2ME.

Objetivos principales

Las principales entregas de la especificación JSR 172 son dos nuevos e independientes paquetes opcionales:

Un paquete opcional adicionando soporte para interpretar XML a la plataforma.

Los datos estructurados enviados a dispositivos móviles desde aplicaciones existentes, serán probablemente en formato XML. Para evitar incluir código para procesar esta información en cada aplicación, es deseable definir un paquete opcional que pueda ser incluido con la plataforma.

Crear un paquete opcional para facilitar el acceso a servicios web basados en XML desde los perfiles CDC Y CLDC.

Este paquete opcional definirá un API que permita a los dispositivos móviles acceder a servicios web basados en XML.

Objetivos Secundarios de la interpretación XML

El paquete opcional de interpretación XML tiene los siguientes objetivos adicionales:

Subconjunto estricto de la funcionalidad JSR63 JAXP 1.2 donde sea posible.

Conocer los requerimientos de tamaño de la plataforma.

Asegurarse que la API se ajusta a los requerimientos de espacio de los dispositivos objetivo.

Conocer los requerimientos de desempeño de la plataforma

Asegurar que la API puede ser implementada dentro de los requerimientos de memoria y procesamiento para los dispositivos objetivo.

Objetivos Secundarios de los Servicios Web

El paquete opcional de servicios web tiene los siguientes objetivos adicionales:

Funcionalidad de un subconjunto de JAX-RPC 1.1

Separarlo del paquete distribuible opcional de interpretación XML

El paquete opcional de servicios web no debe depender del paquete opcional de interpretación XML. Debe ser posible la entrega del paquete de servicios web de manera independiente al paquete de interpretación XML.

Proveer acceso a servicios web desde J2ME sin capacidades de servidor

Esta JSR no define puntos finales para los dispositivos objetivo. Esta funcionalidad puede ser adicionada en una versión futura de la especificación.

Conocer los requerimientos de recursos

Asegurar que la especificación se ajusta a los requerimientos de espacio de los dispositivos finales.

Conocer los requerimientos de desempeño de la plataforma

Asegurarse que la API puede ser implementada dentro de los requerimientos de memoria y procesamiento de los dispositivos objetivo.

1.2. SUBCONJUNTO JAXP

✓ Objetivo General

El objetivo de este paquete opcional es definir un subconjunto estricto donde sea posible la funcionalidad de interpretación XML definida en JSR-063 JAXP 1.2 que puede ser usada en la plataforma Java 2 Micro Edition (J2ME).

✓ ¿Por qué XML sobre J2ME?

XML se está convirtiendo en un estándar para la interacción de clientes con servidores, sus bases de datos y servicios relacionados. Con su neutralidad a la plataforma y un fuerte soporte de la industria, XML está empezando a ser usado por los desarrolladores para conectar clientes con información remota empresarial. Un creciente número de estos clientes está basado en la plataforma J2ME, con una amplia selección de dispositivos móviles, PDAs, y otros dispositivos portables. Debido a que muchos desarrolladores utilizan estos dispositivos móviles más para acceder a información remota empresarial, el soporte XML sobre la plataforma J2ME se está volviendo una necesidad.

✓ Requerimientos de la Plataforma

Para suministrar implementaciones que sean útiles para el más amplio rango de configuraciones y perfiles, la especificación trata la CLDC (Conected Limited Device Configuration) como plataforma mínima. El objetivo en tamaño para una implementación completa, incluyendo interfaces sin comprimir es 35 Kb.

✓ Requerimientos de la API

Este paquete opcional es el subconjunto de la especificación JAXP 1.2. Las siguientes modificaciones definen la funcionalidad:

- Una implementación no debe proveer algún soporte para las interfaces de la interpretación SAX 1.0 (Simple Api for XML).

- SAX 1.0 ha sido reemplazado por SAX 2.0 el cual está incluido.
- Una implementación no debe proveer algún soporte para DOM 1.0 o 2.0 (Document Object Model)
- DOM es considerado demasiado pesado en términos del tamaño de la implementación y el espacio de memoria de ejecución necesario, para ser usado sobre la plataforma J2ME.
- Una implementación no provee algún soporte para XSLT
- Una implementación debe soportar SAX 2.0
- Una implementación debe proveer soporte para espacios de nombres XML.
- Una implementación debe proveer soporte para codificaciones UTF-8 y UTF-16.

✓ Soporte para Intérpretes con Validación

Una implementación puede soportar validación de documentos contra DTDs. La validación XML es un proceso costoso en términos de potencia de procesamiento y uso de memoria y probablemente no sea soportado por la mayoría de dispositivos J2ME. Sin embargo, si la plataforma tiene la habilidad de soportarlo, puede proveerse un intérprete con validación (debido a la naturaleza limitada de la mayoría de los dispositivos J2ME, se espera solo un intérprete sea soportado, pero es válido soportar dos).

Si una implementación suministra un intérprete XML con validación, ese intérprete debe cumplir con la especificación XML 1.0.

✓ Soporte de Intérpretes Sin Validación

La mayoría de plataformas J2ME probablemente tendrán un intérprete sin validación XML, porque consume menos recursos que su contraparte con validación.

Si una implementación provee un intérprete sin validación este debe cumplir con la especificación XML 1.0 lo cual solo requiere que la implementación interprete un subconjunto interno del DTD

✓ Requisitos de Conformidad

Para lograr un entorno predecible para aplicaciones, implementaciones de la especificación se deben conocer ciertos requisitos de conformidad.

Las implementaciones de la especificación deben estar conforme a la recomendación XML 1.0. Además, las implementaciones deben estar conforme a la recomendación de los espacios de nombres XML tanto como las interfaces SAX 2.0.

✓ Subconjunto de APIs JAXP

Son tres los paquetes que comprenden el subconjunto de la API JAXP:

- javax.xml.parsers
- org.xml.sax
- org.xml.sax.helpers

Cuando se observa este conjunto, se puede notar que mucho de lo que existe en el API J2SE JAXP no está en el conjunto del API J2ME JAXP. Los requerimientos de tamaño para la plataforma J2ME son estrictos, permitiendo solo aproximadamente 35 Kb para una implementación completa de JAXP. Sin embargo, aunque muchas de estas clases no están, gran parte de su funcionalidad permanece.

❖ *javax.xml.parsers*

El paquete javax.xml.parsers contiene las clases para obtener y referenciar la implementación de un intérprete dado. El paquete contiene cuatro clases:

- SAXParser
- SAXParserFactory
- FactoryConfigurationError
- ParserConfigurationException

❖ *org.xml.sax*

El paquete org.xml.sax contiene un subconjunto de las clases e interfaces del API SAX 2.0. Las interfaces incluidas en este paquete son:

- Attributes
- Locator

Las clases incluidas en este paquete son:

- SAXException

- SAXNotRecognizedException
- SAXNotSupportedException
- SAXParseException

❖ *org.xml.sax.helpers*

El paquete org.xml.sax.helpers contiene una clase para aplicaciones que extiende y recibe eventos de interpretación:

- DefaultHandler

1.3. VISIÓN GENERAL DEL SUBCONJUNTO JAX-RPC

JAX-RPC es una API de Java para interactuar con servicios web basados en SOAP. Esta especificación define un subconjunto de la especificación JAX-RPC 1.1 que es apropiado para la plataforma J2ME.

La funcionalidad proporcionada en el subconjunto refleja las limitaciones de la plataforma: cantidad de memoria y poder de procesamiento, tanto como las limitaciones del entorno de despliegue; bajo ancho de banda y alta latencia.

✓ Visión General del subconjunto de Funcionalidad

Las siguientes secciones proveen una visión general de la funcionalidad del subconjunto JAX-RPC 1.1.

❖ *Invocación Basada en Stub*

Las implementaciones deben soportar invocación basada en stub local (también conocido como stubs estáticos) por la interface Stub. No hay soporte en el subconjunto para proxy dinámicos o interface de invocación dinámica (DII). El soporte para proxy dinámicos y DII será evaluado para incluirlo en futuras versiones de esta especificación.

❖ *Modo de Operación*

Las implementaciones deben generar stubs que usen el estilo de documento y el uso del literal (document/literal).

❖ *Tipos de Datos*

Las implementaciones deben soportar documentos WSDL que referencien los siguientes tipos de datos:

- boolean
- byte
- short
- int
- long
- float
- String
- complex types
- arreglos de tipos primitivos y complejos

❖ *Fallas SOAP*

Una falla SOAP debe estar mapeada a una excepción Java específica del servicio o a RemoteException. El SOAPFaultException no está incluido en el API subconjunto**.**

❖ *Modelo de Proveedor de Servicios*

No hay soporte para el modelo de proveedor de servicios. El subconjunto solo provee soporte para clientes que acceden a servicios de proveedores de servicios. Dispositivos tan limitados, actualmente no pueden actuar como proveedores de servicios. El subconjunto JAX-RPC no incluye el paquete javax.xml.rpc.server**.**

❖ *Mapeo Extensible de Tipos*

No hay soporte para mapeo extensible de tipos. El subconjunto no incluye el paquete javax.xml.rpc.encoding.

❖ *Servicios En Tiempo de Ejecución JAX-RPC*

Las implementaciones deben soportar autenticación básica HTTP y administración de sesiones como se definió en la especificación JAX-RPC 1.1, Autenticación Básica HTTP y administración de sesiones respectivamente.

✓ Notas específicas de la plataforma j2me

Las siguientes clases están incluidas en el paquete opcional de Servicios Web J2ME para satisfacer dependencias de JAX-RPC sobre las plataformas basadas en CLDC:

- java.rmi.Remote
- java.rmi.RemoteException
- java.rmi.MarshalException
- java.rmi.ServerException

Un paquete opcional RMI está disponible para plataformas basadas en CDC y si el paquete opcional está presente las versiones de java.rmi.Remote, java.rmi.RemoteException, java.rmi.MarshalException y java.rmi.ServerException incluidas en el paquete opcional RMI deben ser usadas.

1.4. REQUERIMIENTOS DEL SUBCONJUNTO JAX-RPC

Los siguientes requerimientos están basados en los requerimientos para JAX-RPC y tienen que ser actualizados para reflejar el subconjunto de funcionalidad JAX-RPC definida por su especificación. Los requerimientos adicionales, específicos a este subconjunto JAX-RPC, son también mencionados.

✓ Recursos

Una implementación del subconjunto JAX-RPC definido por la especificación tiene los siguientes requerimientos mínimos:

50 Kb de RAM

25 Kb de ROM

Estos requerimientos son sin contar con aquellos definidos para la plataforma base.

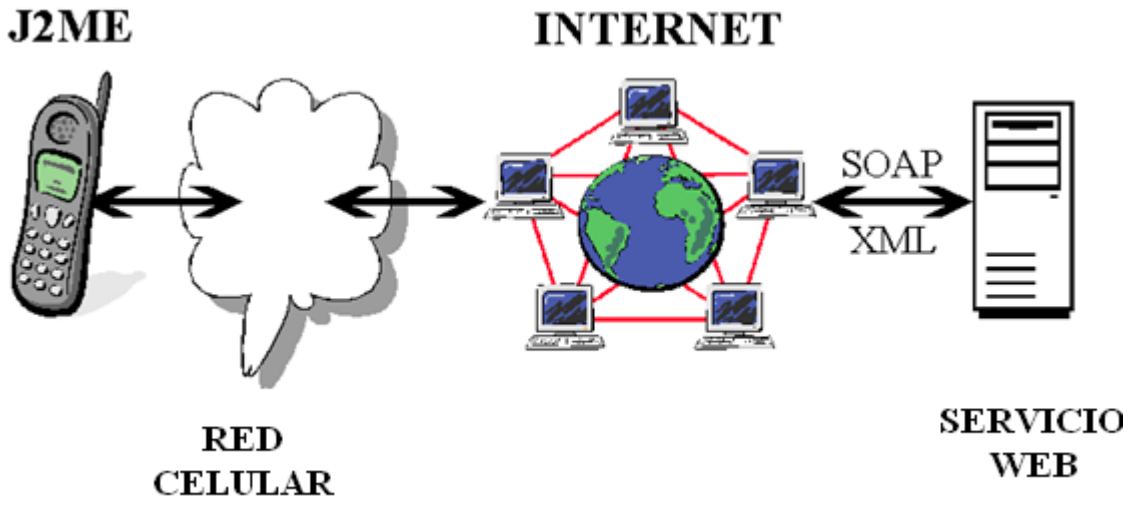
✓ Alineamiento con el perfil básico WS-I

El perfil básico WS-I (WS-I BP) y su conjunto contenedor - JAX-RPC 1.1 proveen recomendaciones y aclaraciones para muchas descripciones referenciadas por esta especificación. Para proveer interoperabilidad con otras implementaciones de servicios web, la implementación del subconjunto JAX-RPC deben seguir las recomendaciones del WS-I BP donde coincida con la funcionalidad definida en esta especificación.

✓ Protocolo de Codificación

Para proveer interoperabilidad con la infraestructura existente de servicios web, las implementaciones del subconjunto JAX-RPC deben comunicarse con los proveedores de servicios web usando mensajes codificados con SOAP 1.1 usando un protocolo basado en XML. En la siguiente imagen este requerimiento corresponde al flujo de mensajes al proveedor del servicio web a través de internet/intranet.

Figura 196. Arquitectura de una red



Debido a las limitaciones de la red celular (capacidad de procesamiento de los dispositivos, latencia, limitaciones de ancho de banda de la red) no hay requerimientos para que la ejecución del subconjunto JAX-RPC sobre el cliente soporte codificación XML.

Si una implementación de la JAX-RPC runtime no produce y consume mensajes codificados basados en el protocolo XML esto debe ser transparente al servicio web. En tales casos los mensajes a enviar por la red inalámbrica deben ser transformados a mensajes XML codificados en SOAP 1.1 antes de ser enviados al servicio web.

A la inversa, los mensajes entrantes codificados con SOAP 1.1 (la respuesta desde el proveedor del servicio) deben ser transformados para ser entendidos por el dispositivo.

✓ Protocolo Bindings

Un objetivo del subconjunto de la especificación JAX-RPC es permitir el soporte para múltiples protocolos de enlace que están basados en el conjunto de información XML (InfoSet). Por ejemplo, los mensajes SOAP 1.2 son especificados como XML Infosets. JAX-RPC permite soporte para protocolo binario de enlace que está basado en el infoset XML pero no lleva documentos XML 1.0. Observe que el uso del término "protocolo basado en XML" en este capítulo es consistente con este objetivo.

Basado en este objetivo, las APIs núcleo de JAX-RPC (definidas en el paquete javax.xml.rpc) están definidas para ser independientes de cualquier protocolo de enlace.

Una implementación interoperable del subconjunto de JAX-RPC debe soportar el protocolo SOAP 1.1. La implementación de subconjunto JAX-RPC runtime no requiere producir y consumir documentos basados en la codificación XML 1.0. Como consecuencia, una implementación del subconjunto JAX-RPC runtime que no produce documentos XML 1.0 debe usar una representación que pueda ser transformada en mensajes interoperables SOAP 1.1.

Nota – El subconjunto JAX-RPC definido en la especificación considerará soportar el protocolo SOAP 1.2 cuando sea soportado por una futura versión de la especificación JAX-RPC 1.1.

✓ Transporte

Una implementación del sistema runtime del subconjunto JAX-RPC debe soportar HTTP 1.1 como el transporte para mensajes SOAP. El HTTP binding de los mensajes SOAP está basado en la especificación SOAP 1.1.

Note que se requiere soporte para HTTP 1.1, esto no significa que HTTP sea el único protocolo de transporte que puede ser soportado por la implementación del sistema runtime JAX-RPC. El núcleo de APIs de JAX-RPC está diseñado para ser neutral al transporte. Esto permite a las APIS JAX-RPC ser útiles con cualquier transporte con la capacidad para entregar mensajes SOAP y tiene definido un protocolo binding para el protocolo SOAP 1.1.

Las características de seguridad disponibles en la implementación JAX-RPC dependerán de las características de seguridad suministradas en la configuración o perfil J2ME usado por esa implementación. Por ejemplo el MIDP 2.0 security framework.

La especificación JAX-RPC no impide el uso de SOAP binding con un transporte que soporte mecanismos de seguridad. Sin embargo, la especificación de SOAP bindings para transportes que soporten seguridad está fuera del alcance de la especificación JAX-RPC.

Una implementación del sistema runtime del subconjunto JAX-RPC puede soportar HTTP/S como el protocolo subyacente de seguridad.

Una implementación del sistema runtime del subconjunto JAX-RPC debe cumplir con los requerimientos de la sección "Usando SOAP en HTTP" de la especificación WS-I BP, donde estos requerimientos coincidan con el subconjunto de la especificación JAX-RPC.

✓ Sistema de Tipos Soportado

Las implementaciones del subconjunto JAX-RPC deben tener soporte para los siguientes mapeos de tipos XML:

✓ Codificación XML para Mensajes SOAP

Las implementaciones del subconjunto de la especificación JAX-RPC deben soportar la representación literal de un mensaje SOAP representando una llamada RPC o una respuesta.

La codificación SOAP 1.1 no debe ser soportada.

Una implementación del sistema runtime del subconjunto JAX-RPC debe adicionalmente cumplir los requerimientos hechos en la sección "representación XML de mensajes SOAP" de la WS-I BP, donde estos requerimientos coincidan con el subconjunto de la especificación JAX-RPC.

✓ Sistema Runtime de JAX-RPC

El sistema runtime JAX-RPC forma el núcleo de una implementación JAX-RPC. El sistema runtime del subconjunto JAX-RPC es una librería del lado del cliente que provee un conjunto de servicios requeridos para los mecanismos de ejecución del subconjunto JAX-RPC.

El subconjunto de la especificación JAX-RPC solo define el cliente J2ME y no define algo en el lado del servidor.

El subconjunto de la especificación JAX-RPC requiere una implementación JAX-RPC compatible del lado del cliente debe estar basado sobre la plataforma J2ME (CLDC 1.0) Connection Limited Device Configuration 1.0 o Connected Device Configuration 1.0 (CDC 1.0).

El subconjunto de APIs JAX-RPC define la interface de programación al sistema runtime del subconjunto JAX-RPC. Consultar el capítulo "APIs núcleo del subconjunto JAX-RPC" para la especificación del subconjunto de APIs JAX-RPC.

✓ Mapeo de Tipos por Defecto

El subconjunto de la especificación JAX-RPC especifica los siguientes mapeos estándares de tipos:

Tipos de datos XML a tipos de Java

Una implementación del sistema runtime del subconjunto JAX-RPC debe soportar este mapeo de tipos estándar.

✓ Mapeo de Tipos extensible

El subconjunto de la especificación JAX-RPC no provee suporte para mapeo de tipos extensible.

✓ Modelo del Proveedor de Servicios

La especificación del subconjunto de JAX-RPC no especifica un modelo estándar de programación para un servicio corriendo sobre la plataforma J2ME.

✓ Descripción del Servicio

Una implementación del sistema runtime del subconjunto JAX-RPC debe cumplir adicionalmente con los requerimientos de la sección “Descripción del Servicio” de la WS-I BP, donde estos requerimientos coincidan con el subconjunto de la especificación JAX-RPC.

✓ Registro del Servicio y Descubrimiento

Ver el requerimiento R010, Registro del Servicio y Descubrimiento, de la especificación JAX-RPC 1.1

✓ Java API for XML Binding (JAXB)

Ver el requerimiento R011, Java API for XML Binding (JAXB), de la especificación JAX-RPC.

✓ Modos de Interacción del Nivel de Aplicación

Las implementaciones del subconjunto JAX-RPC deben soportar el siguiente modo de interacción entre un cliente y un proveedor de servicio. Note que este modo de interacción es visible como parte del modelo de programación de JAX-RPC y es llamado modo de interacción del nivel de aplicación.

La especificación JAX-RPC no está direccionada al como la implementación del sistema runtime de JAX-RPC provee soporte para estos modos de interacción del nivel de aplicación. Un sistema runtime JAX-RPC puede usar una implementación más primitiva de modos de interacción específicos para implementar el soporte al modo de interacción de nivel de aplicación.

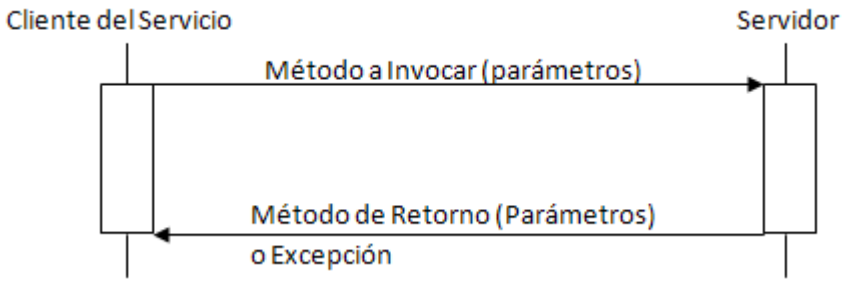
La especificación JAX-RPC requiere que cualquier mecanismo específico de implementación o modos de interacción de nivel de aplicación no sea expuesto al modelo de programación JAX-RPC.

La especificación JAX-RPC no define ningún tipo de calidad del servicio QoS (por ejemplo: garantías de la entrega del mensaje, mensajería confiable, uso de intermediarios) relacionados a los modos de interacción de nivel de aplicación. Observe que la especificación JAX-RPC no impide la implementación específica del soporte de QoS.

- *Modelo Petición-Respuesta Síncrono*

Un cliente de un servicio invoca un método remoto sobre el proveedor del servicio objetivo y recibe un valor de retorno o una excepción. El hilo de invocación del cliente se bloquea mientras la invocación del método remoto es procesada por el proveedor del servicio. Eventualmente, el cliente del servicio obtiene un retorno (esto puede ser un valor Void) o una excepción desde el método remoto invocado. Este modelo es conceptualmente el mismo que el usado en el modelo cliente/servidor.

Figura 197. Modelo Petición respuesta - síncrono



La especificación JAX-RPC no define como la implementación del sistema runtime JAX-RPC soporta el modelo de petición-respuesta síncrono en términos del protocolo subyacente y el transporte. Consulte la especificación SOAP 1.2 Parte 2 para mayores detalles sobre patrones de intercambio de mensajes y conexión por defecto HTTP en el transporte.

El subconjunto de APIs JAX-RPC y el modelo de programación del cliente del servicio soportan el modelo de petición-respuesta síncrono a través del modelo basado en stub.

✓ Relación de JAXM Y SAAJ

El subconjunto de la especificación no depende de la especificación JAXM 1.0 (Java API for XML messaging) o el API SAAJ (SOAP with Attachments API for Java) definido por JAXM 1.1.

✓ Semántica del Paso de Parámetros

Ver requerimiento R014, Semántica del paso de parámetros, de la especificación JAX-RPC 1.1.

✓ Contexto del Servicio

El subconjunto de la especificación JAX-RPC no requiere implementaciones para proveer soporte para contexto del servicio.

✓ Mensajes SOAP con Anexos

El subconjunto de la especificación JAX-RPC no soporta el uso de mensajes SOAP con anexos como el protocolo subyacente.

✓ Manejador del Mensaje SOAP

El subconjunto de la especificación JAX-RPC no soporta manejadores de mensajes SOAP.

✓ Modo de Literal

Cuando se usa el protocolo SOAP, se construye una llamada RPC con sus parámetros y valor de retorno dentro del elemento body (cuerpo) de un mensaje SOAP. Una parte del mensaje puede ser codificada usando las mismas reglas de codificación o puede representar una definición específica Schema; la última es llamada representación literal.

Las implementaciones del subconjunto de la especificación JAX-RPC deben soportar la representación literal de una petición o respuesta RPC en el cuerpo del mensaje SOAP. La representación codificada no debe ser soportada.

✓ Portabilidad de la Aplicación

El subconjunto de la especificación JAX-RPC requiere que el código del cliente del servicio sea portable a través de múltiples implementaciones del sistema runtime JAX-RPC. La portabilidad es lograda definiendo una interface del proveedor del servicio (SPI) al subconjunto runtime JAX-RPC. La implementación generada de un stub debe interactuar con el runtime por esta SPI. Consulte el capítulo "Runtime Service Provider Interface" para detalles y requerimientos.

1.5. CORRESPONDENCIA DE WSDL/XML CON JAVA

Este capítulo especifica el mapeo estándar de las definiciones WSDL con la representación Java y el mapeo de los tipos de datos XML a los tipos de Java.

La especificación de la correspondencia de WSDL/XML con Java incluye lo siguiente:

- Mapeo de tipos de datos XML a los tipos de Java
- Mapeo de definiciones abstractas de tipos de puertos, operaciones y mensajes con interfaces y clases java.
- Representación Java de una especificación de dirección wsdl:port
- Representación Java de una definición wsdl:service

Consultar los ejemplos ilustrativos de la especificación JAX-RPC 1.1

✓ Nombres XML

Ver JAX-RPC 1.1 "Nombres XML".

✓ Correspondencia de tipos de XML a Java

Esta sección especifica el mapeo de tipos estándar de tipos de datos XML a los tipos de Java. La implementación debe soportar los tipos especificados.

Observe que las reglas y el formato de serialización para tipos de datos XML están basadas en el estilo de codificación.

❖ *Tipos Simples*

La siguiente tabla especifica la correspondencia Java para los tipos de datos simples XML incorporados. Estos tipos de datos XML están definidos en la especificación XML Schema.

Tabla 65. Mapeo Java para tipos de datos simples XML incorporados.

Tipo Simple	Tipo de Java
xsd:string	java.lang.String
xsd:int	Int
xsd:long	Long
xsd:short	Short
xsd:boolean	Boolean
xsd:QName	javax.xml.namespace.QName
xsd:base64Binary	byte[]
xsd:hexBinary	byte[]

El subconjunto de la especificación JAX-RPC no define la correspondencia estándar para el xsd:anyType. Una implementación del subconjunto JAX-RPC no requiere soportar el tipo xsd:anyType.

Hay un número de casos en los cuales en los cuales la construcción de un tipo de datos XML debe será mapeado a una correspondiente clase Java para un tipo de dato primitivo:

Una declaración de un elemento con el atributo nillable igual a true.

La declaración de un elemento con el atributo minOccurs establecido a cero y el maxOccurs establecido a uno o ausente.

La declaración de un atributo con el atributo use establecido a “optional” o ausente.

La siguiente tabla especifica la correspondencia de las declaraciones de elementos para los anteriores casos.

Tabla 66. Correspondencia de las declaraciones de elementos con nillable fijado a true.

La declaración de un elemento con atributo nillable fijado a true y el siguiente tipo	Tipo de Java
xsd:int	java.lang.Integer
xsd:long	java.lang.Long
xsd:short	java.lang.Short
xsd:boolean	java.lang.Boolean
xsd:byte	java.lang.Byte

- *Mapeo para los tipos de datos xsd: float y xsd: double*

La configuración CLDC 1.1 no provee los tipos nativos float y doublé. Para soportar estos las implementaciones deben generar un código que mapee estos tipos de datos a java.lang.String.

Para que las plataformas CLDC 1.1 y CDC provean soporte a los datos float y double, el generador de stubs debe estar dispuesto a generar código que mapee estos tipos de datos al apropiado tipo Java.

- *Arreglos*

Un arreglo XML es mapeado a un arreglo Java con el operador []. Las implementaciones del subconjunto de la especificación JAX-RPC deben soportar las siguientes definiciones para arreglos XML:

Un arreglo derivado de cualquier elemento donde el atributo maxOccurs es un entero no negativo más grande que uno o “unbounded”

Las implementaciones no deben soportar definiciones de arreglos soapenc: Array o wsdl: arrayType.

El tipo de arreglo de elementos Java es determinado basado en el Schema para el arreglo XML. Observe que la dimensión del arreglo es omitida en la declaración de un arreglo Java. El número de elementos en el arreglo Java está determinado en tiempo de creación más que cuando el arreglo es declarado.

- *Estructura XML y Tipos Complejos*

El subconjunto de la especificación JAX-RPC soporta el mapeo de los siguientes tipos de estructura XML:

El xsd: complexType con secuencias de tipos simples y tipos complejos.

Una estructura XML corresponde a una clase de JavaBeans con el mismo nombre como el tipo de estructura XML. Las clases JavaBeans mapeadas proveen métodos set and get para cada propiedad mapeada desde los elementos miembros de una estructura XML.

El identificador y el tipo Java de una propiedad en la clase JavaBeans es mapeado desde el nombre y el tipo del elemento miembro correspondiente en la estructura XML. Consulte la sección "Nombres XML", para el mapeo de nombres XML con identificadores Java.

Las instancias de las clases mapeadas JavaBeans deben ser capaces de formar la representación en estructura XML correspondiente.

Un elemento en un tipo complejo con el atributo maxOccurs fijado a un entero no negativo más grande que uno o "unbounded" es mapeado a un arreglo Java como un par de métodos set y get en la clase JavaBeans. El tipo arreglo de Java es mapeado desde el atributo tipo del elemento XML.

El subconjunto de la especificación JAX-RPC 1.1 no requiere soporte para todas las combinaciones de ocurrencias de restricciones (minOccurs, maxOccurs).

- *API JavaBeans*

No todos los perfiles y configuraciones J2ME proveen soporte para las APIs JavaBeans. Todas las referencias hechas a la funcionalidad suministrada por el API JavaBeans se refieren a la funcionalidad proporcionada por el J2SE (Java 2 Standard Edition).

Las implementaciones del runtime del subconjunto JAX-RPC no requieren proveer soporte para las APIs JavaBeans. El uso de las APIs JavaBeans en el subconjunto JAX-RPC está restringido a las tareas de tiempo de desarrollo, por ejemplo, la generación de stub.

- *Enumeración*

El subconjunto JAX-RPC no provee soporte para las enumeraciones XML.

✓ Mapeo de WSDL con Java

Esta sección describe la correspondencia de un servicio descrito en un documento WSDL a la representación correspondiente Java.

- *El Documento WSDL*

Un documento WSDL corresponde a un paquete Java. El nombre calificado del paquete java es especificado a una aplicación y es especificado durante el mapeo de WSDL a Java. Una herramienta de mapeo de WSDL a Java debe soportar la configuración del nombre del paquete específico de la aplicación durante el mapeo.

Observe que la especificación JAX-RPC no especifica el mapeo estándar de la definición de un espacio de nombres (en un documento WSDL) al correspondiente nombre de paquete Java. Sin embargo, JAX-RPC requiere que una definición de espacio de nombres en un documento WSDL debe tener correspondencia a un nombre único de paquete Java. El nombre del paquete mapeado Java debe seguir las convenciones definidas en la especificación del Lenguaje Java.

La especificación WSDL 1.1 permite referencias a varias definiciones WSDL (ejemplos: protType, message). Tales referencias basadas en QName en WSDL son mapeadas basadas en el paquete Java y las convenciones de visibilidad de nombres.

- *Elementos de Extensibilidad*

La especificación WSDL 1.1 permite la definición de elementos de extensibilidad (que pueden ser específicos a una conexión o tecnología) bajo diferentes definiciones de elementos.

La especificación del subconjunto JAX-RPC especifica mapeo de elementos de extensibilidad por SOAP. Consulte el capítulo "SOAP Binding". Sin embargo, la especificación JAX-RPC no aborda el mapeo de elementos de extensibilidad de algún distribuidor. Una implementación del subconjunto JAX-RPC puede soportar mapeo de elementos de extensibilidad WSDL al costo de la interoperabilidad y la portabilidad de la aplicación.

- *WSDL Port Type*

Un WSDL port type es un conjunto nombrado de operaciones abstractas y mensajes involucrados.

Un WSDL port type es mapeado a una interface Java (denominado Interface de proveedor de Servicio) que extiende la interface java.rmi.Remote. El mapeo de un wsdl: portType a una interface del proveedor del servicio puede usar el elemento wsdl: binding. Consultar la sección "SOAP binding en WSDL" para detalles sobre el uso de la definición de soap: binding en el mapeo de un WSDL port type.

El nombre de la interface del proveedor del servicio es mapeado al nombre del atributo del elemento wsdl: portType. Observe que un nombre de atributo port type define un nombre

único entre todos los port types definidos en un documento WSDL. Consulte el mapeo de nombres XML en identificadores Java en la sección “Nombres XML”.

La interface Java del proveedor del servicio mapeada contiene métodos que corresponden a los elementos wsdl: operation definidos en el wsdl: portType. Consulte el mapeo estándar de una definición wsdl: operation en la sección “Operación WSDL”.

WSDL no soporta herencia de port types, el mapeo estándar del port type WSDL no define soporte para herencia de interfaces Java mapeadas.

Cada método de una interface Java mapeada debe declarar una excepción java.rmi.RemoteException en su cláusula throws. Una RemoteException es la superclase común para excepciones relacionadas a una invocación remota de un método. Consulte la documentación RMI para más detalles de RemoteException.

Un método podría también lanzar excepciones específicas del servicio basadas en el mapeo de unas fallas WSDL. Consulte más detalles en la sección “WSDL Fault”.

- *Operación WSDL*

Una wsdl: operation definida en un wsdl: portType corresponde a un método Java de la interface del proveedor del servicio mapeada. La parte del mapeo de una wsdl: operation a un método Java del subconjunto JAX-RPC, debe incluir el uso del elemento wsdl: binding. Consulte la sección “SOAP Binding en WSDL” para el uso del elemento soap: binding en el mapeo de una operación WSDL.

Una wsdl: operation es nombrada por su nombre de atributo. El nombre de la operación corresponde al nombre del método sobre la Interface mapeada del proveedor del servicio. Consulte la sección “Nombres XML” para el mapeo de nombres XML a identificadores Java.

De acuerdo con el WS-I Basic Profile, el subconjunto JAX-RPC no soporta la sobrecarga de nombres de operación dentro de un wsdl: portType. Consecuentemente los nombres wsdl: operation deben ser únicos dentro de un wsdl: portType.

Las implementaciones del subconjunto de la especificación JAX-RPC deben soportar el mapeo de operaciones con las primitivas de transmisión petición-respuesta y un-sentido. El mapeo estándar Java de operaciones definidas con otras primitivas de transmisión (notificación, solicitud-respuesta) está considerado fuera del alcance del subconjunto de la especificación JAX-RPC.

Las partes del mensaje en los elementos wsdl: input y wsdl: output definidas en una operación abstracta WSDL son mapeados a parámetros sobre la correspondiente firma del método Java. El nombre del parámetro de un método Java corresponde al nombre del atributo de la parte del mensaje correspondiente. El elemento opcional wsdl: fault mapea a una

excepción. Consulte la sección “WSDL Fault” para mayores detalles sobre la correspondencia entre Java y WSDL faults.

- *Modos de Paso de Parámetros*

El subconjunto de la especificación JAX-RPC no soporta paso de parámetros por referencia para un servicio remoto. JAX-RPC no soporta el paso de una instancia java.rmi.Remote en la invocación de un método remoto.

WSDL parameterOrder (orden de los parámetros)

De acuerdo con la especificación WSDL 1.1, una operación tipo petición-respuesta puede especificar una lista de parámetros usando el atributo parameterOrder. El WS-I Basic Profile hace los siguientes requerimientos para el atributo parameterOrder:

El orden de las partes en un mensaje debe estar en el orden definitivo de los elementos separados en el alambre.

El atributo parameterOrder puede ser usado como una sugerencia para indicar el valor de retorno y las firmas de los métodos.

Si el atributo parameterOrder está presente, debe omitir como máximo una parte del mensaje de salida.

Si no se omite una parte no hay valor de retorno.

La especificación JAX-RPC define las siguientes reglas para el valor de retorno y las formas de paso parámetros: entrada, salida y entrada/salida.

Un parámetro de entrada es pasado como copia. El valor del parámetro de entrada es copiado antes de una invocación remota.

El valor de retorno es creado como copia y es retornado al llamador de un método remoto. El llamador se convierte en el propietario del objeto retornado después de completada la invocación del método remoto.

- *Holder Clases*

El subconjunto JAX-RPC incluye las clases Holder (Ver JAX-RPC 1.0 sección 4.3.5 “Holder Classes”). Los Holders no son requeridos, el WS-I Basic Profile limita el SOAP body a lo máximo una parte del mensaje. Ver capítulo “SOAP Binding” para más detalles.

- *WSDL Fault (Fallas en WSDL)*

El elemento wsdl: fault (un elemento opcional en una operación wsdl) especifica el formato de mensaje abstracto para algunos mensajes de error que pueden salir como resultado de una operación remota. De acuerdo a la especificación WSDL, un mensaje fault debe tener una sola parte.

En el subconjunto JAX-RPC un wsdl: fault debe ser mapeado a una java.rmi.RemoteException (o una subclase), o una excepción Java específica del servicio. Consulte la sección "SOAP Fault" para más detalles sobre el mapeo de un WSDL fault basado en el SOAP binding.

Consulte JAX-RPC 1.0 sección 14.3.6, "Mapeo de Excepciones Remotas" para el mapeo entre las fallas estándar SOAP y la clase java.rmi.RemoteException.

- *Excepción Específica del Servicio*

Una excepción Java específica del servicio (mapeada desde un wsdl: fault y el correspondiente wsdl: message) hereda de la clase java.lang.Exception directa o indirectamente.

La única parte del mensaje en el wsdl: message (referenciado desde el elemento wsdl: fault) puede ser xsd: complexType o un tipo simple XML.

Cada elemento dentro del xsd: complexType es mapeado a un método get y un parámetro en el constructor de la excepción Java. El mapeo de estos elementos sigue el mapeo estándar de XML a tipos Java. El nombre de la clase de excepción Java es mapeada desde el nombre del atributo del xsd: complexType para la única parte del mensaje. Este esquema de nombrado permite el mapeo de WSDL a Java para mapear una derivación jerárquica de xsd: complexType a la correspondiente jerarquía de clases excepción de Java. La siguiente sección muestra un ejemplo. Consulte la sección "Nombres XML" para el mapeo de nombres XML a identificadores Java.

Si la única parte del mensaje en el wsdl: message (referenciado desde el elemento wsdl: fault) tiene un tipo simple XML o un arreglo, entonces este elemento es mapeado a un método get y un parámetro en el constructor de la excepción Java. En este caso, el nombre de la clase excepción Java es mapeado desde el nombre del atributo del elemento wsdl: message.

La excepción Java específica del servicio es declarada como una checked exception (una excepción que debe ser capturada obligatoriamente) en el método correspondiente Java para el elemento wsdl: operation. Esto es una adición al requerido java.rmi.RemoteException.

Ver los ejemplos de JAX-RPC 1.0 sección 4.3.6, "WSDL Fault"

- *WSDL Port*

Un elemento wsdl: port especifica una dirección para un puerto del servicio (o proveedor) basado en el protocolo binding específico. Un wsdl: port debería tener un nombre único entre todos los puertos definidos dentro de un documento WSDL.

En el subconjunto JAX-RPC el modelo de programación del cliente del servicio, un proveedor del servicio (definido usando wsdl: port) es accedido usando una instancia de una clase generada stub.

- *Servicio WSDL*

El subconjunto de JAX-RPC no provee una correspondencia en Java para wsdl: service. Los elementos wsdl: port solo son accesibles a través de una instancia de una clase stub generada.

1.6. SOAP BINDING

Este capítulo especifica el soporte del subconjunto JAX-RPC para el SOAP 1.1 binding.

✓ SOAP Binding en WSDL

El elemento soap: binding en el WSDL identifica que el protocolo SOAP es usado para vincular las definiciones abstractas WSDL.

Las implementaciones del subconjunto de la especificación JAX-RPC deben soportar el siguiente caso (denominado, modo de operación) para una operación con el SOAP binding. Después se especificarán más detalles:

Operación con el document style y uso de literal (document/literal)

Consultar la especificación WSDL 1.1 para más detalles sobre document operation style.

Una implementación del subconjunto JAX-RPC debe usar el anterior modo de operación para el mapeo de una descripción del servicio basada en WSDL a la correspondiente representación Java.

Una implementación del subconjunto JAX-RPC debe adicionalmente cumplir los requerimientos de la sección "SOAP Binding" de la WS-I BP, donde estos requerimientos coincidan con el subconjunto de la especificación JAX-RPC.

✓ Operation Style Atributo

El atributo style (especificado por el elemento soap: operation o como valor por defecto en el elemento soap: binding) indica si una operación es rpc u orientada a documento.

En el modelo de programación del subconjunto JAX-RPC, las operaciones estilo documento son mapeadas a los métodos correspondientes sobre una interface de proveedor de servicio. El estilo de operación rpc no está soportado.

La especificación del subconjunto JAX-RPC requiere soporte para la siguiente representación por defecto del elemento SOAP Body para operaciones estilo documento. La siguiente representación por defecto es aplicable a los servicios que la implementación del subconjunto JAX-RPC debe soportar:

El SOAP body debe conducir a contener como máximo una parte de mensaje (wsdl: part), definido por el elemento form en el nivel abstracto. Esto concuerda con el WS-I Basic Profile.

Todas las partes del mensaje (los parámetros o el valor de retorno) deben aparecer dentro de un solo elemento que los contiene, el cual es el primer elemento hijo del elemento SOAP Body. El elemento contenedor para el requisito debe tener un nombre idéntico al único nombre de operación. El nombre del elemento contenedor para una petición es usado en el lado servidor para resolver el método sobre el proveedor del servicio objetivo.

La parte del mensaje debe tener un accesor con el nombre correspondiente al nombre del parámetro y tipo correspondiente al tipo del parámetro. Consulte la especificación SOAP para detalles completos sobre la representación de las invocaciones y respuestas RPC en el elemento SOAP Body.

Las implementaciones de la especificación JAX-RPC, no deberían hacer uso de SOAPAction. Observe que la especificación SOAP 1.2 especifica SOAPAction como opcional con una recomendación que el SOAPAction no debería ser usado pero puede ser requerido en ciertas implementaciones del servidor.

Una implementación del subconjunto JAX-RPC del lado del cliente debe soportar el uso de servicios que sigan los requerimientos de operaciones estilo documento JAX-RPC especificados.

La especificación JAX-RPC requiere que los requerimientos anteriores basados en el estilo de operación deberían estar ocultos desde el modelo de programación JAX-RPC. Una implementación JAX-RPC debería tomar la responsabilidad para la representación apropiada de un mensaje SOAP basado en el estilo de operación.

- ✓ **Representación Literal**

Las implementaciones del subconjunto de la especificación JAX-RPC deben usar el estilo de documento de la operación, con el uso de literal. Observe que el uso de literal está definido sobre el elemento soap: body en el WSDL.

❖ *Mapeo Java de la Representación Literal*

El mapeo Java para una parte de un mensaje con la representación depende sobre si JAX-RPC especifica un estándar de mapeo de Java para los tipos XML de esta parte del mensaje. Consulte la sección "Mapeo de XML a tipos Java" para el mapeo especificado de su subconjunto de XML Schema.

✓ SOAP Fault

Esta sección especifica el mapeo de SOAP faults.

El atributo nombre relaciona el elemento soap: fault con el elemento wsdl: fault. El elemento wsdl: fault (un elemento opcional en un wsdl: operation) especifica el formato de mensaje abstracto para cualquier mensaje de error que puede ser salido como un resultado de una operación remota.

El elemento soap: fault es modelado después del elemento soap: body en términos del uso literal. De acuerdo a la especificación WSDL 1.1, el elemento soap: fault debe contar solo una única parte de mensaje.

El subconjunto JAX-RPC no provee una clase SOAPFaultException. Un SOAP fault es mapeado a una clase excepción específica del servicio o java.rmi.RemoteException.

1.7. JAX-RPC SUBCONJUNTO DE APIS NÚCLEO

Este capítulo especifica el subconjunto de APIs que soportan los mecanismos JAX-RPC runtime. Estas APIs están empaquetadas en el paquete javax.xml.rpc.

✓ APIs del Lado del Servidor

La Especificación del subconjunto JAX-RPC no define APIs del lado del servidor. Consulte más detalles sobre las APIs del lado del servidor en la especificación JAX-RPC 1.0

✓ APIs del Lado del Cliente

El subconjunto JAX-RPC especifica las siguientes APIs del lado del cliente:

La interface javax.xml.rpc.stub

La clase javax.xml.JAXRPCException

Una implementación del sistema runtime del subconjunto JAX-RPC debe implementar las anteriores APIs

- *Clase Stub Generada*

Ver la sección "Clase Stub Generada" y "configuración Stub" de JAXRPC 1.0 para detalles y requerimientos.

- *JAXRPCException*

La excepción javax.xml.rpc.JAXRPCException es lanzada desde las APIs núcleo para indicar excepciones relacionadas con los mecanismos de tiempo de ejecución JAX-RPC. Una excepción JAXRPCException es mapeada a java con java.rmi.RemoteException si aquel es lanzado durante el procesamiento de una invocación a un método remoto.

✓ Modelo de Programación del Cliente del Servicio basado en J2ME

Un cliente basado en J2ME usa clases generadas stub para acceder al servicio. Los siguientes pasos son requeridos para que un cliente del subconjunto JAX-RPC interactúe con un proveedor del servicio:

Generar un stub desde la descripción del servicio WSDL

Instanciar un objeto del stub

Invocar métodos del stub correspondientes a la implementación de wsdl: operation del proveedor del servicio.

Empaquetar el stub con la aplicación cliente J2ME

- *Generación de Stub*

El código para un stub es generado en el momento de desarrollo. La implementación de la interface stub debe heredar de javax.xml.rpc.Stub. El desarrollador J2ME usa una herramienta que lee la descripción del servicio WSDL que el cliente accederá y generará el código Java apropiado.

El código generado por el generador de stub usa un subconjunto de runtime SPI para invocar las operaciones del servicio. Un stub debe usar solo el SPI para interactuar con el runtime. Ver capítulo 8 "Interface del Proveedor del Servicio Runtime".

- *Instanciación de Stub*

Un cliente J2ME instancia un objeto del stub como un "proxy" para el servicio desde el cual fue creado. El programa cliente simplemente crea una nueva instancia de la clase stub generada por el generador de stub, por ejemplo:

StockQuoteService_Stub stub = new StockQuoteService_Stub();

- *Operaciones del Stub*

Una aplicación cliente J2ME usa una instancia de un stub para:

Cambiar las propiedades del stub, por ejemplo, un nombre de usuario y password para la autenticación básica HTTP, o fijar una dirección del proveedor del servicio.

Invocar operaciones del proveedor del servicio

El siguiente ejemplo muestra como una aplicación cliente cambia las propiedades de una instancia stub:

stub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, "http://quotes-r-us.com:8080:/quoteservice/");

Se cambió la propiedad dirección de un objeto Stub.

La aplicación cliente usa el stub para invocar una operación, en este ejemplo la operación "getMostActive" es suministrada por el servicio:

StockQuote [] mostActive = stub.getMostActive ();

Código ejemplo de la invocación de una operación de un proveedor de servicio a través del objeto Stub.

- *Empaquetado*

Una aplicación cliente J2ME que hace uso de un stub para invocar operación de un proveedor de servicio debe incluir el stub generado, acompañado con otros artefactos generados, en su paquete de despliegue.

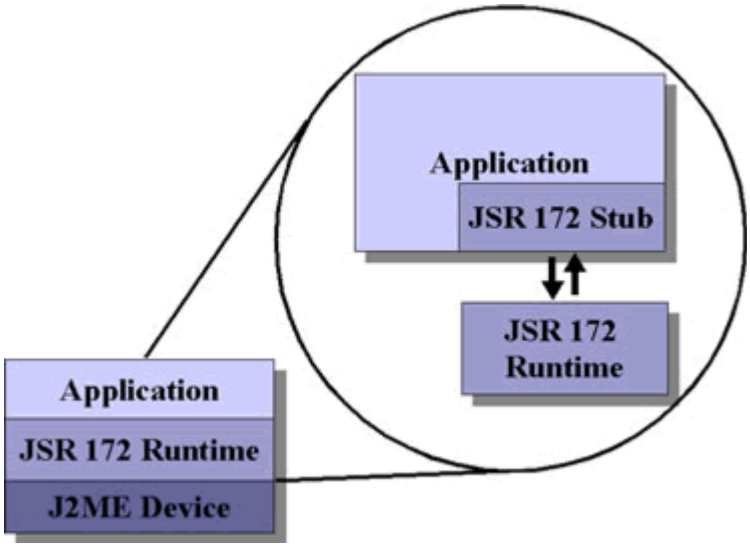
1.8. INTERFACE DEL PROVEEDOR DEL SERVICIO RUNTIME

Este capítulo describe la interface del proveedor del servicio (SPI) del subconjunto runtime JAX-RPC. Vea el paquete javax.microedition.xml.rpc de la documentación java, para obtener una descripción completa de las clases e interfaces que forman el SPI.

✓ Visión General

El subconjunto runtime JAX-RPC SPI es usado por stubs generados para ejecutar llamadas RPC, no está pensado para ser usado por desarrolladores de aplicaciones. El SPI solo soporta la funcionalidad descrita en el subconjunto JAX-RPC, por ejemplo, no hay soporte para DII o para proxy dinámicos. El SPI está definido por la clase ValueType y la clase abstracta Operation.

Figura 198. Clases SPI del subconjunto JAX-RPC Runtime



La clase ValueType es usada por el stub para describir y pasar los valores de algunos parámetros de tipo complejo o valores de retorno de un RPC, a y desde el runtime. Una instancia de un objeto ValueType representa un serialización de los valores en un tipo complejo.

✓ Invocando un RPC

Un stub usa el SPI para ejecutar las siguientes acciones:

Crear un objeto Operation representando una invocación de un RPC

Un conjunto de propiedades que son necesarias para invocar un RPC

Crear y poblar un arreglo ValueType de los parámetros de entrada

Poblar el mapa de tipos de algunos tipos complejos usados en una invocación RPC

Pasar la descripción de los parámetros de entrada y valores de retorno para un RPC al runtime

Invocar un RPC

Recuperar valores de retorno desde la invocación RPC

Las siguientes secciones describen estos pasos con mayor detalle.

- *Fijando Propiedades*

El método setProperty de la clase Operation es llamado por un stub generado para cambiar los valores de una propiedad de una invocación RPC. El stub es responsable de la conversión del valor de la propiedad en el formato apropiado.

La siguiente tabla describe los nombres de propiedad que deben ser soportados:

Tabla 67. Nombre de las propiedades requeridas

Nombre de la Propiedad	Descripción
Stub.ENDPOINT_ADDRESS_PROPERTY	Dirección del proveedor del servicio
Stub.PASSWORD_PROPERTY	Password para la autenticación
Stub.USERNAME_PROPERTY	Nombre de usuario para la autenticación
Stub.SESSION_MAINTAIN_PROPERTY	Indicar si un cliente quiere participar en una sesión con un proveedor de servicio

Una excepción JAXRPCException debe ser lanzada por el runtime si una propiedad suministrada no es soportada.

- *Parámetros de Entrada*

Un stub usa la clase ValueType para pasar a o recibir desde, los parámetros de una Operation. El stub es responsable de tomar los parámetros de entrada y crear la clase apropiada ValueType poblada. Cada elemento value de un ValueType es objeto wrapper de los tipos primitivos soportados (Integer, Short, String, etc) o, si el valor es un tipo complejo, otro

ValueType. Los elementos deben ser presentados en el orden en el cual ellos ocurren primero de acuerdo al orden de declaración de los parámetros en la firma del método.

Si los parámetros de entrada, o tipo de retorno, de una operación contienen tipos complejos, el stub debe pasar un map de la información de tipos, como una Hashtable, al runtime. Cada entrada en el Hashtable es el tipo de un elemento value. El QName del elemento es usado como clave. El método setTypeMap de Operation es usado por el stub para asignar el tipo map para una operación.

Si un parámetro de entrada, o tipo retorno, es declarado como nillable el stub debe llamar el método setNillable de ValueType para marcar el tipo como nillable. El método isNillable de ValueType puede ser usado para determinar si un tipo ha sido marcado para ser nillable.

Observe – Separar la información del nombre y el tipo en un descriptor separado (el tipo map) evita tener que repetirlo para cada elemento en el arreglo de tipo complejo.

En el siguiente ejemplo un stub crea un ValueType para pasar una orden de compra (la clase PurchaseOrder) conteniendo ambas direcciones de facturación y de envío, ambas de las cuales son instancias de la clase Address:

```
public class Address {

public String street;

public String city;

public String getStreet() {

...

public String getCity() {

...

}

public class PurchaseOrder {

public Address shippingAddress;

public Address billingAddress;

public Address getShippingAddress() {

...

pubic Address getBillTo() {
```

```
...
}
// order method of stub
public order(PurchaseOrder po) {
...
Address shipTo = po.getShippingAddress();
Address billTo = po.getBillTo();
Hashtable map = new Hashtable();
map.put(streetQName, ValueType.STRING);
map.put(cityQName, ValueType.STRING);
map.put(addressQName,
new Object[] { streetQName, cityQName });
map.put(purchaseOrderQName,
new Object[] { addressQName, addressQName });
operation.setTypeMap(map);
ValueType vtShipTo = new ValueType(addressQName,
ValueType.COMPLEXTYPE,
new Object[] { shipTo.getStreet(), shipTo.getCity()});
ValueType vtBillTo = new ValueType(addressQName,
ValueType.COMPLEXTYPE,
new Object[] { billTo.getStreet(), billTo.getCity()});
ValueType vtOrder[] = {
new ValueType(purchaseOrderQName,
ValueType.COMPLEXTYPE,
new Object[] { vtShipTo, vtBillTo })
```

};

...

Código Ejemplo, Construyendo una clase ValueType

Nota sobre Estilos Wrapped y Un-wrapped

Con el document/literal style/use hay dos posibles estilos para las firmas de los métodos del stub. El primero se refiere a “wrapped”. En el ejemplo anterior el wrapper es la clase PurchaseOrder pasada al método order. Incluso en ejemplos más sencillos donde todos los parámetros de un método son primitivos la firma del método requiere una clase wrapper.

La segunda posibilidad es el estilo “unwrapped. Para adaptar el ejemplo anterior al uso de este estilo, el método tomaría dos objetos Address como parámetros en lugar de PurchaseOrder.

Debemos considerar requerir que las implementaciones soporten ambos estilos de forma transparente. La información puede moverse desde la firma del método en la implementación stub así ambos estilos parecen idénticos en tiempo de ejecución.

Ejecutando un RPC

Un stub ejecuta una operación usando el método invoke pasándole un arreglo de ValueType con los parámetros de esta llamada, o null si la operación no necesita parámetros, y ValueType describiendo la forma del tipo de retorno, ver sección “Valores de Retorno” para más detalles.

try {

ValueType return[] = op.invoke(vtIn, vtOut);

Catch (JAXRPCException jre) {

...

}

Código Ejemplo Invocando una Operación

La referencia ValueType retornada por invoke debe ser como el parámetro ValueType describiendo la forma del tipo de retorno.

Valores de Retorno

El método invoke retorna un arreglo de ValueType con el valor de retorno de una operación o null si la operación no retorna un valor.

El stub usa la clase ValueType para describir cualquier valor de retorno al runtime. Como con parámetros de entrada, si el valor de retorno contiene algunos tipos complejos, estos deben ser descritos en el mapa de tipos y pasado al runtime usando el método setTypeMap. Los elementos value del valor de retorno ValueType debe ser null.

Un solo mapa de tipos contiene información de tipos para ambos parámetros de entrada y parámetros de retorno. Una sola llamada a setTypeMap es requerida.

Manejo de Errores

Si un error (SOAP: Fault) ocurre mientras se ejecuta una operación una JAXRPCException es lanzada por el método invoke de Operation. El método getLinkedCause de JAXRPCException es usado por el stub para recuperar la excepción generada.