



Universidad Complutense

Facultad de Informática



Asignatura: Ingeniería del Software
Curso Académico: 2008/2009
Grupo: 4º B

Índice

Introducción	4
Escenarios de Calidad del Software	5
1. Introducción	5
2. Modificabilidad	6
3. Usabilidad.....	10
4. Disponibilidad	14
Vistas	16
1. Vista de Asignación de Trabajo.....	16
1.1. Interfaz.....	16
1.2. Servidor + Cliente-Servidor.....	16
2. Vista de Implementación	18
2.1. Aspectos Generales.....	18
2.2. Herramientas de Desarrollo	20
3. Vista de Despliegue	21
3.1. Presentación de la Vista	21
3.2. Catálogo de Elementos.....	21
4. Vistas de Descomposición.....	23
4.1. General	23
4.2. Cliente	25
4.3. Gestor de Archivos.....	30
4.4. Interfaz de la aplicación Cliente.....	31
5. Vistas de Uso	34
5.1. Gestor de Archivos.....	34
5.2. GUI de la aplicación Cliente	37
5.3. Protocolo P2P	40
6. Vista de Datos Compartidos	45
6.1. Presentación de la Vista	45
6.2. Catálogo de Elementos.....	46
6.3. Guía de Variabilidad	47
6.4. Información sobre la Arquitectura.....	47
7. Vista de Cliente-Servidor.....	48
7.1. Cliente como Servidor.....	48
7.2. Cliente como Cliente	52

7.3.	Protocolo P2P	57
8.	Vista de Procesos Comunicados.....	63
8.2.	Mensajes.....	63
8.3.	Envío Cliente-Servidor.....	64
8.4.	Envío Cliente-Cliente	64
8.5.	Traza de las peticiones	64



Introducción

En este documento se presenta al lector una perspectiva general de la **Arquitectura del Sistema** correspondiente a la aplicación P2P **eGorilla** desarrollada por los alumnos de la asignatura de **Ingeniería del Software** de la **Facultad de Informática en la Universidad Complutense de Madrid** durante el curso académico 2008/2009 en el grupo 4º B.

El documento se divide en dos grandes áreas:

- Una primera parte correspondiente a los **Escenarios de Calidad del Software** que nos permiten identificar **los requisitos no funcionales** del mismo.
- Una segunda parte correspondiente a las **Vistas** que definen tanto comportamiento como estructura del sistema mediante diversos diagramas de diversa naturaleza que ayudan al lector a la comprensión del comportamiento del sistema.





Escenarios de Calidad del Software

1. Introducción

Los atributos de calidad del software, son **requisitos no funcionales** que debe cumplir un sistema. Para permitirnos identificar cuáles son esos atributos que debe cumplir nuestra aplicación, vamos a utilizar los **escenarios de calidad**.

Los escenarios de calidad son **situaciones** que nos permiten identificar qué es **deseable** en nuestro sistema **para poder cumplir** determinados **atributos de calidad**.

Los atributos en los que nos vamos a centrar son los marcados en el índice pues nos parecen los **primordiales** para nuestro tipo de aplicación, a saber:

Modificabilidad, Usabilidad, Disponibilidad.





2. Modificabilidad

Un factor a tener en cuenta como muy buen atributo de calidad del software, es que éste sea razonablemente modificable durante la fase de desarrollo y posterior vida del producto generado. Por eso el diseño de la arquitectura de la aplicación es un diseño modular, que permite acometer a los desarrolladores y posteriores mantenedores de la aplicación, cambios en la misma de manera aislada.

Parte escenario	del	Valores posibles
Fuente		Desarrollador/Cliente
Estímulo		El desarrollador desea: cambiar la GUI; cambiar el protocolo de la aplicación; cambiar el protocolo de red; cambiar la información presentada por las estadísticas.
Artefacto		Código fuente de la aplicación
Entorno		En el tiempo de desarrollo o mantenimiento del sistema



**Respuesta**

El sistema proporciona las siguientes respuestas:

- Para dar soporte al estímulo “**cambiar la GUI**”:

Modificación del módulo de la aplicación encargado de la gestión de la GUI. No debe ser necesario modificar ningún otro punto de la aplicación.

- Para dar soporte al estímulo “**cambiar el protocolo de la aplicación**”:

Modificación del módulo de la aplicación encargado de la implementación del protocolo de comunicación entre clientes y servidores. No debe ser necesario modificar ningún otro punto de la aplicación.

- Para dar soporte al estímulo “**cambiar el protocolo de red**”:

Modificación del módulo de la aplicación encargado de la gestión de la capa de transporte (protocolos TCP/UDP). No debe ser necesario modificar ningún otro punto de la aplicación.

- Para dar soporte al estímulo “**cambiar la información presentada por las estadísticas**”:

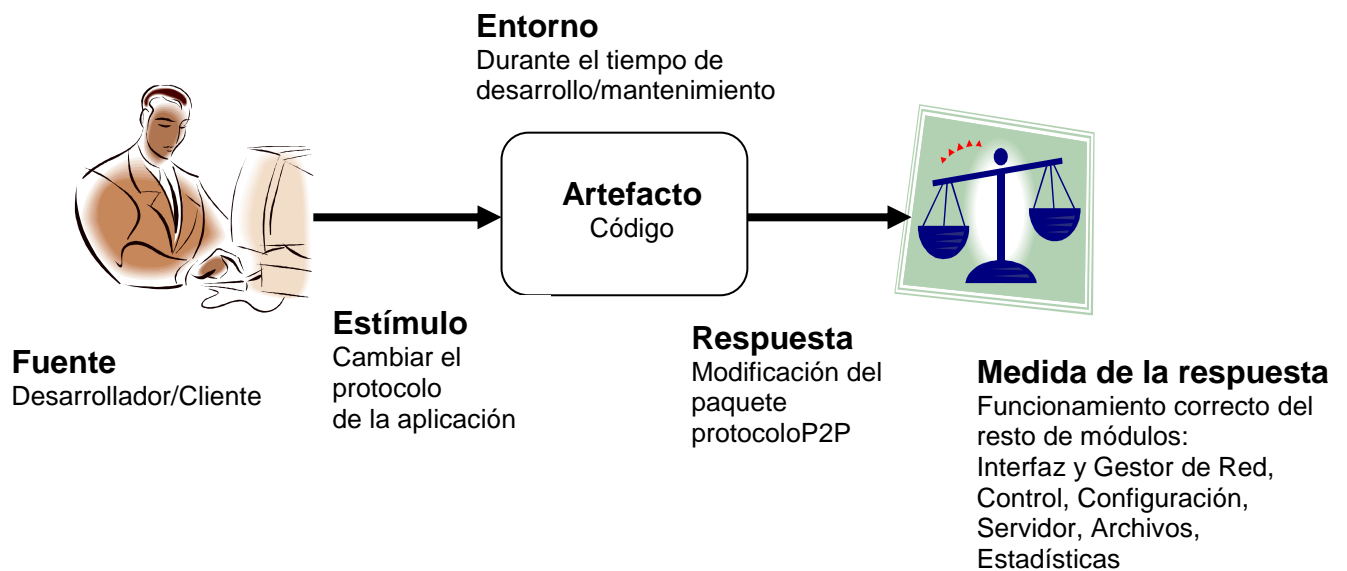
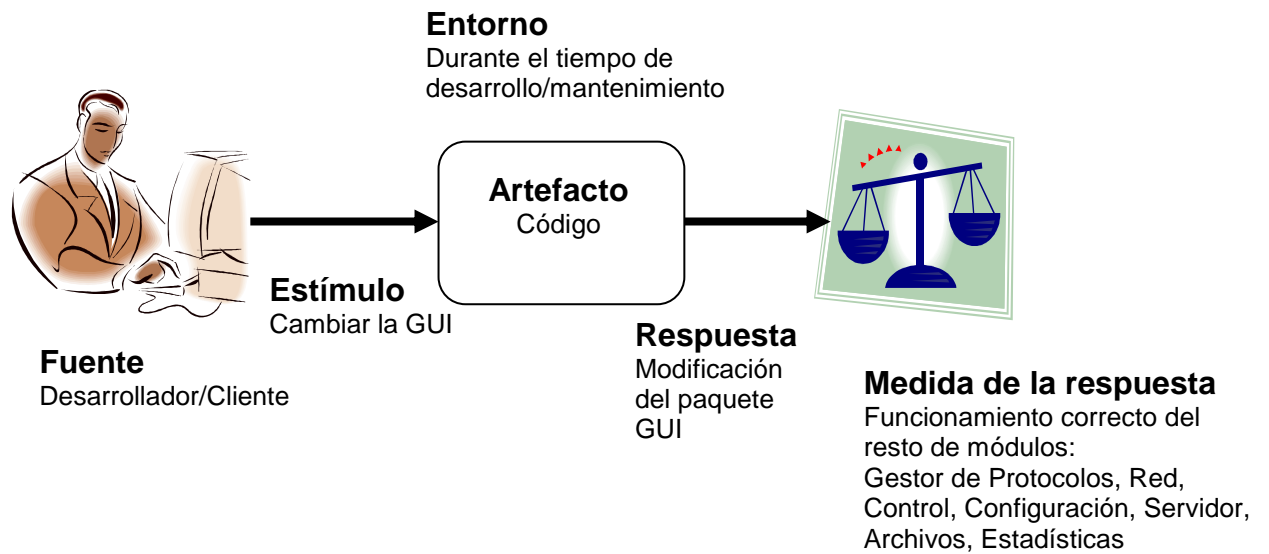
Modificación del módulo de la aplicación encargado de la recepción y proceso de las estadísticas (más o menos suscripciones a eventos, mayor o menor procesamiento de los datos recibidos, etc.)

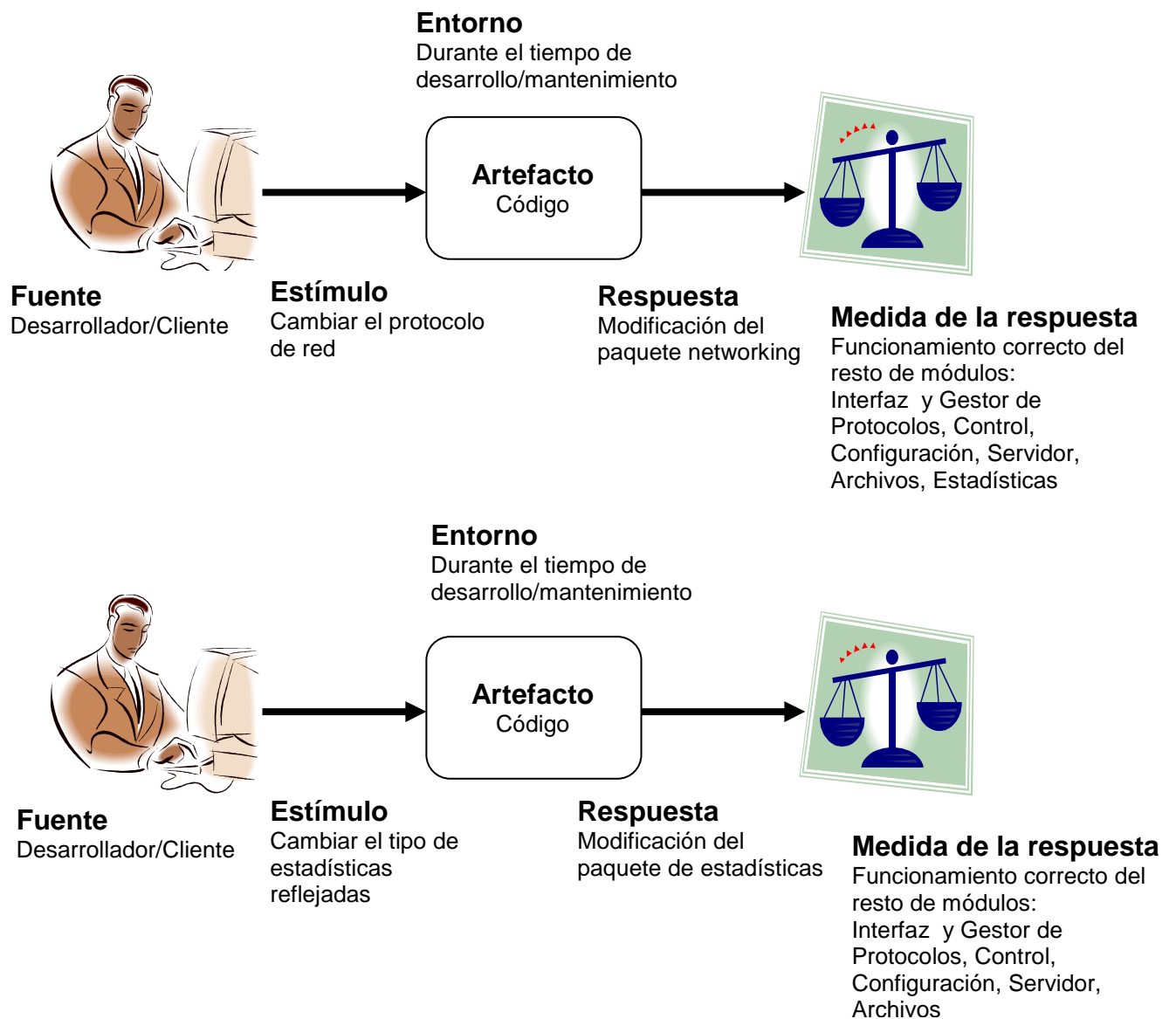
Medida de la respuesta

En todos los casos, la capacidad de modificabilidad será medida por el alcance del objetivo nuevo en el módulo o módulos cambiados, manteniendo al margen el resto de funcionalidad no relacionada. Esto es, que además de constatar las nuevas funcionalidades, éstas no produzcan efectos secundarios no deseados en el resto de módulos de la aplicación.

A continuación, se presentan estos escenarios gráficamente, y se concreta qué módulo cambia, y como medida de la respuesta, aquellos que no deberían verse afectados:









3. Usabilidad

La usabilidad es la mayor o menor facilidad con la que un sistema es utilizado por sus usuarios. Por esta razón es un atributo de calidad importante y que resulta decisivo en la aceptación del sistema por los usuarios.

Parte del escenario	Valores posibles
Fuente	Usuario final
Estímulo	El usuario final desea: aprender a utilizar la aplicación rápidamente, usar el sistema de forma eficiente, configurar rápidamente la aplicación, conocer las posibilidades de uso del sistema, sentirse cómodo en el manejo de la aplicación.
Artefacto	Sistema
Entorno	Tiempo de ejecución del sistema.



**Respuesta**

El sistema proporcionará las siguientes respuestas:

- Para dar soporte al estímulo **“aprender a utilizar la aplicación rápidamente”**:

Interfaz sencilla y muy intuitiva (uso de técnicas como el “click”, “arrastrar y soltar”, etc), entorno familiar al usuario; ayuda sensible al contexto.

- Para dar soporte al estímulo **“configurar rápidamente la aplicación”**:

Fácil acceso a la configuración general del sistema; acceso inmediato a los parámetros de configuración más utilizados; posibilidad de restaurar la configuración “estándar”.

- Para dar soporte al estímulo **“usar el sistema de forma eficiente”**:

Configuración de los parámetros que afectan a la funcionalidad y/o rendimiento de la aplicación; realización de varias actividades simultáneamente.

- Para dar soporte al estímulo **“conocer las posibilidades de uso del sistema”**:

Ayuda de la aplicación integrada en la interfaz del sistema, complementaria de la ayuda sensible al contexto.

- Para dar soporte al estímulo **“sentirse cómodo en el manejo de la aplicación”**:

Mostrar información del estado del sistema; mostrar de forma gráfica el rendimiento del sistema; mostrar avisos al usuario cuando sea necesario.

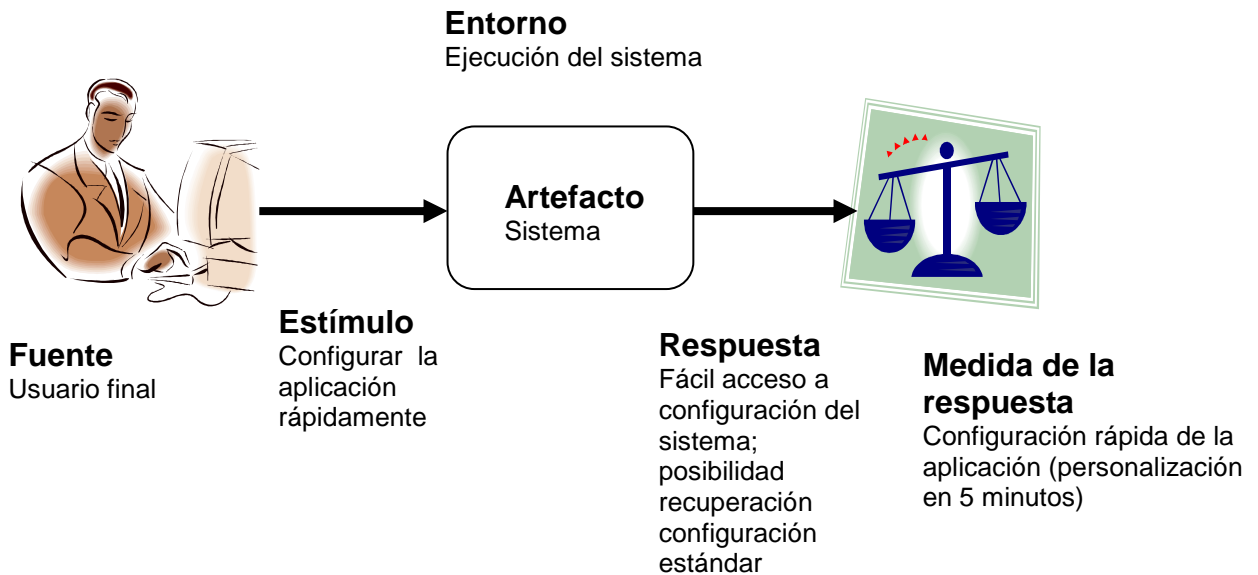
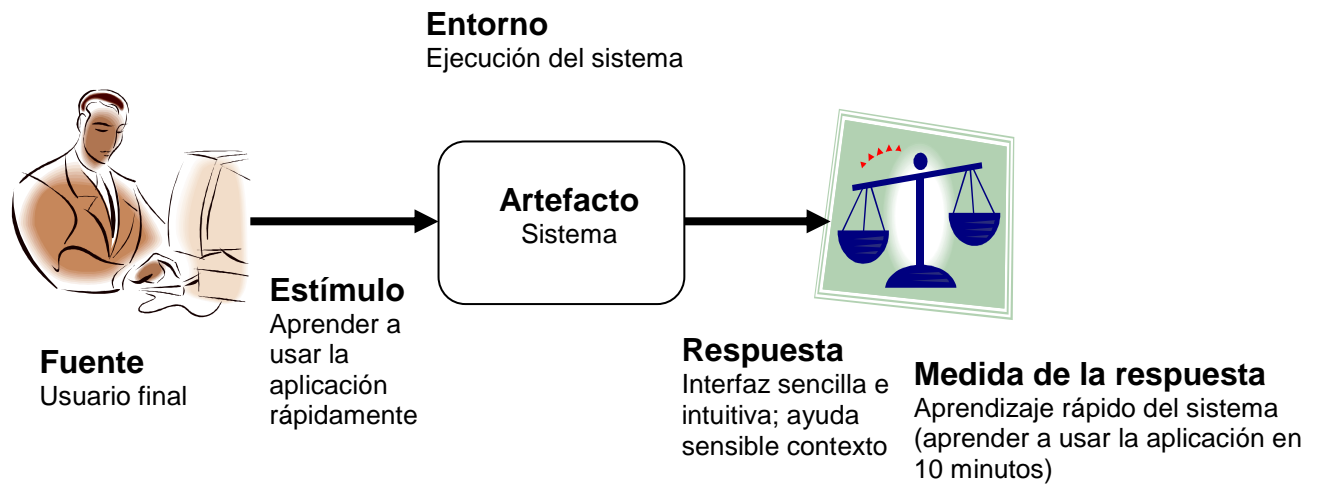
Medida de la respuesta

Satisfacción del usuario; aprendizaje rápido del sistema (aprender a usar la aplicación en 10 minutos); configuración de la aplicación adecuada a sus necesidades (personalización de la aplicación en 5 minutos/configuración de los parámetros de rendimiento en 5 minutos); progreso en el conocimiento de la aplicación (aprender a utilizar cualquier funcionalidad del sistema mediante la ayuda en 10 minutos).



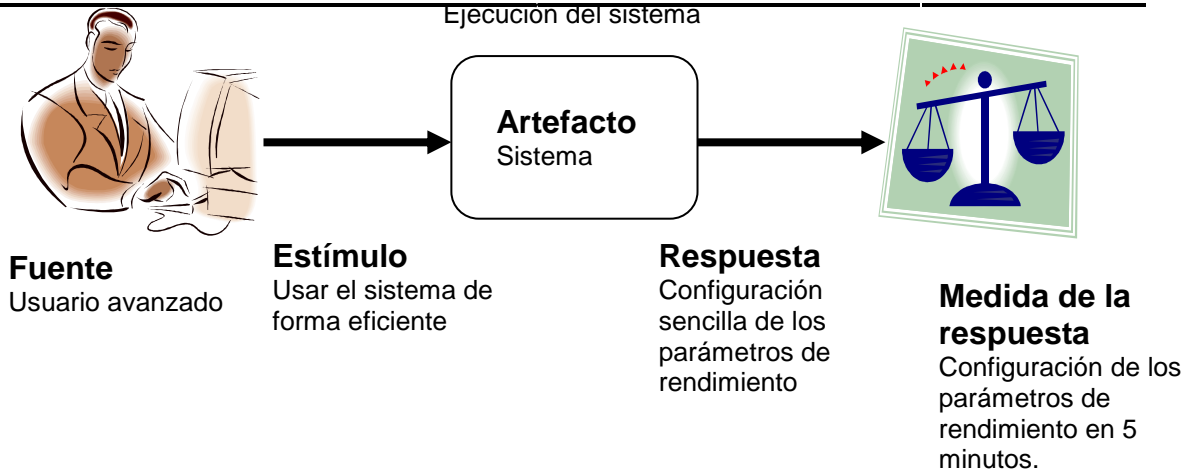


A continuación, se presentan estos escenarios gráficamente:



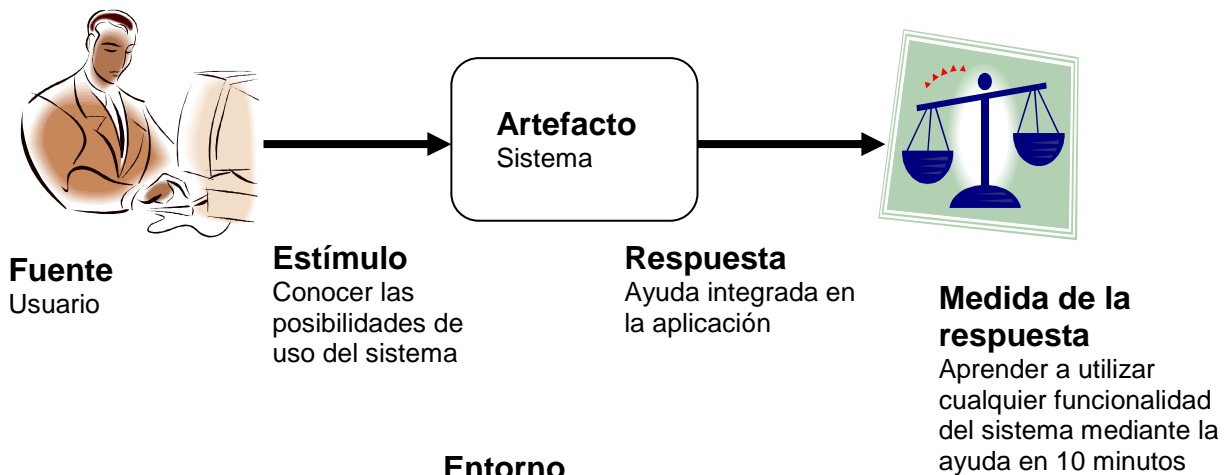
Entorno

Ejecución del sistema



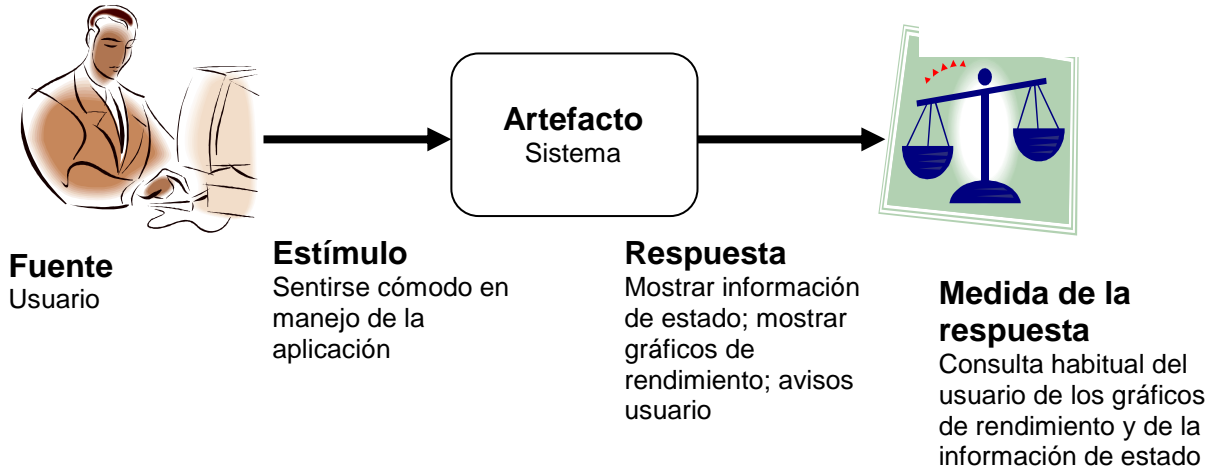
Entorno

Ejecución del sistema



Entorno

Ejecución del sistema





4. Disponibilidad

Un factor a tener en cuenta como muy buen atributo de calidad del software, es que éste sea razonablemente modificable durante la fase de desarrollo y posterior vida del producto generado. Por eso el diseño de la arquitectura de la aplicación es un diseño modular, que permite acometer a los desarrolladores y posteriores mantenedores de la aplicación, cambios en la misma de manera aislada.

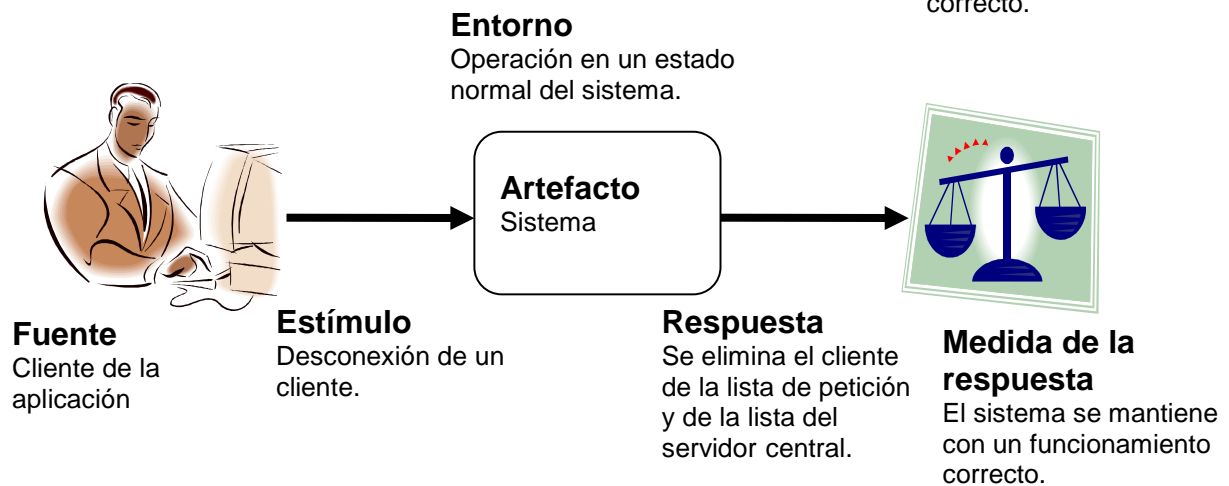
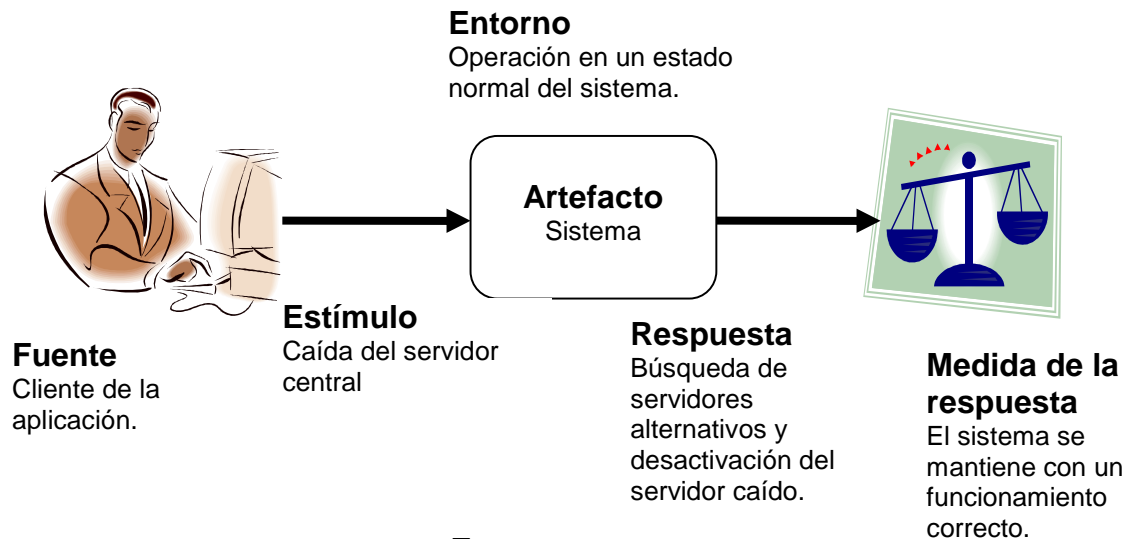
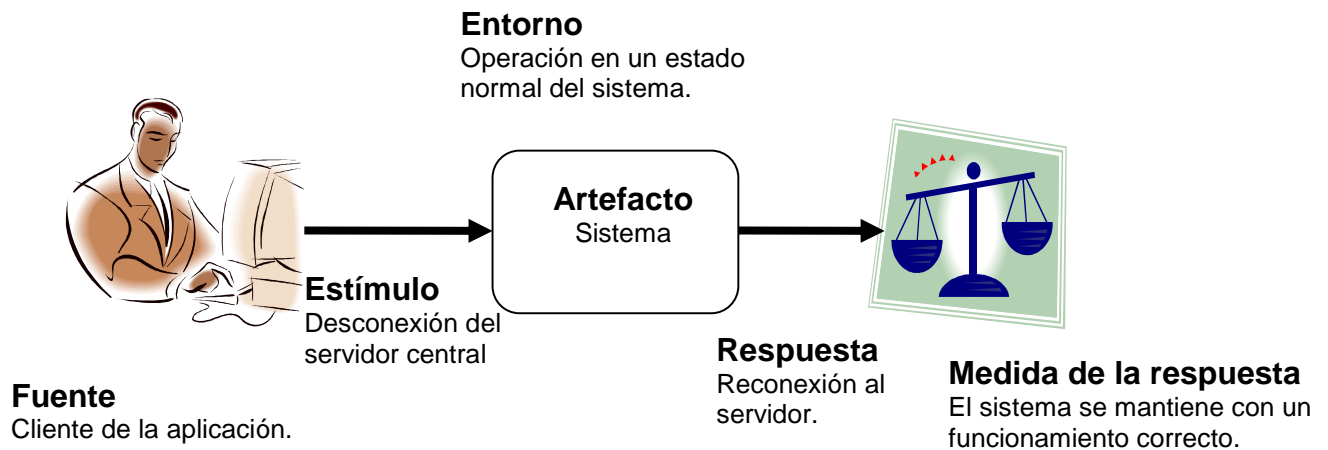
Parte del escenario	Valores posibles
Fuente	Cliente de la aplicación.
Estímulo	Fallo por desconexión a servidor central, fallo por caída del servidor central, fallo de conexión a cliente servidor.
Artefacto	Sistema
Entorno	Operación en un estado normal del sistema.
Respuesta	<p>El sistema proporciona las siguientes respuestas:</p> <ul style="list-style-type: none"> Para dar soporte al estímulo “pérdida de conexión al servidor central”: La aplicación es capaz de reconectarse al servidor si se ha desconectado de éste, y restablecer el funcionamiento normal. Para dar soporte al estímulo “fallo por caída del servidor central”: La aplicación busca servidores alternativos para conectarse, deshabilita el servidor caído y restablece el modo normal de funcionamiento. Para dar soporte al estímulo “fallo de conexión a cliente servidor”: Se dispone de una red de clientes con servicio de compartición de archivos. El sistema cliente es capaz de obtener el archivo de todos los clientes de la red simultáneamente de manera que el fallo de algún cliente de la red no afecte a la disponibilidad del sistema.
Medida de la respuesta	Intervalo de disponibilidad: El sistema debe disponible permanentemente, se ofrecen caminos alternativos a la caída o mantenimiento de un servidor u otro componente de la red.

A continuación se detallan los escenarios de forma gráfica:

Arquitectura del Sistema

Aplicación P2P eGorilla







Vistas

1. Vista de Asignación de Trabajo

1.1. Interfaz

1.1.1. PanelServidores → David

1.1.2. PanelBuscador → Iñaki

1.1.3. PanelTrafico → José Miguel

1.1.4. PanelCompartidos → José Miguel

1.1.5. PanelEstadisticas → Mercedes

1.1.6. PanelConfiguración → Francisco Javier

1.2. Servidor + Cliente-Servidor

1.2.1. Gestor de Archivos → Pedro, Iván

1.2.2. GestorDeRed → Víctor, Luis

1.2.3. GestorDeProtocolos → Víctor, Luis

1.2.3.1. Cliente-Servidor

1.2.3.1.1. ProtocoloDeConexion

1.2.3.1.2. ProtocoloDeDescargas

1.2.3.1.3. ProtocoloDeBusquedas

1.2.3.1.4. ProtocoloDeSubidas

1.2.3.2. Cliente-Cliente





1.2.3.2.1. *ProtocoloDeDescargas*

1.2.4. *GestorDeEstadísticas → Qiang, Daniel*

1.2.5. *GestorDeConfiguración → Daniel, Iván*

1.2.6. *ControlAplicacion → Qiang, Pedro*





2. Vista de Implementación

2.1. Aspectos Generales

2.1.1. Clases

2.1.1.1. Nombres

Empiezan por mayúscula todas las primeras letras de las palabras que conforman el nombre de la clase.

EJEMPLO: ControlPanelCompartidos.

2.1.1.2. Atributos de Clase

Empiezan por '_'. La primera letra de la primera palabra en minúscula y la primera letra de las siguientes palabras en mayúsculas.

EJEMPLO: _ejemploCorrectoCorrectisimo, _ejemplo.

2.1.1.3. Nombres de los métodos

Empiezan por minúscula y la primera letra de las siguientes palabras en mayúsculas. No se utilizará '_' entre palabras.

EJEMPLO: getContador(), actualizarEstadoDeAutomata().

2.1.1.4. Componentes de Swing

Siguen empezando por '_' seguidos de una información acerca del tipo de componente que es.

EJEMPLO: _btnConectar, _txtEntrada, _lblHola, _panelEstado.





2.1.1.5. *Constantes*

En mayúsculas todas las letras. Si tuviera dos o más palabras estarán separadas por '_ '.

EJEMPLO: RUTA_FICHERO_FICHEROSO, RUTA2.

2.1.2. *Variables*

Nombres significativos **en español**, no importando que sean nombres largos, empezando por minúscula y la primera letra de las siguientes palabras en mayúsculas.

EJEMPLO: conexionServidor, contadorIteraciones, archivoTemporal.

2.1.3. *Constantes y Enumerados*

Uso de **public enum** y **no public static final**. Los nombres de los enumerados han de ser de igual forma significativos.

EJEMPLO CORRECTO: public enum EventoPanelCompartidos {INICIO, PULSACION}

EJEMPLO INCORRECTO: public final static int ACCION1.

2.1.4. *Comentarios*

2.1.4.1. *Javadoc*

Cada método de clase llevará sus comentarios en **javadoc** separado por la **línea divisoria correspondiente** y **en español**.

2.1.4.2. *Normales*

Cuando haya **algo pendiente añadir la etiqueta TODO**: con una pequeña explicación acerca de la tarea. Además se tiene en consideración:

2.1.4.2.1. Una línea → //

2.1.4.2.2. Más de una línea → /* */





2.2. Herramientas de Desarrollo

2.2.1. NetBeans 6.1 sobre jdk 1.6

La razón en particular de ésta versión y no de la anterior es porque ofrece un **mayor rendimiento para hacer los UMLs** ya que la versión 6.5 no funciona del todo bien en este aspecto.

NetBeans ofrece también la posibilidad de **edición** de interfaces gráficas, hecho que junto con el anterior acabó convenciendo de su uso frente a la otra herramienta de desarrollo propuesta Eclipse.

2.2.2. Codificación UTF-8.





3. Vista de Despliegue

3.1. Presentación de la Vista

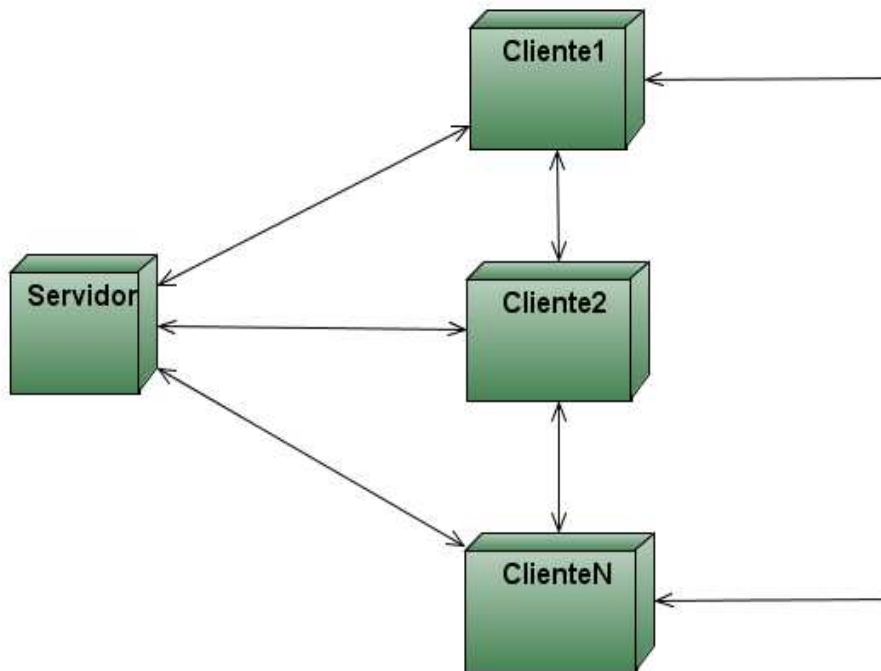


Ilustración 1 - Vista de Despliegue

3.2. Catálogo de Elementos

3.2.1. Elementos y propiedades

3.2.1.1. Servidor

Único servidor de la aplicación. Habrá **un único servidor** al cual estarán conectados todos los clientes.

3.2.1.2. ClienteX

Uno de los múltiples clientes que tiene el sistema.





3.2.2. Relaciones

3.2.2.1. Servidor-ClienteX

El servidor se comunica con los clientes que se ejecutan en máquinas diferentes mediante el protocolo diseñado.

3.2.2.2. ClienteX-ClienteY

Una vez obtenidos los datos requeridos por parte del servidor, los clientes se comunican entre sí para formar así el sistema peer-to-peer usando el/los protocolo/s definido/s.

3.2.3. Comportamientos

Tanto los clientes como el servidor han de ejecutarse en máquinas diferentes.





4. Vistas de Descomposición

4.1. General

4.1.1. Presentación de la Vista



Ilustración 2 - Vista de Descomposición General

4.1.2. Catálogo de Elementos

4.1.2.1. Elementos y propiedades

4.1.2.1.1. Cliente

El cliente de la aplicación. Se encarga de gestionar las comunicaciones entre clientes, comunicación con el servidor para realizar búsquedas, descargas y todas las acciones necesarias propias de una aplicación P2P. Es la parte de la aplicación controlada por el usuario y éste será el encargado de realizar todas esas acciones.

4.1.2.1.2. Servidor

El servidor de la aplicación. Guarda toda la información referente a los archivos y clientes conectados en cada momento. Sirve de **nexo de comunicación** entre los clientes de la red de modo que los clientes averiguan lo que necesitan a través del servidor.





4.1.2.2. Relaciones

4.1.2.2.1. Cliente-Servidor

El servidor se comunica con el cliente a través del gestor de red y viceversa cuando ambos necesitan intercambiar información a través de la red.

4.1.3. Guía de Variabilidad

Cada uno de los elementos descritos con anterioridad se comporta como una **caja negra**, es decir, todo lo que atañe a dicho elemento se gestiona de forma independiente por él mismo y proporciona una interfaz al resto de módulos que interactúan con él a modo de abstraer a los mismos de su implementación y contenido interno.

4.1.4. Información sobre la Arquitectura

4.1.4.1. Decisiones de Diseño

Se ha optado por este diseño principalmente porque da una visión global clara y concisa del sistema a desarrollar y dicha subdivisión modular permite una mejor reestructuración de las tareas de cada componente del grupo de trabajo, permitiendo explotar el paralelismo de desarrollo no existente con anterioridad a la especificación de la Arquitectura del Sistema.





4.2. Cliente

4.2.1. Presentación de la Vista

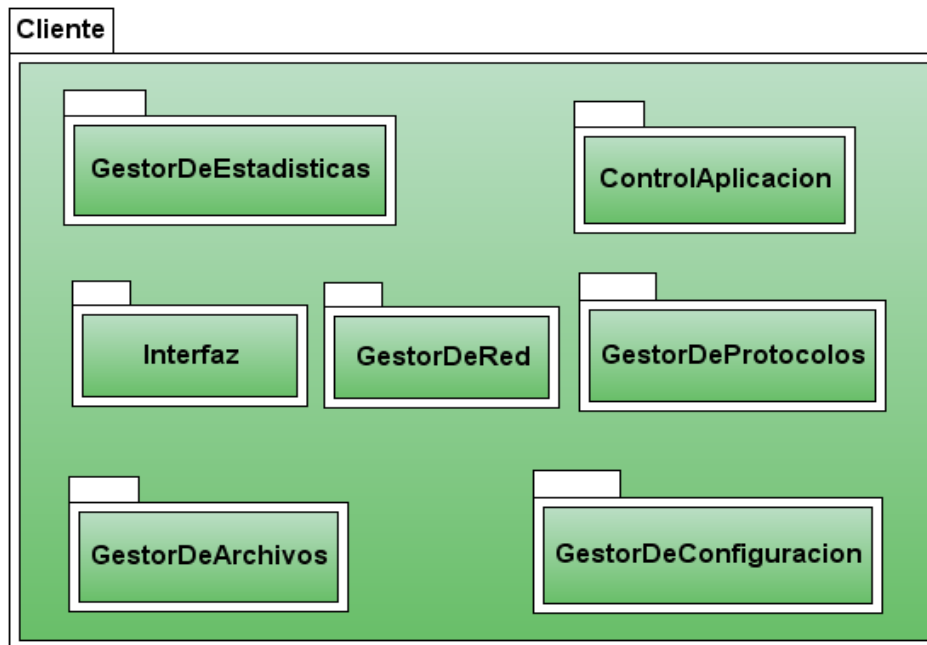


Ilustración 3 - Vista de Descomposición de la aplicación Cliente

4.2.2. Catálogo de Elementos

4.2.2.1. Elementos y propiedades

4.2.2.1.1. Interfaz

Como su propio nombre indica representa la interfaz gráfica de la aplicación. Consiste en una **ventana gráfica** dividida en diferentes pestañas con funcionalidad independiente en cada una de ellas.





4.2.2.1.2. ControlAplicación

Encargado del control de la aplicación, guardando los datos requeridos en **ficheros de propiedades** para un control absoluto y eficaz ante posibles errores del sistema.

4.2.2.1.3. GestorDeEstadísticas

Encargado de la generación de estadísticas en función de los datos extraídos de los diferentes módulos de los que se compone la aplicación. Su manejo **no hará uso de Bases de Datos** ni similares, generando la información a partir de eventos producidos en los diferentes módulos de la aplicación.

4.2.2.1.4. GestorDeArchivos

Encargado de la **fragmentación** y el **reensamblado** de los ficheros manejados en la aplicación.

4.2.2.1.5. GestorDeProtocolos

Divido en dos partes:

4.2.2.1.5.1. Cliente-Servidor

Define los protocolos de comunicación entre el cliente y el servidor. Se pueden distinguir los siguientes subprotocolos:

4.2.2.1.5.1.1. ProtocoloDeConexion

Encargado de definir el protocolo a seguir entre el servidor y el cliente cuando el segundo solicita la conexión al primero.

4.2.2.1.5.1.2. ProtocoloDeDescargas

Encargado de definir el protocolo a seguir entre el servidor y el cliente cuando el segundo solicita la descarga de un archivo.





4.2.2.1.5.1.3. ProtocoloDeBusquedas

Encargado de definir el protocolo a seguir entre el servidor y el cliente cuando el segundo solicita la búsqueda de un archivo al servidor.

4.2.2.1.5.1.4. ProtocoloDeSubidas

Encargado de definir el protocolo a seguir entre el servidor y el cliente cuando el segundo sube (o comparte) un archivo al servidor que puede ser descargado por parte de otro cliente.

4.2.2.1.5.2. Cliente-Cliente

Define los protocolos de comunicación entre dos clientes. Se pueden distinguir los siguientes protocolos:

4.2.2.1.5.2.1. ProtocoloDeDescargas

Encargado de definir el protocolo a seguir entre dos clientes cuando uno le solicita al otro la descarga de un archivo.

4.2.2.1.6. GestorDeRed

Encargado de la comunicación con la red mediante **sockets**. Envía y recibe objetos por la red usando el **protocolo TCP**.

4.2.2.1.7. GestorDeConfiguración

Guarda la configuración de la aplicación, tanto en la parte del servidor como en la del cliente. **Para ello se deberá guardar la información en algún archivo de configuración de properties.**





4.2.2.2. Relaciones

4.2.2.2.1. GestorDeRed-GestorDeProtocolos

El gestor de protocolos y el gestor de red se envían paquetes mutuamente. Cuando se produce actividad por la red la captura el gestor de red y le pasa esa información al gestor de protocolos.

Cuando el gestor de protocolos quiere realizar una comunicación por la red se lo pasará al gestor de red para que éste lo envíe por la red.

4.2.2.2.2. Interfaz-Resto

La interfaz representa de manera visual toda la lógica del resto de paquetes por lo que estará en continua comunicación (instantánea por medio de eventos) con el resto de los módulos para actualizar la diferente información repartida entre las diferentes pestañas.

4.2.2.2.3. ControlAplicación-Resto

El módulo de control al igual que la interfaz estará en continua comunicación con el resto de módulos de la aplicación para actualizar los datos que gestiona y almacenarlos correctamente en el **archivo de control**.

4.2.2.2.4. GestorDeEstadísticas-GestorDeFicheros

El gestor de estadísticas se comunica con el gestor de archivos mediante **sondas** que se encuentran en el módulo del gestor de archivos de tal forma que cuando se produzcan ciertos **eventos** el módulo de estadísticas pueda actualizar su información en **tiempo real**.

4.2.2.2.5. GestorDeEstadísticas-GestorDeProtocolos

El gestor de estadísticas al igual que hace con el gestor de ficheros se comunica con el gestor de protocolos por medio de **sondas** siguiendo el mismo criterio de actualización en **tiempo real**.

4.2.3. Guía de Variabilidad





Cada uno de los elementos descritos con anterioridad se comporta como una caja negra, es decir, todo lo que atañe a dicho elemento se gestiona de forma independiente por él mismo y proporciona una interfaz al resto de módulos que interactúan con él a modo de abstraer a los mismos de su implementación y contenido interno.

4.2.4. Información sobre la Arquitectura

4.2.4.1. Decisiones de Diseño

Se ha optado por este diseño principalmente porque da una visión global clara y concisa del sistema a desarrollar y dicha subdivisión modular permite una mejor reestructuración de las tareas de cada componente del grupo de trabajo, permitiendo explotar el paralelismo de desarrollo no existente con anterioridad a la especificación de la Arquitectura del Sistema.

Este es el primer boceto realizado de la Arquitectura del Sistema. El módulo Control está por definir su ubicación o uso, incluyendo su posible integración en otro de los módulos.

Un posible destino sería en el módulo de **GestorDeProtocolo**, pero fue desechada por la complejidad existente ya de por sí en dicho módulo y en aras de explotar más un poco más la modularidad y el reparto equitativo de trabajo.





4.3. Gestor de Archivos

4.3.1. Presentación de la Vista

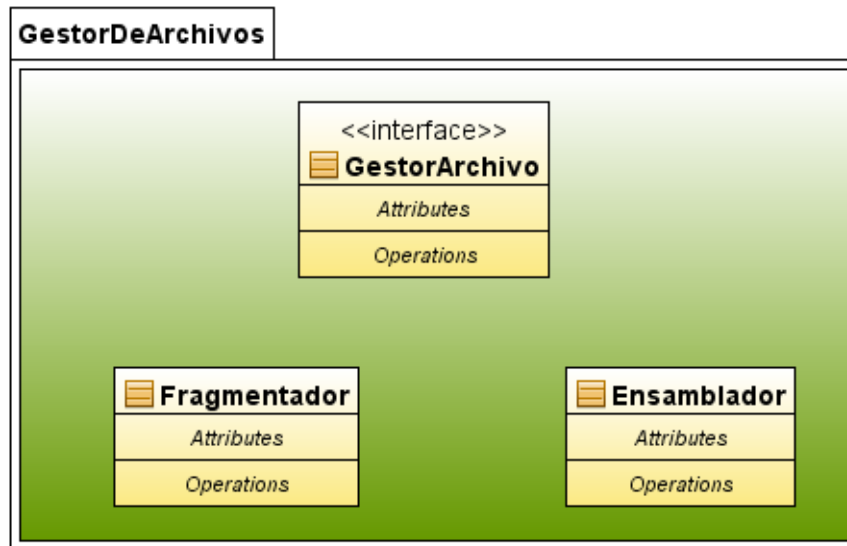


Ilustración 4 - Vista de descomposición del Gestor de Archivos





4.4. Interfaz de la aplicación Cliente

4.4.1. Presentación de la Vista

El módulo Interfaz Gráfica del Programa Cliente se descompone a su vez en los dos siguientes paquetes:

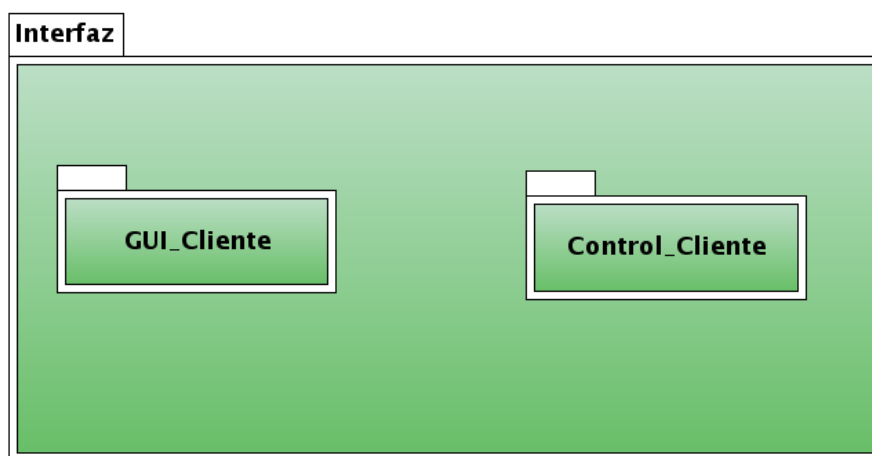


Ilustración 5 - Vista de Descomposición de la aplicación Cliente

4.4.2. Catálogo de Elementos

4.4.2.1. Elementos y propiedades

Se describe a continuación el significado de cada elemento que aparece en la vista de la arquitectura.

4.4.2.1.1. Paquete GUI_Cliente

Encargado de implementar la *interfaz gráfica* propiamente dicha que el usuario del programa cliente va a tener a su disposición para poder manejar la aplicación.





4.4.2.1.2. Paquete Control_Cliente

Este paquete implementa la comunicación de la interfaz gráfica con el resto de la aplicación. Esto se hace así para poder desacoplar el paquete que implementa dicha interfaz del resto de módulos. Así, cualquier modificación en la alguna otra parte, no debería afectar a la GUI_Cliente, y viceversa.

4.4.2.2. Relaciones

Al tratarse de una vista de descomposición, no reflejamos aquí las relaciones existentes entre los paquetes presentados. Consultar para ello el apartado Vista de Uso de la Interfaz Gráfica del Programa Cliente.

4.4.3. Guía de Variabilidad

Cualquier modificación puede ser acometida de manera completamente independiente. Es de esperar que el mayor número de modificaciones recaiga sobre el paquete GUI_Cliente, ya que se trata de la interfaz de uso del programa cliente, y puede estar sujeta a muchas nuevas decisiones por parte de los desarrolladores y usuarios de la aplicación.

Consultar el **Escenario de Calidad del Software** relativo a la **modificabilidad**, y este mismo punto en el apartado **Vista de Uso de Interfaz de la aplicación Cliente** para más información.





4.4.4. Información sobre la Arquitectura

4.4.4.1. Decisiones de Diseño

Se mantiene aquí la idea de modularidad e independencia de desarrollos y modificaciones presentes en toda la aplicación.

Esta descomposición en paquetes de la *interfaz del cliente*, consideramos que se ajusta bien a estos objetivos.

La implementación definitiva de dichos paquetes, es completamente flexible, y si las modificaciones que se proponen, tienen que ver sin embargo más con módulos del resto de la aplicación, de nuevo bastaría con respetar la comunicación del resto de módulos con la interfaz gráfica (a través del paquete Control_Cliente).





5. Vistas de Uso

5.1. Gestor de Archivos

5.1.1. Presentación de la vista

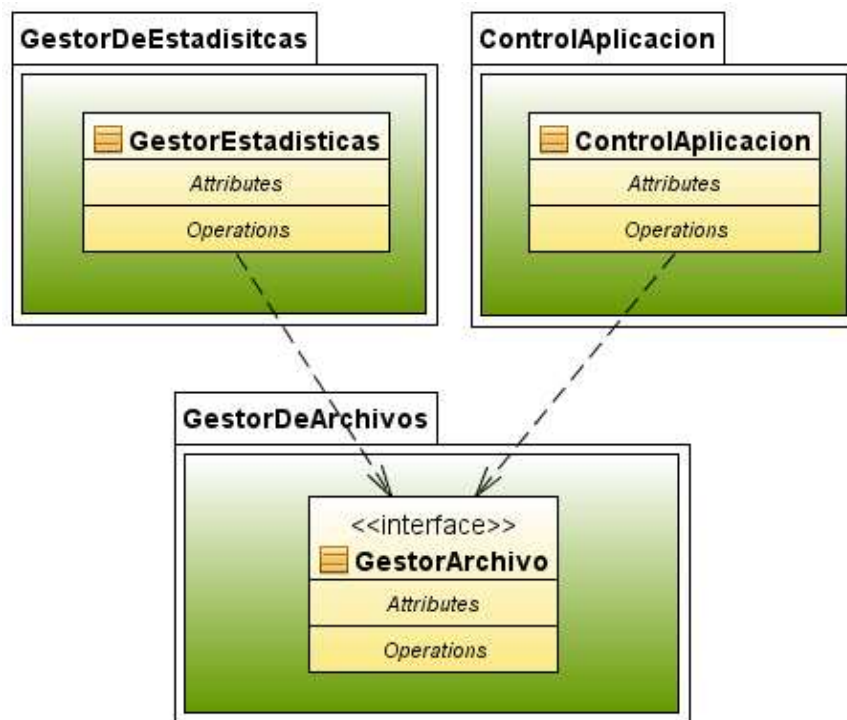


Ilustración 5 - Vista de uso/comunicación con otros módulos.

5.1.2. Catálogo de Elementos

5.1.2.1. Elementos y propiedades

5.1.2.1.1. GestorDeArchivos

Encargado de la **fragmentación** y el **reensamblado** de los ficheros manejados en la aplicación cliente.





5.1.2.1.1. GestorArchivos

Encargado de interactuar con el resto de partes de la aplicación, proporcionando una interfaz de comunicación con entre los ficheros y el resto de módulos de la aplicación que lo necesiten.

5.1.2.1.2. Fragmentador

Encargado de obtener, a partir de un archivo, las partes o fragmentos de las que está compuesto el mismo.

5.1.2.1.3. Ensamblador

Es el encargado de juntar las partes, de un archivo, de las que se dispone para finalmente obtener el archivo original.

5.1.2.2. Relaciones

5.1.2.2.1. GestorDeEstadísticas- GestorDeArchivos

El gestor de estadísticas se comunica con el gestor de archivos mediante **sondas** que se encuentran en el módulo del gestor de archivos de tal forma que cuando se produzcan ciertos **eventos** el módulo de estadísticas pueda actualizar su información en **tiempo real**.

5.1.2.2.2. ControlAplicación- GestorDeArchivos

El control de la aplicación se comunica con el gestor de archivos a través de la interfaz del mismo, para según el caso, obtener o guardar las partes de los archivos, dependiendo de la tarea que se esté llevando a cabo.

5.1.3. Guía de Variabilidad

El elemento base descrito anteriormente se comporta como una **caja negra**, es decir, todo lo que atañe a dicho elemento se gestiona de forma independiente por él mismo y proporciona una interfaz al resto de





módulos que interactúan con el, abstrayendo a los mismos de su implementación.

Por tanto, cualquier forma de variación interna en *GestorDeArchivos* no afectará al módulo que lo utiliza *GestorDeEstadísticas* o *ControlAplicación*, a excepción que se decida modificar algún servicio que utilicen dichos módulos, o que se decida cambiar la interfaz de comunicación entre los mismos.

5.1.4. Información sobre la Arquitectura

5.1.4.1. Decisiones de Diseño

El dispositivo de almacenamiento suele ser el cuello de botella más común en la mayoría de los sistemas, y más cuando se realizan multitud de operaciones simultáneas sobre el disco, cuando por ejemplo, estamos descargando varios archivos a la vez. Debido a la naturaleza de este módulo y su comunicación con el dispositivo de almacenamiento, se ha abstraído al resto de módulos que necesitan de dicho dispositivo, haciendo que todos interactúen con éste módulo.

Se ha pretendido que sea el único medio para interactuar con los archivos en el disco, como forma de asegurar cierta coherencia con la información y utilizar una misma metodología para el tratamiento de los archivos.

En principio, el diseño de este módulo constaba de un sólo elemento, el cual desempeñaba varios roles, aunque de forma poco segura, ya que en principio permitía funcionar con los distintos roles sin ningún tipo de control. Posteriormente, para evitar que esto fuera posible y se usaran formas de fragmentación cuando realmente necesitábamos un ensamblado, se utilizó un control muy básico mediante atributos internos y excepciones en función de estos.

Pero con el fin de simplificar la implementación y su comprensión, finalmente se aplicó el diseño anterior; más organizado, flexible y seguro frente usos de los diversos módulos que se comunican en él.





5.2. GUI de la aplicación Cliente

5.2.1. Presentación de la Vista

A continuación se muestra la Vista de Uso de los módulos en los que se ha dividido la GUI de la **aplicación Cliente**:

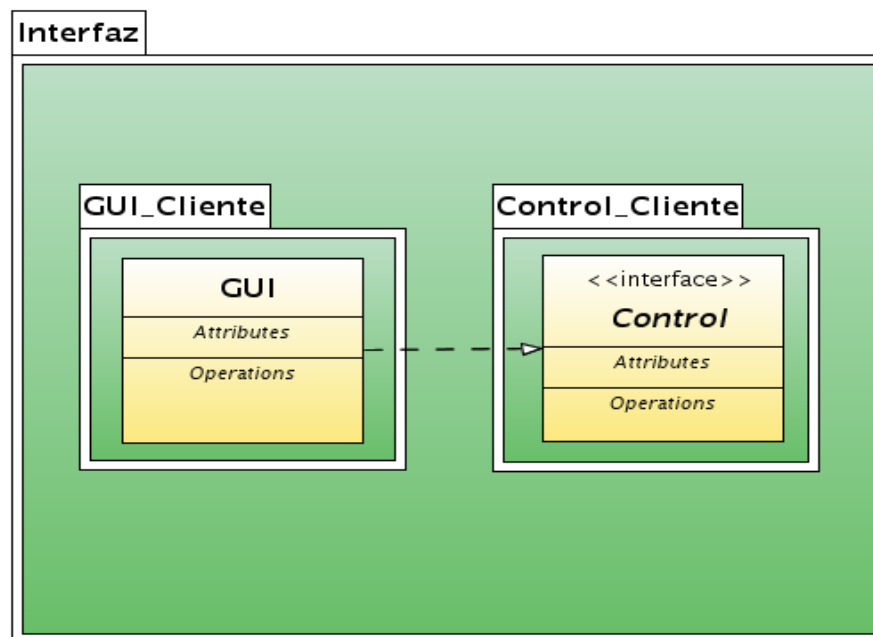


Ilustración 6 - Vista de Uso de la aplicación Cliente

5.2.2. Catálogo de Elementos

5.2.2.1. Elementos y propiedades

Se describe a continuación el significado de cada elemento que aparece en la vista de la arquitectura. Esta descripción coincide con la expuesta en la Vista de Descomposición. No obstante se mantiene también aquí para no tener que desplazarse tanto dentro del documento si se quiere consultar algo.





5.2.2.1.1. **Paquete GUI_Cliente**

Encargado de implementar la *interfaz gráfica* propiamente dicha que el usuario del programa cliente va a tener a su disposición para poder manejar la aplicación.

5.2.2.1.2. **Paquete Control_Cliente**

Este paquete implementa la comunicación de la interfaz gráfica con el resto de la aplicación. Esto se hace así para poder desacoplar el paquete que implementa la interfaz gráfica del resto de módulos. Así, cualquier modificación en la alguna otra parte, no debería afectar a la interfaz, y viceversa.

5.2.2.2. **Relaciones**

Esto es lo más importante en esta vista, y aquí se nos remite desde el mismo punto en el apartado de **Vista de Descomposición de la GUI de la aplicación Cliente**, ya que allí no se muestran las relaciones de uso.

Como se aprecia en la figura, la interfaz gráfica del cliente implementa una **interfaz de servicios** que constituye el módulo del controlador. A través de esta interfaz, se comunica con el resto de la aplicación.

Los servicios proporcionados por la interfaz son los correspondientes a las funciones que se espera puedan realizarse con el cliente:

- **Conexión / Desconexión.**
- **Búsqueda** de fuentes.
- **Gestión de tráfico** (consulta del estado de las descargas, inicio parada y eliminación de las mismas).
- **Configuración de los parámetros del cliente** (velocidades de subida y descarga, ubicación de directorios de almacenamiento de fuentes y compartidos, etc.)
- Solicitud de **información sobre estadísticas.**





5.2.3. Guía de Variabilidad

En el último apartado del punto anterior, se ha visto de manera más concreta cómo la GUI_Cliente hace uso de la implementación de la interfaz que constituye el Control_Cliente para **desacoplarse** del resto de módulos existentes.

Mientras la interfaz gráfica del cliente implemente la interfaz de servicios del Control_Cliente, y el resto de módulos respeten dicha interfaz, se garantiza la posibilidad de modificar de forma independiente al resto de la aplicación, cualquier aspecto de GUI_Cliente. Podría desearse por ejemplo cambiar de una interfaz desarrollada en swing, a una interfaz web, etc... En cualquier caso, siempre se espera que se trate de interfaces de tipo gráfico para el programa cliente.

5.2.4. Información sobre la Arquitectura

5.2.4.1. Decisiones de Diseño

Se ha enfocado el diseño que se ha mostrado en los puntos anteriores de esta *Vista de Uso*, con objeto de satisfacer, en lo que al módulo interfaz se refiere (ver **Vista de Descomposición General**) el **Escenario de Calidad de Software** relativo a cambios en la *GUI_Cliente* (ver apartado de Escenarios de Calidad del Software).





5.3. Protocolo P2P

5.3.1. Presentación de la Vista

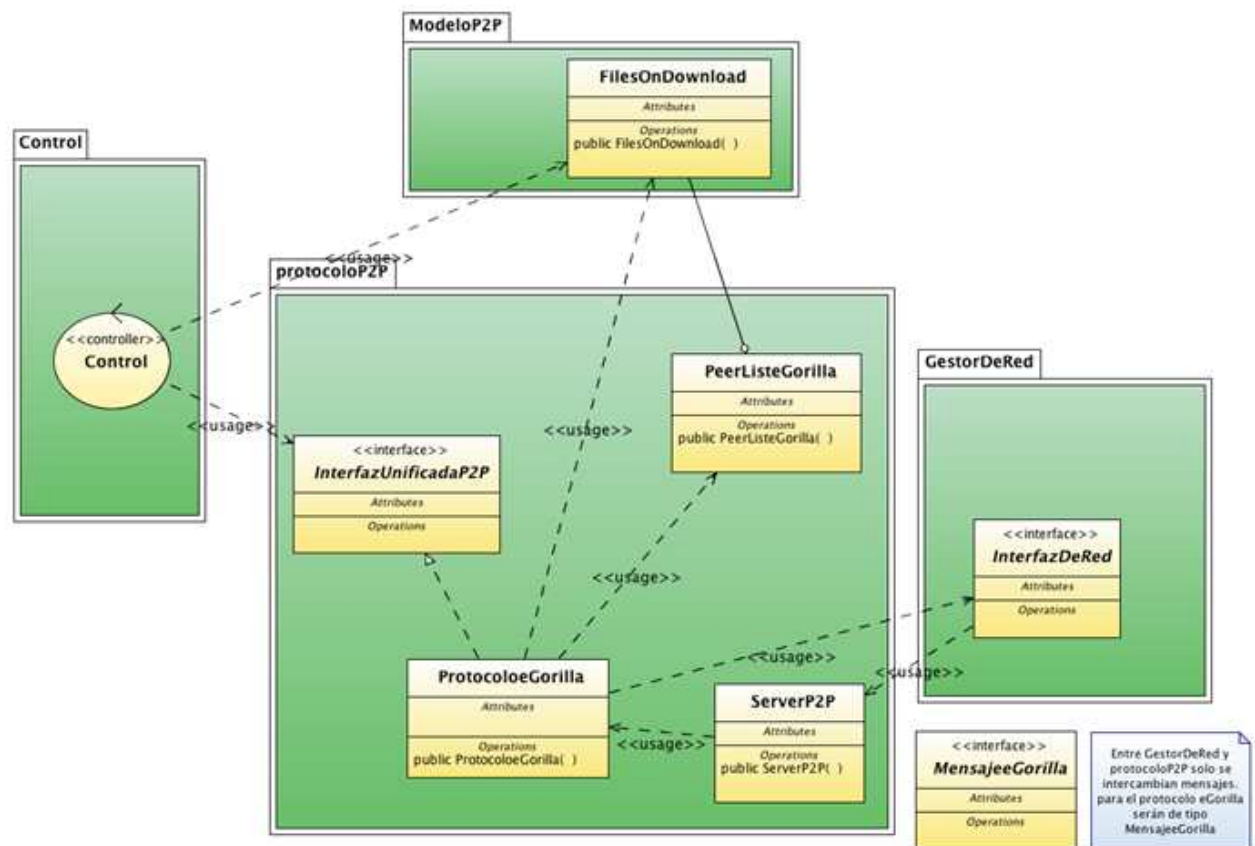


Ilustración 4 - Vista de Uso del Protocolo P2P

5.3.2. Catálogo de elementos

ProtocoloP2P	
Tipo:	Paquete
Descripción:	Contiene la lógica del protocolo p2p, ya sea eGorilla u otro implementado.
Interfaces:	Se comunica con el gestor de red mediante mensajes, recibe ordenes del control y modifica los datos de modeloP2P.



**ModeloP2P**

Tipo:	Paquete
Descripción:	Contiene el estado de nuestra sesión p2p, almacenando el estado individual de cada uno de los ficheros que descargamos y de los que subimos.
Interfaces:	Es modificado por el ProtocoloP2P, el control puede interrogarle para averiguar el estado de los elementos.

GestorDeRed

Tipo:	Paquete
Descripción:	Contiene el bajo nivel de las comunicaciones de red, UDP o TCP.
Interfaces:	Se comunica con el protocoloP2P mediante mensajes.

ProtocoloP2P

Tipo:	Paquete
Descripción:	Contiene la lógica del protocolo p2p, ya sea eGorilla u otro implementado.
Interfaces:	Se comunica con el gestor de red mediante mensajes, recibe ordenes del control y modifica los datos de modeloP2P





ProtocoloP2P	
Tipo:	Paquete
Descripción:	Contiene la lógica del protocolo p2p, ya sea eGorilla u otro implementado.
Interfaces:	Se comunica con el gestor de red mediante mensajes, recibe ordenes del control y modifica los datos de modeloP2P

MensajeeGorilla	
Tipo:	Interfaz
Descripción:	Es un mensaje de los diferentes que puede haber en el protocolo p2p.
Interfaces:	Sirve de comunicación entre el protocoloP2P y el modulo de red.

InterfazUnificadaP2P	
Tipo:	Interfaz
Descripción:	Interfaz de ordenes para interactuar con cualquier protocolo p2p.
Interfaces:	Recibe ordenes del control de la aplicación. Convierte estas en un estado y comienza mecanismos para satisfacerlas.

ProtocoloeGorilla	
Tipo:	Clase-implementación
Descripción:	Satisface la interfazUnificadaP2P para poder comunicarnos con otros clientes y con servidores eGorilla.
Interfaces:	Recibe ordenes del Control y las resuelve mediante el protocolo eGorilla.





PeerListeGorilla	
Tipo:	Clase
Descripción:	Lista los clientes con los que tenemos algún tipo de transito. Ya sea cliente-servidor o servidor cliente. Para cada cliente se le vinculará con los archivos involucrados.
Interfaces:	Lo usa el ProtocoloeGorilla para almacenar los extremos de las transferencias eGorilla.

FilesOnDownload	
Tipo:	Clase Almacenamiento.
Descripción:	Conservará el estado de todos los archivos con los que se relaciona este cliente, tanto los compartidos como los que están en proceso de descarga. Será capaz de ser almacenado de forma no volátil para continuar en diferentes ejecuciones.
Interfaces:	Se comunica con ProtocoloeGorilla, control y con los peer almacenados en PeerListeGorilla

5.3.3. Información sobre la arquitectura

5.3.3.1. Elementos y propiedades

La arquitectura parece un poco complicada para lo que podría ser el sistema, pero de esta manera nos aseguramos la compatibilidad con futuros cambios.

El diseño se ha hecho pensando en transferencias aisladas y no en conexiones establecidas, de forma que se permite el uso de otros protocolos para poder utilizar otras técnicas.

Además se añade la posibilidad de añadir compatibilidad con otras





arquitecturas de p2p al solo tener que implementar la **interfazUnificadaP2P**.

5.3.4. Guía de Variabilidad

El protocolo aquí reflejado hace referencia **exclusivamente** al protocolo eGorilla, de forma que se estableciera una interfaz de comunicación con el resto de paquetes y control de aplicación generico. Para que pueda ser añadida compatibilidad con cualquier otro protocolo.

Añadir nuevos diálogos al protocolo no deberá modificar el comportamiento de los ya establecidos, ya que cada uno de los mensajes será procesado de forma independiente.

5.3.5. Vistas relacionadas

5.3.5.1. Vista Cliente Servidor

5.3.5.2. Vista de Procesos Comunicados





6. Vista de Datos Compartidos

6.1. Presentación de la Vista

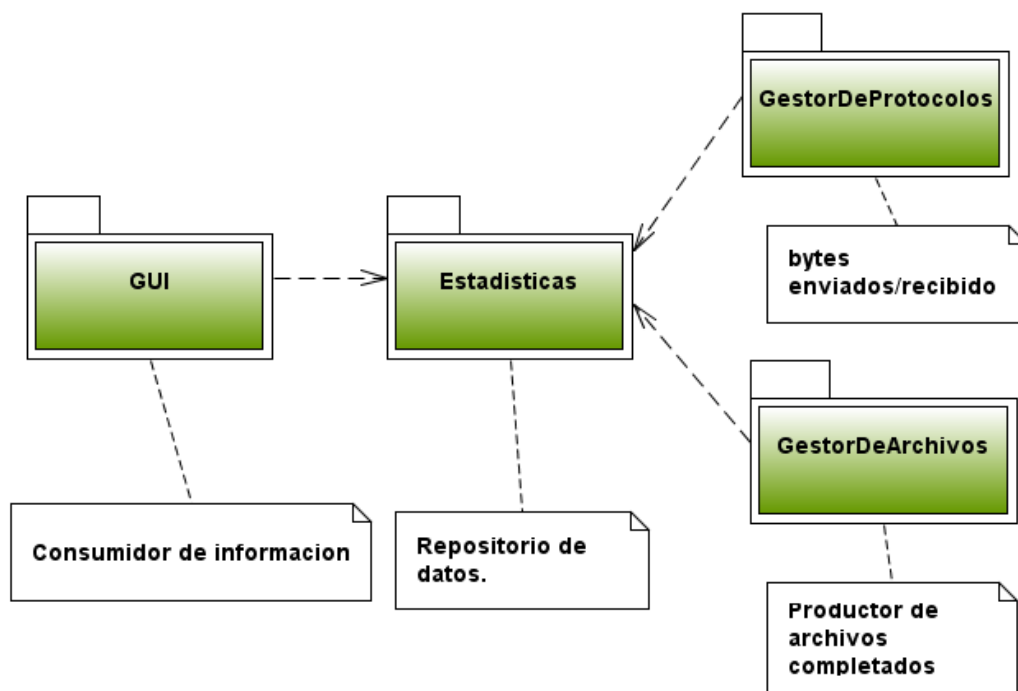


Ilustración 7 - Vista de Datos Compartidos





6.2. Catálogo de Elementos

6.2.1. Elementos y propiedades

6.2.1.1. GUI

Este componente hace de consumidor de los datos que están disponibles en el módulo de estadísticas, este los solicita cada vez lo necesite y les da un formato gráfico para dar mejor información al usuario.

6.2.1.2. GestorDeEstadisticas

El componente GestorDeEstadisticas, funciona de subsistema para almacenamiento persistente de la información que le llega a través del gestor de archivos y del gestor de protocolos, ordenando y proporcionándosela a la GUI con diversos formatos. También hace función de repositorio ya que no avisa al consumidor o consumidores. Esto es producido porque los datos están disponibles en cualquier momento que el consumidor quiera obtenerlos.

6.2.1.3. GestorDeRed

Este componente se encarga de enviar y recibir los datos entre los distintos clientes de la red y también con el servidor.

6.2.1.4. GestorDeProtocolos

Este componente se encarga de almacenar los datos recibidos en el soporte físico encargándose que los datos sean coherentes y correctos.

6.2.2. Relaciones

6.2.2.1. GestorDeEstadisticas- GestorDeProtocolos

El gestor de protocolos es el que realiza la función de productor de datos, éste almacena en el gestor de estadística todos los bytes que se envía y se reciben y así evita perder toda la información.





6.2.2.2. *GestorDeEstadisticas-GestorDeArchivos*

El gestor de archivos realiza la función de productor de datos, éste almacena en el gestor de estadísticas todos los archivos que se completan, evitando así perder toda la información.

6.2.2.3. *Interfaz-GestorDeEstadisticas*

La interfaz realiza la función de consumidor, obteniendo los datos a partir del gestor de estadísticas.

6.3. *Guía de Variabilidad*

Con esta implementación se permite que para la función de gestión de datos informativos no se conozcan directamente por lo que facilita la independencia entre ellos, además permite añadir componentes que accedan a la información sin que se realice ninguna modificación sobre los módulos anteriores.

6.4. *Información sobre la Arquitectura*

6.4.1. *Decisiones de Diseño*

Se ha optado por este diseño porque permite una fácil gestión de los datos estadísticos adapta a la visión modular del sistema global, evita una saturación innecesaria de los productores como hubiese ocurrido en un sistema productor-consumidor y además facilita la adición de otros módulos que querrán tener acceso a los datos. También facilita añadir nuevas funcionalidades sobre el manejo y tratamiento de los datos estadísticos ya que están encapsulados en el módulo de **gestorDeEstadísticas**.





7. Vista de Cliente-Servidor

7.1. Cliente como Servidor

7.1.1. Presentación de la Vista

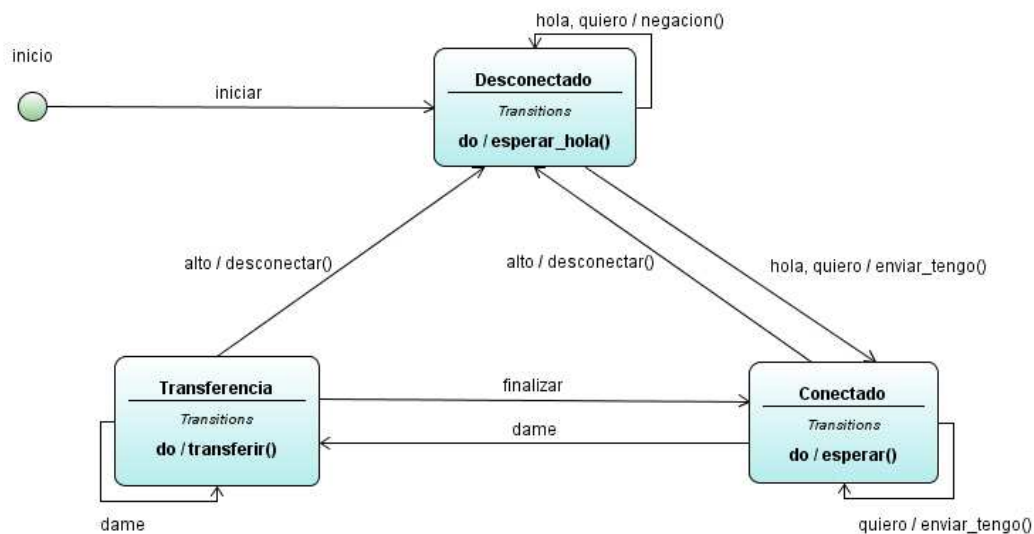


Diagrama de estados 1. Cliente como Servidor

En el diagrama de estados “Cliente como Servidor” se muestran los diferentes estados por los que puede pasar la aplicación cliente del eGorilla en modo servidor, a la hora de interactuar con otro cliente del eGorilla en modo cliente.





7.1.2. Catálogo de elementos

7.1.2.1. Elementos y propiedades

7.1.2.1.1. Desconectado

Estado inicial del cliente eGorilla en modo servidor. En este estado, el servidor estará a la espera de conexiones entrantes de clientes que quieren descargar algún archivo. Si el servidor puede atender al cliente, le enviará las partes del archivo deseado que tiene del fichero que se quiere descargar. En el caso de no poder atender la petición del cliente por haberse alcanzado el número máximo de usuarios conectados, se enviará la negación de conexión al cliente y se procederá a su desconexión.

7.1.2.1.1.1. Transiciones

7.1.2.1.1.1.1. hola, quiero

Transición producida por un cliente entrante donde nos indica que archivo desea que le envíe el servidor. Si el servidor es capaz de atender su petición, enviará las partes del archivo que desea mediante la acción `enviar_tengo()`. En caso contrario negará la petición de conexión y cerrará la conexión con el cliente mediante la acción `negación()`.

7.1.2.1.1.1.2. alto

Transición producida por un cliente conectado para indicar al servidor que no necesita más de los servicios del servidor. El servidor cerrará la conexión con el cliente y se quedará en el estado "Desconectado" a la espera de nuevas peticiones con la acción `esperar_hola()`.

7.1.2.1.2. Conectado

Estado en el que el servidor se encuentra conectado con un cliente y a la espera de poder atender a sus peticiones. El servidor en este estado puede pasar al estado de transferencia para comenzar el envío de las partes del fichero deseado; puede desconectarse del cliente por no necesitar ningún servicio más del servidor o bien puede indicar las partes de algún archivo que tiene manteniéndose en el mismo estado "Conectado".





7.1.2.1.2.1. Transiciones

7.1.2.1.2.1.1. quiero

Transición producida por un cliente donde nos indica que archivo desea que le envíe el servidor. Dado que la conexión esta ya establecida el servidor atenderá su petición y enviará las partes del archivo que desea mediante la acción `enviar_tengo()`.

7.1.2.1.2.1.2. dame

Transición producida por un cliente donde nos solicita las partes del archivo que desea. En este caso el servidor pasará al estado "Transferencia" para enviar las partes del archivo deseado por el cliente.

7.1.2.1.2.1.3. alto

Transición producida por un cliente conectado para indicar al servidor que no necesita más de los servicios del servidor. El servidor cerrará la conexión con el cliente y se quedará en el estado "Desconectado" a la espera de nuevas peticiones con la acción `esperar_hola()`. El servidor también puede producir la transición "alto" en el caso de llevar un tiempo "t" sin actividad con el cliente conectado.

7.1.2.1.2.1.4. finalizar

Transición producida por el servidor en el estado de "Transferencia" y que hacen entrar al servidor en el estado de "Conectado". Esta transición se produce al terminar de enviar las partes del archivo solicitadas por el cliente o bien al producirse un error por solicitar una parte no disponible en el servidor. Esta transición también puede producirse por el cliente para solicitar nuevas partes al servidor.

7.1.2.1.3. Transferencia

Estado en donde el servidor transfiere las partes del archivo deseado por el cliente. En este estado, el cliente puede solicitar la transferencia de las partes solicitadas. Si el cliente no necesita de los servicios del servidor puede solicitar la desconexión con el mismo. En el caso de no producirse ninguna transferencia al cliente, el servidor pasará a modo "Conectado".





7.1.2.1.3.1. Transiciones

7.1.2.1.3.1.1. dame

Transición producida por un cliente donde nos solicita que le enviemos una nueva parte del archivo solicitado. La transferencia de las partes se realizará de forma aleatoria para mejorar la eficiencia de las transmisiones entre clientes.

7.1.2.1.3.1.2. alto

Transición producida por un cliente conectado para indicar al servidor que no necesita más de los servicios del servidor. El servidor cerrará la conexión con el cliente y se quedará en el estado “Desconectado” a la espera de nuevas peticiones con la acción esperar_hola().

7.1.2.1.3.1.3. finalizar

Transición producida por el servidor en el estado de “Transferencia” y que hacen entrar al servidor en el estado de “Conectado”. Esta transición se produce al terminar de enviar las partes del archivo solicitadas por el cliente o bien al producirse un error por solicitar una parte no disponible en el servidor. Esta transición también puede producirse por el cliente para solicitar nuevas partes al servidor. El servidor también puede producir la transición “finalizar” en el caso de llevar un tiempo “t” sin que el cliente haya solicitado nuevas transferencias.





7.2. Cliente como Cliente

7.2.2. Presentación de la vista

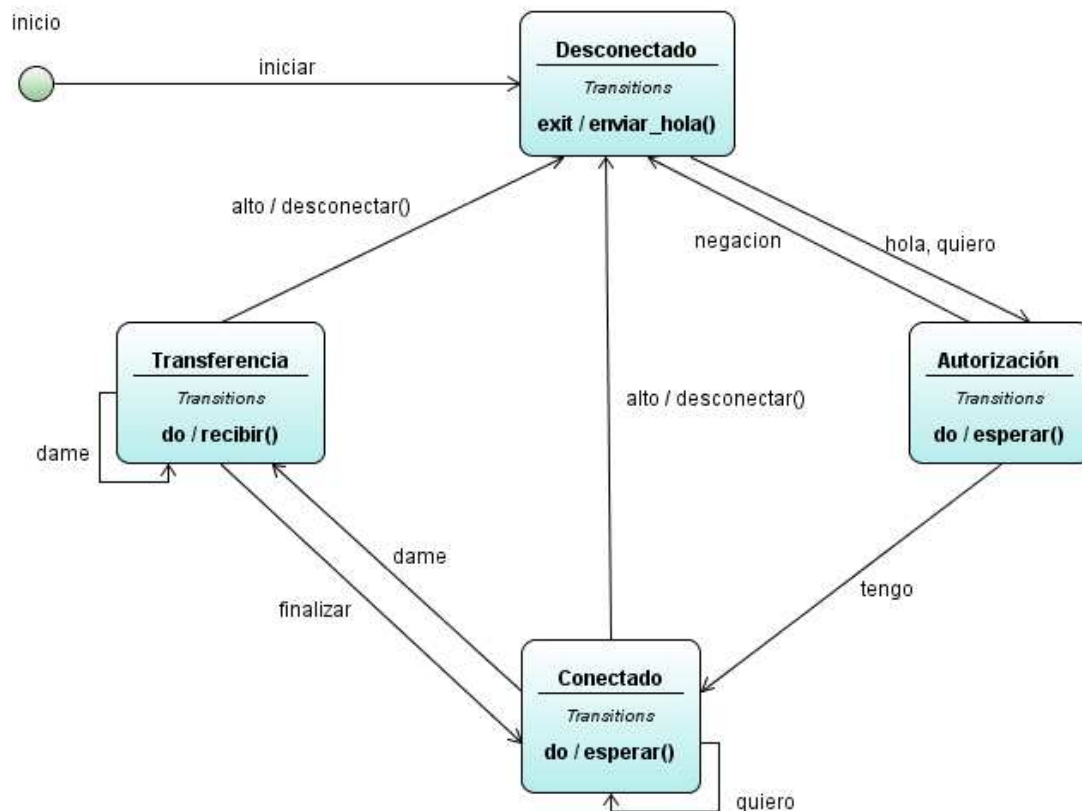


Diagrama de estados 2. Cliente como cliente





En el diagrama de estados “Cliente como Cliente” se muestran los diferentes estados por los que puede pasar la aplicación cliente del eGorilla en modo cliente, a la hora de interactuar con otro cliente del eGorilla en modo servidor.

7.1.3. Catálogo de elementos

7.1.3.1. Elementos y propiedades

7.1.3.1.1. Desconectado

Estado inicial del cliente eGorilla en modo cliente. En este estado, el cliente podrá solicitar una petición de conexión al servidor para descargar las partes del archivo deseado mediante la acción enviar_hola().

7.1.3.1.1.1. Transiciones

7.1.3.1.1.1.1. hola, quiero

Transición producida por el cliente, mediante la acción enviar_hola(), donde se indica que archivo desea que le envíe el servidor. Una vez producida está transición el cliente pasará al estado “Autorización”.

7.1.3.1.1.1.2. negación

Transición producida por el servidor donde se rechaza la conexión al mismo debido a un cierto motivo indicado. El cliente pasa a estado “Desconectado”.

7.1.3.1.1.1.3. alto

Transición producida por un cliente conectado para indicar al servidor que no necesita más de los servicios del servidor. El servidor cerrará la conexión con el cliente y se quedará en el estado “Desconectado” a la espera de realizar nuevas conexiones mediante la acción enviar_hola().





7.1.3.1.2. Autorización

Estado de espera del cliente para conectarse a un servidor. En este estado, el servidor comprueba si puede atender al cliente o si por el contrario se rechaza la solicitud de conexión. En el caso de poder atender la conexión del cliente se le enviarán las partes del archivo deseado que tiene del fichero que se quiere descargar. En el caso de no poder atender la petición del cliente por haberse alcanzado el número máximo de usuarios conectados, se enviará la negación de conexión al cliente y se procederá a su desconexión.

7.1.3.1.2.1. Transiciones

7.1.3.1.2.1.1. hola, quiero

Transición producida por un cliente entrante donde nos indica que archivo desea que le envíe el servidor. Si el servidor es capaz de atender su petición, enviará las partes del archivo que desea mediante la acción `enviar_tengo()`. En caso contrario negará la petición de conexión y cerrará la conexión con el cliente mediante la acción `negación()`.

7.1.3.1.2.1.2. negación

Transición producida por el servidor donde se rechaza la conexión al mismo debido a un cierto motivo indicado. El cliente pasa a estado "Desconectado".

7.1.3.1.2.1.3. tengo

Transición producida por el servidor donde se indica al cliente las partes del archivo deseado que dispone. Con esta transición indicamos la disponibilidad del servidor a atender nuestra petición haciéndole pasar al estado de "Conectado".





7.1.3.1.3. Conectado

Estado en el que el cliente se encuentra conectado con un servidor que está a la espera de poder atender a sus peticiones. El cliente en este estado puede solicitar al servidor: pasar al estado de transferencia para comenzar el envío de las partes del fichero deseado; desconectarse del cliente por no necesitar ningún servicio más del servidor o bien realizar nuevas consultas de partes de archivos deseadas.

7.1.3.1.3.1. Transiciones

7.1.3.1.3.1.1. quiero

Transición producida por el cliente donde se indica que archivo desea que le envíe el servidor. Dado que la conexión esta ya establecida el servidor atenderá su petición mientras que el servidor espera su respuesta mediante la acción esperar().

7.1.3.1.3.1.2. dame

Transición producida por el cliente donde se solicita las partes del archivo que desea. En este caso el cliente pasará al estado "Transferencia" para recibir las partes del archivo deseado del servidor.

7.1.3.1.3.1.3. alto

Transición producida por el cliente para indicar al servidor que no necesita más de sus servicios. El servidor cerrará la conexión con el cliente y le hará pasar al estado "Desconectado" a la espera de realizar nuevas conexiones con la acción enviar_hola().

7.1.3.1.3.1.4. finalizar

Transición producida por el servidor en el estado de "Transferencia" que hace entrar al cliente en el estado de "Conectado". Esta transición se produce al terminar de enviar las partes del archivo solicitadas por el cliente o bien al producirse un error por solicitar una parte no disponible en el servidor. Esta transición también puede producirse por el cliente para solicitar nuevas partes al servidor.





7.1.3.1.4. Transferencia

Estado en donde el cliente recibe las partes del archivo deseado desde el servidor. En este estado, el cliente puede solicitar la transferencia de las partes solicitadas. Si el cliente no necesita de los servicios del servidor puede solicitar la desconexión con el mismo. En el caso de querer realizar nuevas peticiones el cliente pasara al estado "Conectado" mediante la transición finalizar.

7.1.3.1.4.1. Transiciones

7.1.3.1.4.1.1. dame

Transición producida por el cliente donde se solicita al servidor el envío de una nueva parte del archivo solicitado. El cliente recibirá las partes solicitadas de manera aleatoria para mejorar la eficiencia de las transmisiones entre clientes.

7.1.3.1.4.1.2. alto

Transición producida por el cliente para indicar al servidor que no necesita más de sus servicios. El servidor cerrará la conexión con el cliente y le hará pasar al estado "Desconectado" a la espera de realizar nuevas conexiones con la acción enviar_hola().

7.1.3.1.4.1.3. finalizar

Transición producida por el servidor en el estado de "Transferencia" que hace entrar al cliente en el estado de "Conectado". Esta transición se produce al terminar de enviar las partes del archivo solicitadas por el cliente o bien al producirse un error por solicitar una parte no disponible en el servidor. Esta transición también puede producirse por el cliente para solicitar nuevas partes al servidor.





7.3. Protocolo P2P

7.3.2. Presentación de la Vista

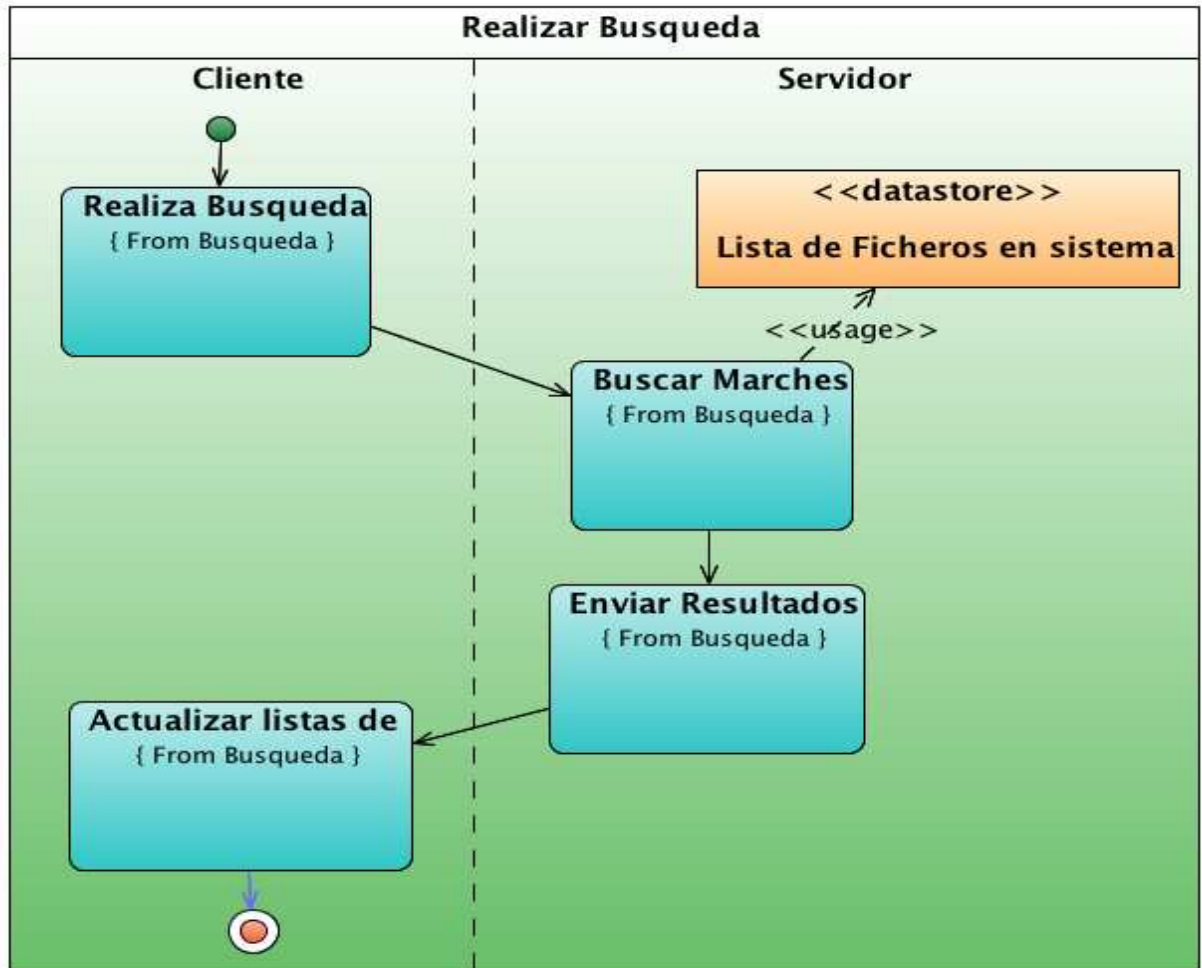


Ilustración 8 – Realizar Búsqueda



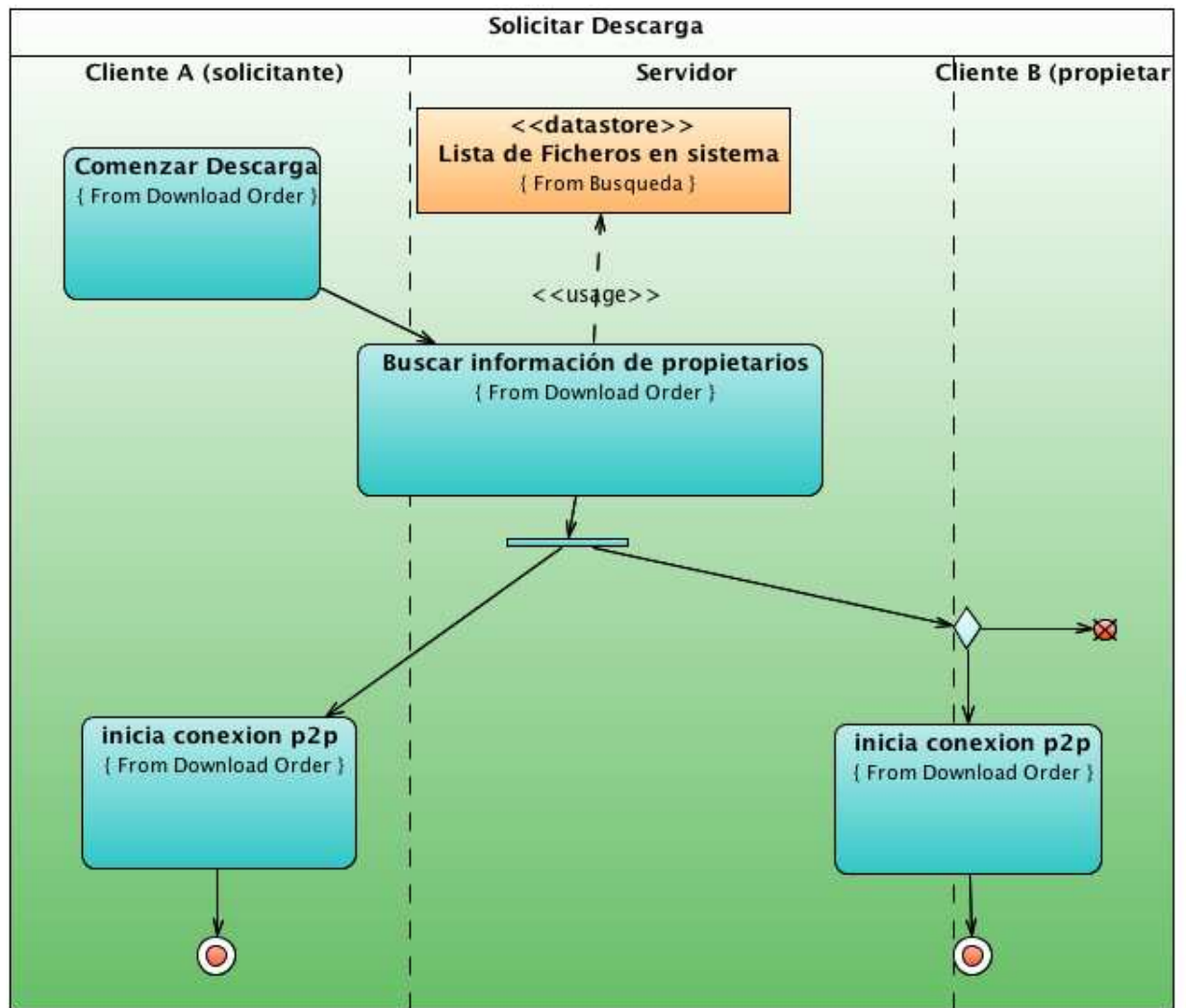


Ilustración 9 – Solicitar Descarga





7.3.3. Catálogo de elementos:

Realizar Búsqueda	
Tipo	Diagrama
Ubicación	Cliente y servidor, interacción entre ambos
Descripción	El cliente conectado al servidor solicita una búsqueda. Esta se comunica y el servidor la resuelve.

Solicitar Descarga	
Tipo	Diagrama
Ubicación	Cliente y servidor, implica comunicación entre ambos
Descripción	EL cliente solicita la descarga de un archivo concreto, se comunica al servidor para que este pueda recuperar la lista de propietarios que tienen este fichero.

Realizar Búsqueda	
Tipo	Actividad
Ubicación	Cliente
Descripción	El usuario acaba de rellenar el formulario de descarga y la valida. Se construye un mensaje de tipo PeticionBusqueda y se envía al servidor por la conexión establecida





Buscar Matches	
Tipo	Actividad
Ubicación	Servidor
Descripción	El Servidor realiza una búsqueda de todas las coincidencias para la frase de búsqueda entre los nombres de los ficheros que hay dados de alta en el sistema. Después se filtran los que no cumplan con las condiciones de filtro

Enviar Resultados	
Tipo	Actividad
Ubicación	Servidor
Descripción	Se empaquetan las coincidencias de la búsqueda o "matches" en un mensaje de tipo RespuestaBúsqueda y se envía al cliente que realizó la búsqueda.

Actualizar listas de búsquedas	
Tipo:	Actividad
Ubicación	Cliente
Descripción	Se invalida la pestaña de búsquedas y se actualiza el contenido de la tabla que se muestra al usuario.

Lista de ficheros del Sistema	
Tipo	Almacenamiento de datos
Ubicación	Servidor
Descripción	Los archivos dados de alta en el sistema se almacenan en aquí de forma volátil. junto con cada fichero identificado con un hash, se almacena la lista de clientes que tienen dicho fichero.



**Comenzar descarga**

Tipo	Actividad
Ubicación	Cliente
Descripción	El cliente selecciona un archivo para comenzar la descarga. Se crea un mensaje PeticionDescarga y se envía al servidor.

Buscar información de propietarios

Tipo	Actividad
Ubicación	Servidor
Descripción	Se recupera una lista con datos de los propietarios de un fichero especificado. Entre estos se encuentran la dirección IP y el puerto de escucha. Se crea un mensaje de tipo RespuestaDescarga, este se envía de vuelta al cliente que solicitó la información y se envía también a cada uno de los propietarios que aparecen en él.

Inicia conexionP2P

Tipo	Actividad
Ubicación	Cliente
Descripción	Si hemos recibido un paquete RespuestaDescarga como consecuencia de una solicitud, comenzaremos la descarga de este con los diferentes clientes propietarios poniéndonos en contacto con estos. Si por el contrario no hemos solicitado dicho archivo, entonces somos propietarios de este y comenzaremos la transferencia en caso de que el solicitante no lo haya hecho ya. Si es un archivo que estamos descargando, podremos actualizar la lista de propietarios de este con el contenido del mensaje





7.3.4. *Guía de variabilidad*

El protocolo aquí dibujado hace referencia **exclusivamente** al protocolo eGorilla, de forma que se establecerá una interfaz de comunicación con el resto de paquetes y control de aplicación genérico. Para que pueda ser añadida compatibilidad con cualquier otro protocolo.

Añadir nuevos diálogos al protocolo no deberá modificar el comportamiento de los ya establecidos, ya que cada uno de los mensajes será procesado de forma independiente.





8. Vista de Procesos Comunicados

8.2. Mensajes

En la siguiente imagen se muestran los distintos mensajes que se pasaran entre clientes y el cliente con el servidor. Cada clase implementa la interfaz de Mensaje para así realizar el envío con un solo nombre.

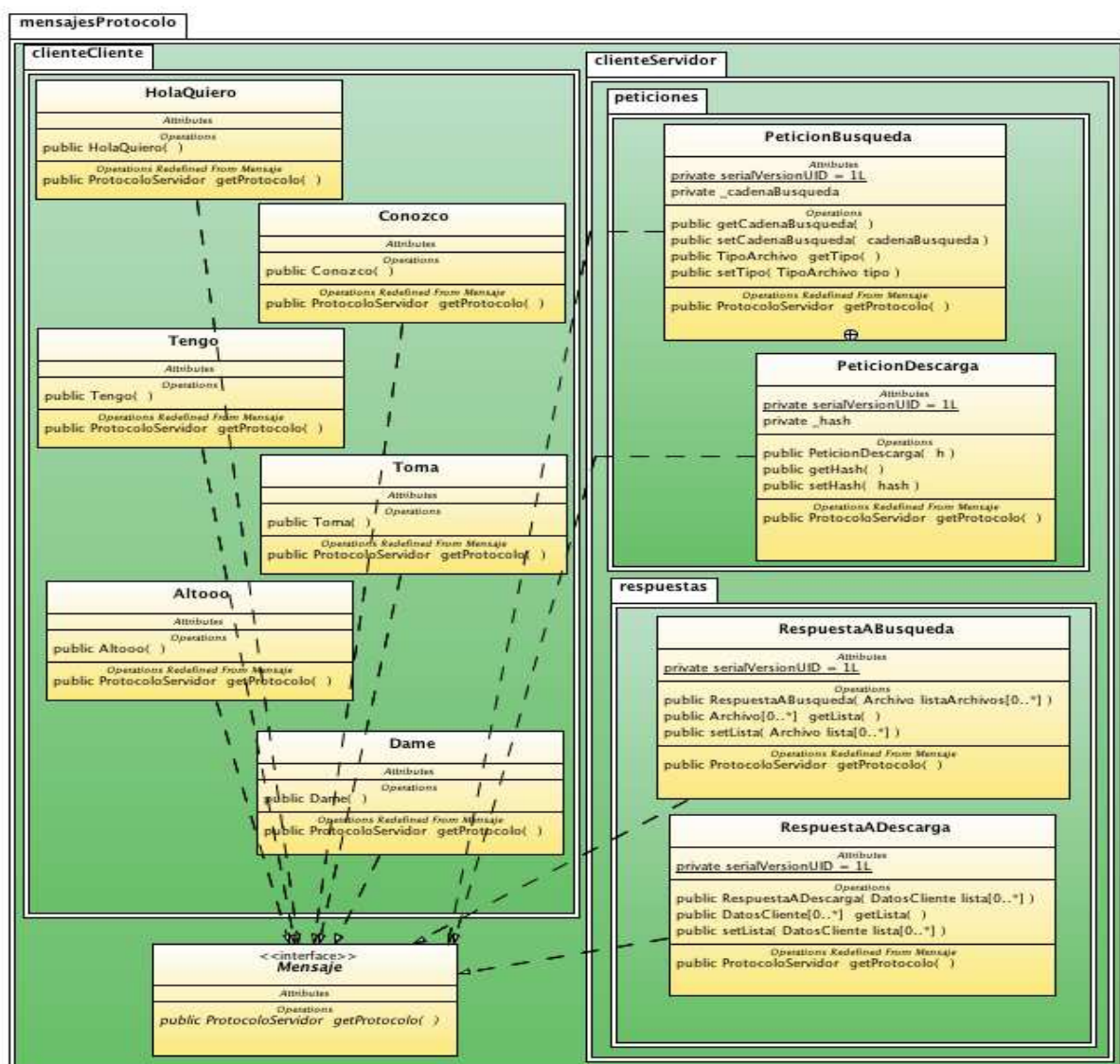


Ilustración 1 - Paquetes y clases de los mensajes





8.3. *Envío Cliente-Servidor*

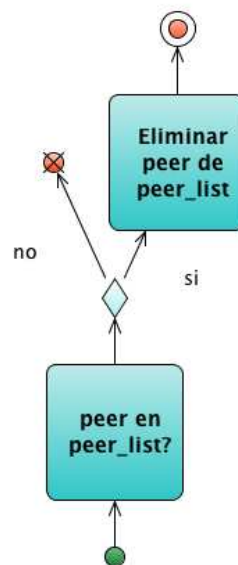
El servidor estará escuchando en un puerto, se establecerá una conexión TCP para atender al usuario. Tras atender la petición del usuario se cerrará el Socket abierto para la conexión.

8.4. *Envío Cliente-Cliente*

El cliente que enviará el archivo, así como cualquier otro, tendrá un puerto de escucha para las peticiones ya que cada cliente se comporta a su vez como servidor. Cuando otro cliente quiera algo de éste, se establecerá una conexión TCP entre ambos y al finalizar el envío de la(s) parte(s) correspondiente(s) se cerrará dicha conexión.

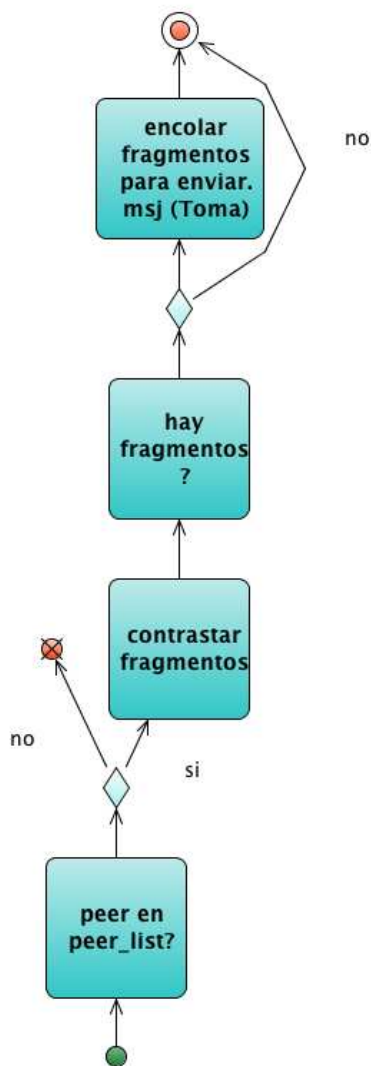
8.5. *Traza de las peticiones*

Cada cliente en modo servidor tiene un ServerP2P que se encargará de generar threads (hijos) que atiendan los mensajes. Cada hijo podrá a su vez generar otros dependiendo de la rama o camino que se tome.

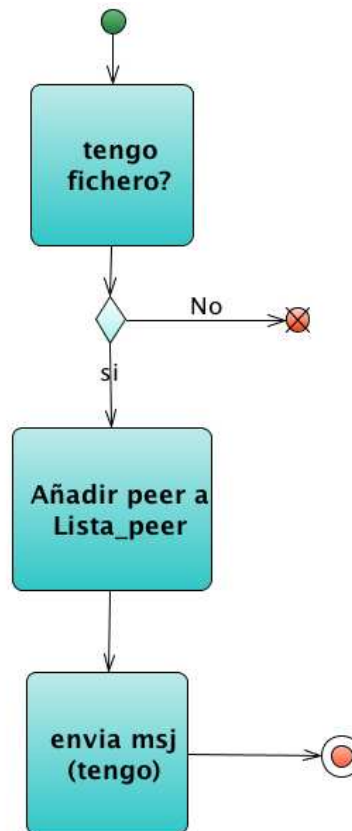


5 - Altooo





6 - Dame



3 - holaquero



