Universidad Complutense

Facultad de Informática

eGorilla

Diseño del Sistema

Asignatura: ***Ingeniería del Software***

Curso Académico: ***2008/2009***

Grupo: ***4º B***

Índice

1. Introducción

En este documento se procede a la descripción del **diseño** del sistema, una vez revisada casi por completo la etapa de **análisis** (siempre habrá algo por revisar correspondiente a ésta etapa) siguiendo el orden establecido por el estándar del **Proceso Unificado de Desarrollo** ó **RUP**.

El contenido del presente documento se puede clasificar en los siguientes conceptos:

- **Contratos de operaciones:** Se analizan y describen las operaciones más críticas del sistema.

- **Casos de uso reales:** Se analizan y describen las acciones reales producidas por el sistema mediante diagramas de casos de uso en los que se refleja precisamente dicho comportamiento real. Un claro ejemplo para ilustrar la idea al lector es el funcionamiento de la interfaz de la aplicación en la que un caso de uso concreto sería **pulsarBotónServidores**.

- **Diagramas del diseño**: Se describen los diagramas correspondientes al diseño, principalmente **diagramas de clases**, de actividad ó secuencia y de estados.

- **Diagrama de Clases de Comunicación entre Módulos**: En éste diagrama se ilustran la clase ó clases dentro de cada módulo que son las encargadas de la comunicación con el resto de los módulos de la aplicación.

2. Contratos de operaciones

2.1. Introducción

Los contratos de operaciones contribuyen a definir el comportamiento de un sistema en términos de los cambios de estado que éste experimenta cuando se invoca una operación. Éste no profundiza en detalles de funcionalidad asociadas a las operaciones invocadas sino que interpreta al sistema como una caja negra.

Éste documento estará redactado declarativamente expresándose a partir de los cambios de estado de las precondiciones y las poscondiciones.

Se describirán las operaciones de mayor relevancia para la funcionalidad principal de la aplicación, éstas estarán condicionadas por el diseño del modelo del dominio y los diagramas de secuencia del sistema.

2.2. Cliente

2.2.1. Contrato para caso de uso conectarServidor

Operación	**conectarServidor ()**
Responsabilidades	Establece la conexión con el servidor, se identifica en él e informa de los archivos compartidos.
Tipo	Sistema
Referencias Cruzadas	Casos de uso: BuscarArchivos, IniciarDescarga, DesconectarServidor.
Excepciones	Si la versión del cliente no es válido, indicar que hay un error de versión.
Salida	Información al usuario mediante mensaje de información.
Precondiciones	No existe una conexión activa con ningún servidor.
Postcondiciones	• Fue creada una instancia de ServidorCliente. • Fue creada una cola de peticiones de InicioDeDescarga. • Se inicializó estadísticas de sesión. • La conexión con el servidor quedó establecida.

2.2.2. Contrato para caso de uso *buscarArchivos*

Operación	buscarArchivos (nombre, tipo)
Responsabilidades	Realiza una búsqueda de los archivos que están en la red que cumplan los requisitos de los parámetros.
Tipo	Sistema
Referencias Cruzadas	
Excepciones	Si no tiene establecida una conexión con el servidor se notificará al usuario con un error.
Salida	Información al usuario mediante una tabla con los archivos resultantes de la búsqueda.
Precondiciones	Se ha establecido conexión con el servidor.
Postcondiciones	Se creó una lista de archivos coincidentes a los parámetros buscados.

2.2.3. Contrato para caso de uso *iniciarDescarga*

Operación	inciarDescarga(identificador)
Responsabilidades	Realiza una serie de peticiones para descargar el archivo solicitado.
Tipo	Sistema
Referencias Cruzadas	
Excepciones	Archivo erróneo, Muestra un mensaje de error.
Salida	Información al usuario mediante una tabla con los archivos resultantes de la búsqueda.
Precondiciones	Se ha establecido conexión con el servidor y además posee el identificador único del archivo a descargar.
Postcondiciones	• Se obtuvo una lista de clientes poseedores del archivo deseado. • Se creó una lista de peticiones de las partes del archivo deseado.

2.2.4. Contrato para caso de uso consultarEstadísticas

Operación	consultarEstadísticas(parametros)
Responsabilidades	Realiza una petición para obtener las estadísticas conforme a la información solicitada por el usuario.
Tipo	Sistema
Referencias Cruzadas	
Excepciones	No existen datos estadísticos, Muestra un mensaje de error.
Salida	Información al usuario mediante una gráfica (de diferente tipo: barras verticales, horizontales, pie) con los datos obtenidos.
Precondiciones	Existencia de datos para mostrar.
Postcondiciones	Se obtuvo una gráfica de con la información solicitada por el usuario.

2.3. Servidor

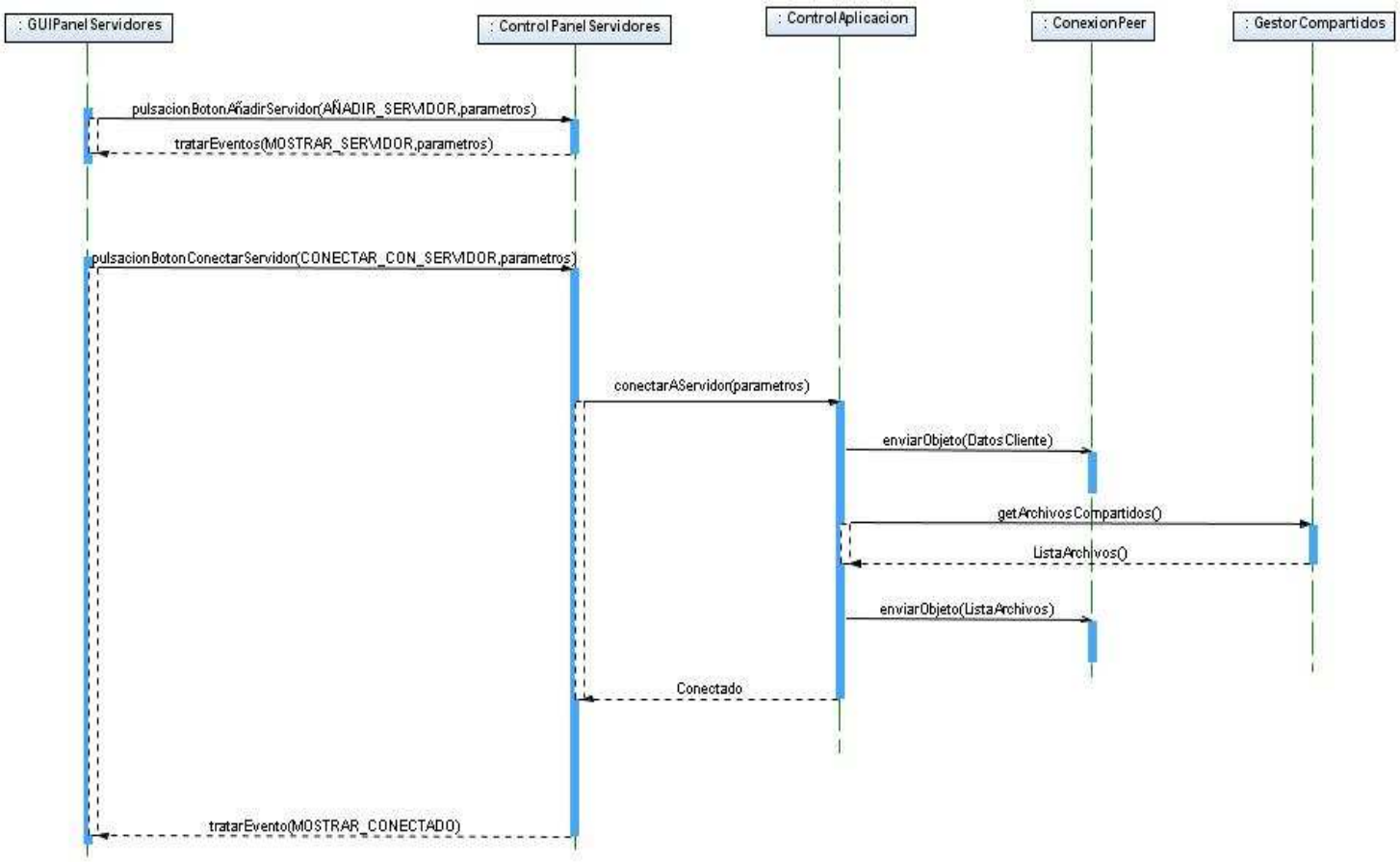
2.3.1. Contrato para caso de uso IniciarServidor

Operación	inciarServidor()
Responsabilidades	Iniciar el sistema Servidor de la aplicación.
Tipo	Sistema
Referencias Cruzadas	
Excepciones	Puerto de escucha ocupado, Muestra un mensaje de error.
Salida	Información al usuario mediante una tabla con los archivos resultantes de la búsqueda.
Precondiciones	No puede haber más de un servidor eGorilla en una misma máquina ejecutándose a la vez.
Postcondiciones	• Se crea una instancia de Servidor. • Se crea una instancia de ArchivosDisponibles. • Se crea una instancia de UsuariosConectados.

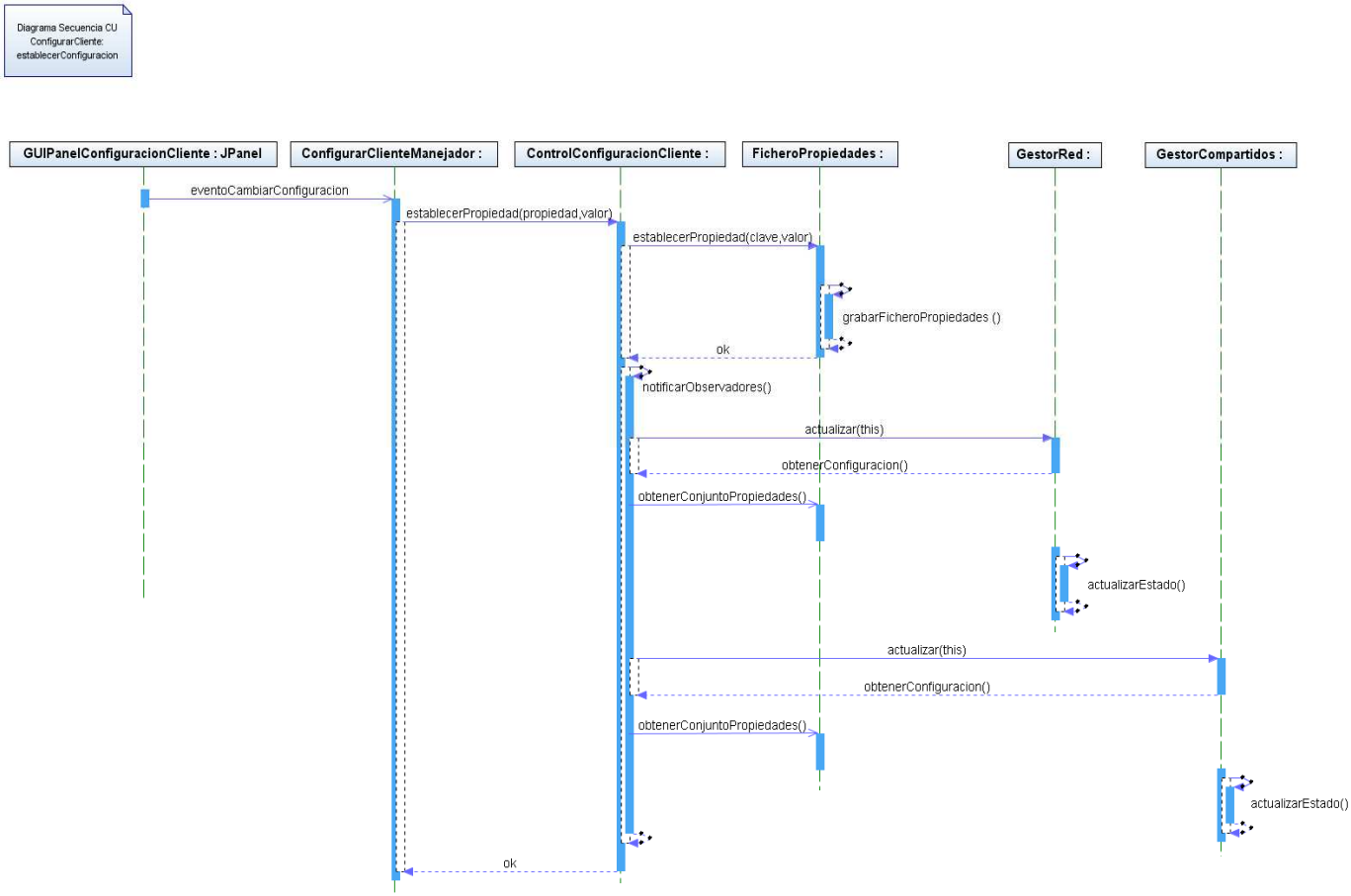
3. Casos de uso reales

3.1. Cliente

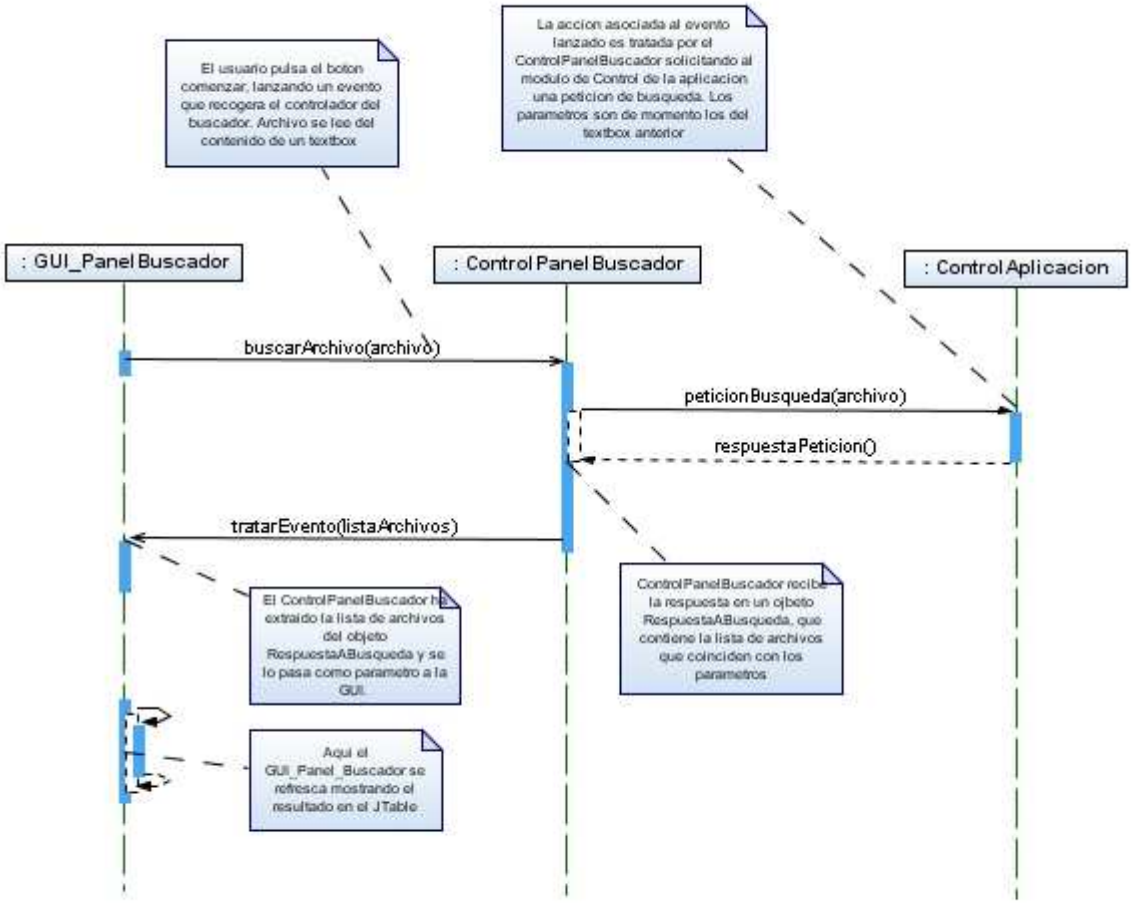
3.1.1. Servidores



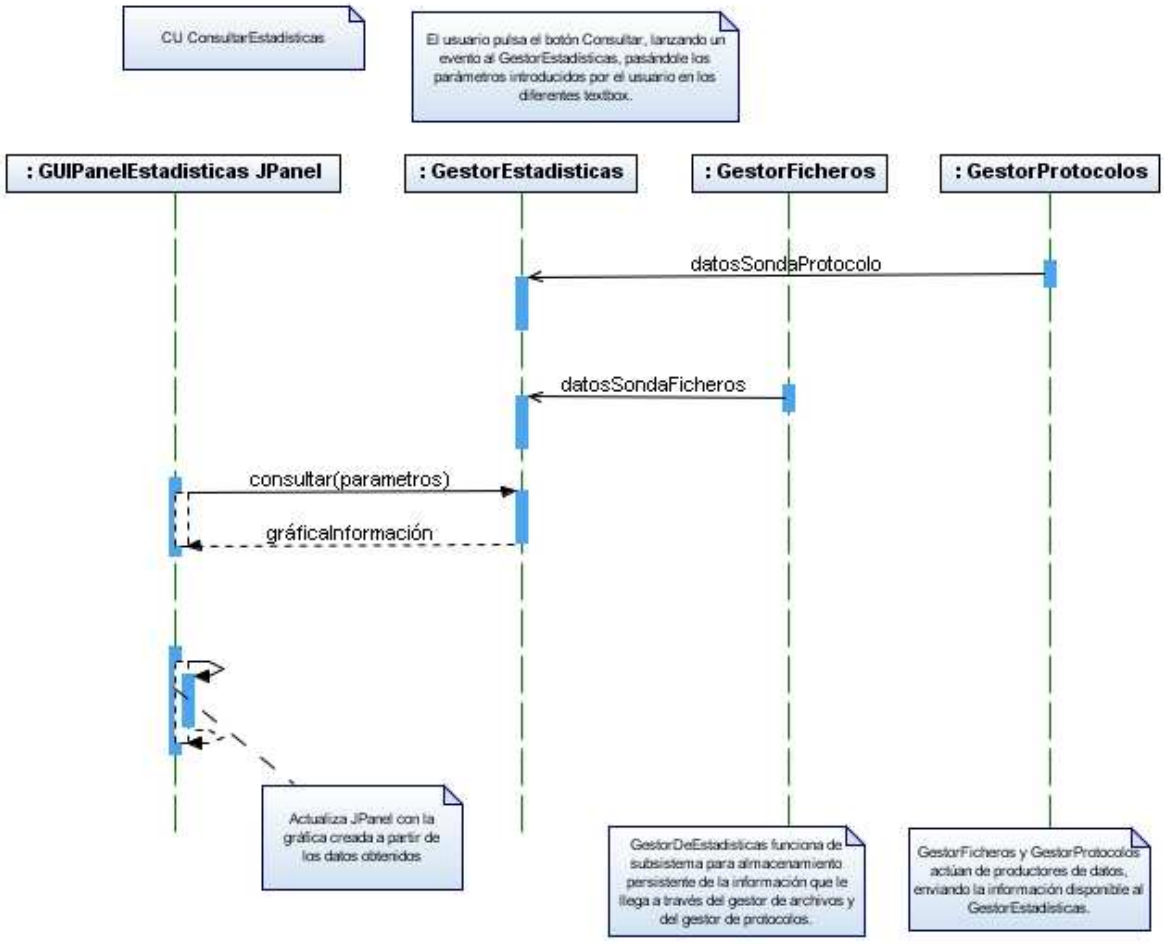
3.1.2. ConfigurarCliente



3.1.3. BuscarArchivos

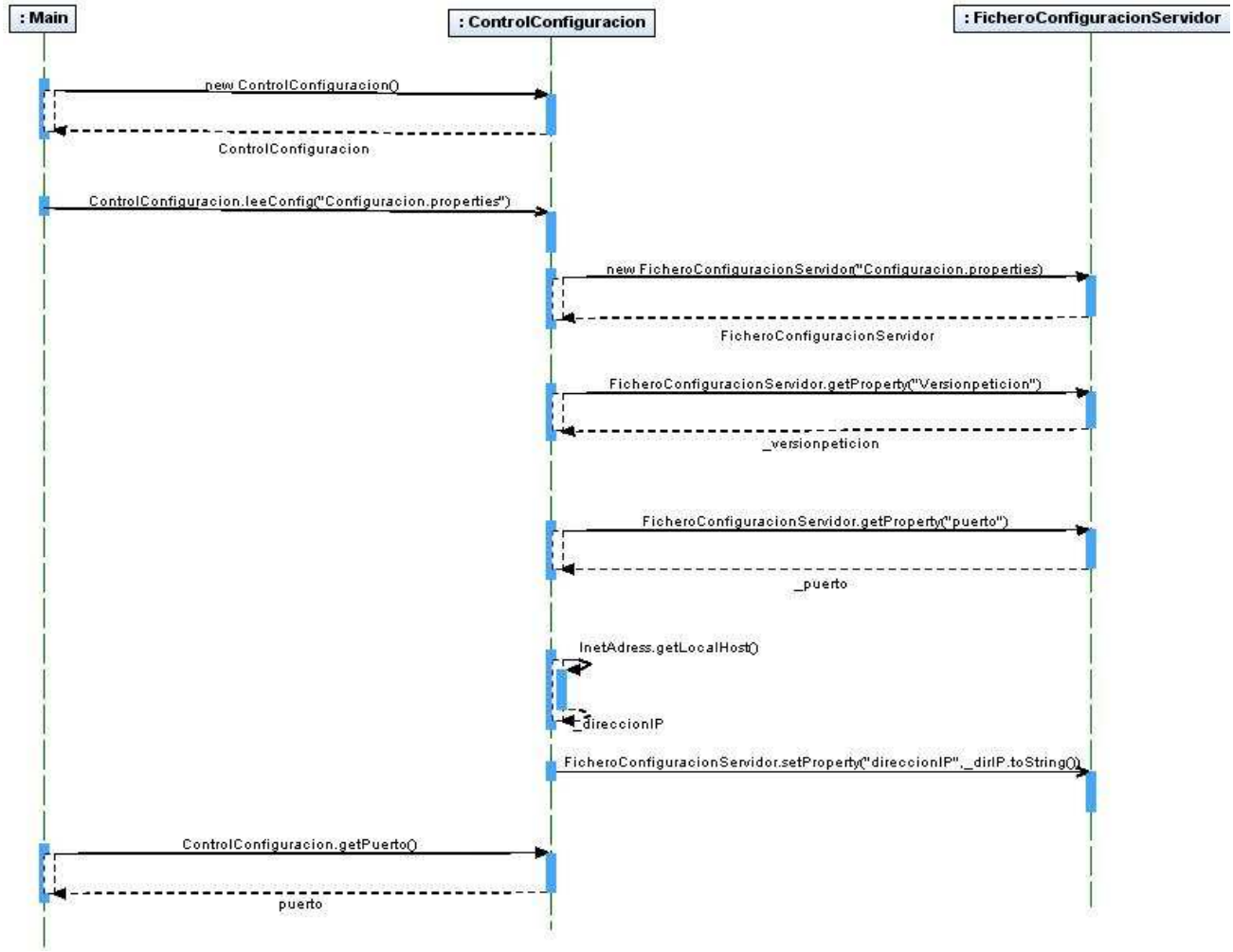


3.1.4. ConsultarEstadísticas



3.2. Servidor

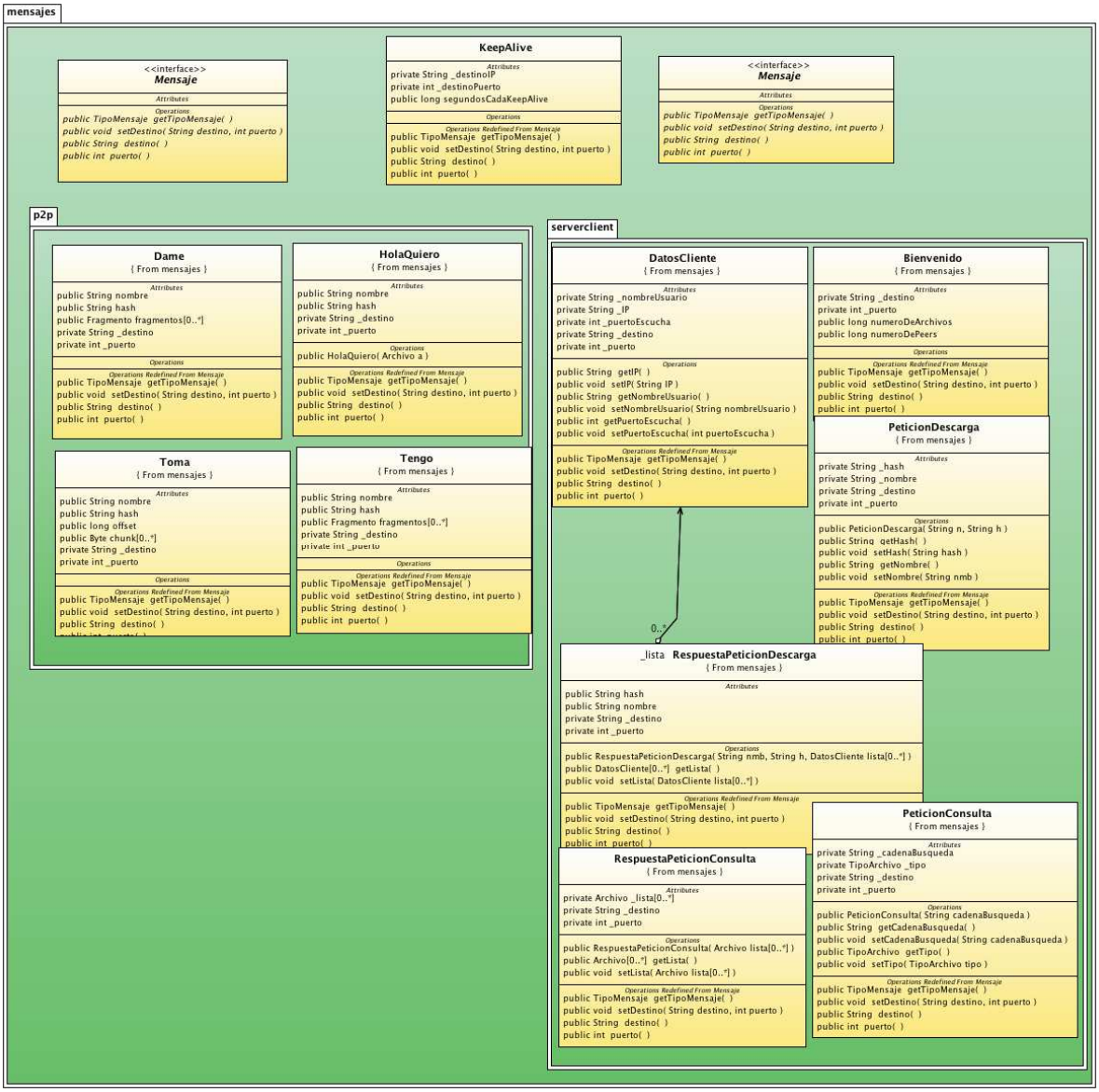
3.2.1. GestorDeConfiguración



4. Diagramas de diseño

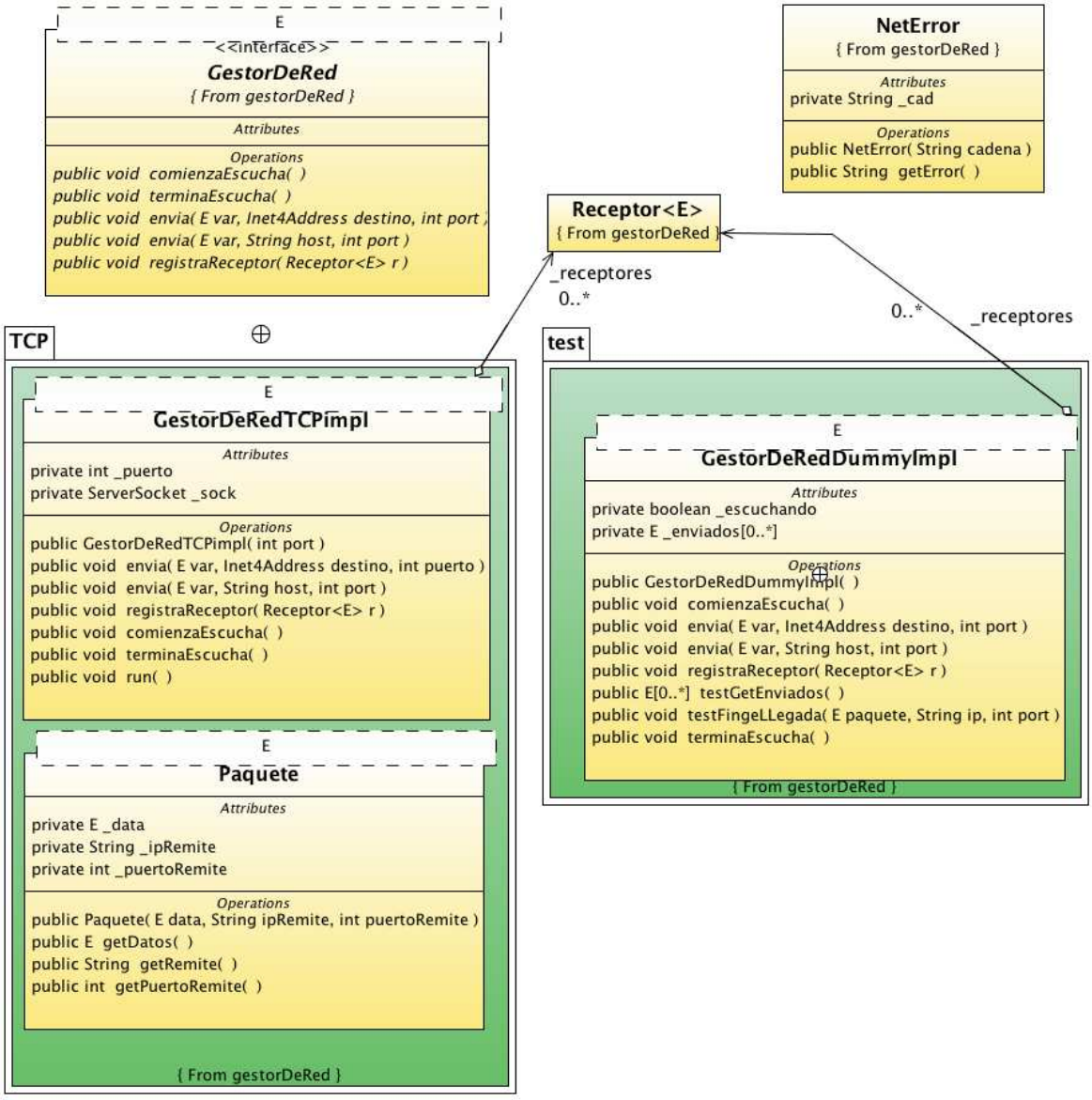
4.1. Diagramas de Clases

4.1.1. Protocolo eGorilla



El **protocolo eGorilla** se comunica compartiendo mensajes ya definidos en un orden establecido. La comunicación puede ser entre dos clientes o entre un cliente y un servidor, por lo que el diálogo es diferente.

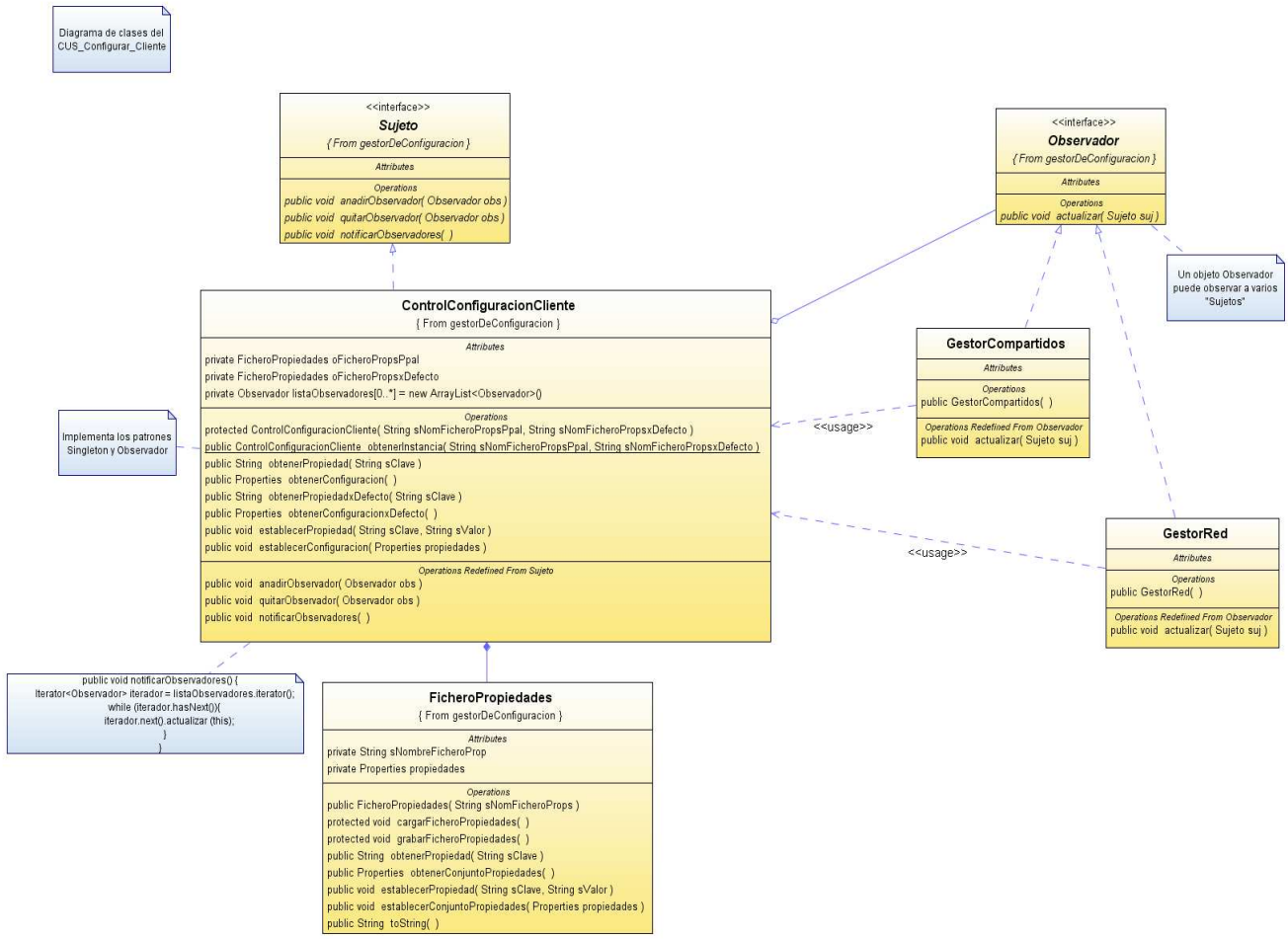
4.1.2. GestorDeRed



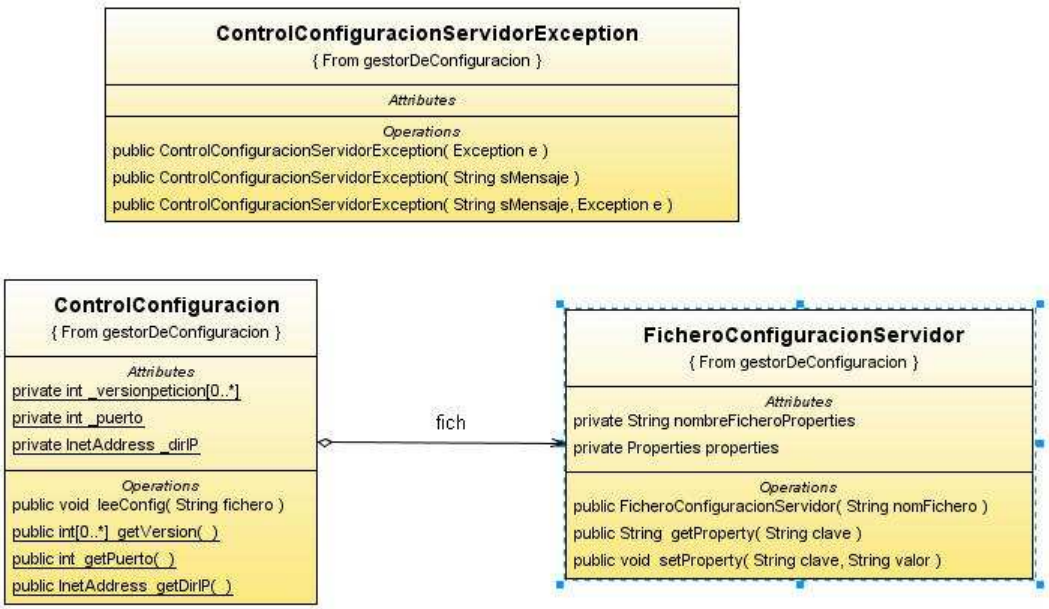
Todas las comunicaciones de red se realizan mediante una interfaz que realiza la abstracción del uso de **sockets** en la aplicación. La **parametrización** de la clase hace que solo sea posible enviar un tipo de objetos. El uso práctico se realizará instanciando la plantilla con la clase Mensaje permitiendo enviar cualquier tipo de mensajes, en nuestro caso se enviarán **mensajes eGorilla**.

Para la realización de pruebas se ha creado una implementación del gestor que no realiza ninguna comunicación de forma que se pueden realizar pruebas sin la necesidad de montar la infraestructura de red.

4.1.3. ConfigurarCliente



4.1.4. GestorDeConfiguración



4.1.5. GestorDeFicheros

El gestor de disco puede verse como una "fachada" que concentra ciertas partes comunes a las clases internas y a las externas, pero sin necesidad de notificar dichos cambios debido al uso de las mismas referencias.

En principio se pensó en utilizar el patrón *Fábrica Abstracta* (*Factory Method*) para poder instanciar otro tipo de clases sin depender de estas mismas, permitiendo después reutilizarla. Pero finalmente como las operaciones comunes para los distintos tipos de clases eran mínimas y sólo había dos tipo de clases a instanciar, aparte un ensamblar o fragmentar archivos no había muchas mas posibilidades, se ha optado por no incluirlo en el diseño.

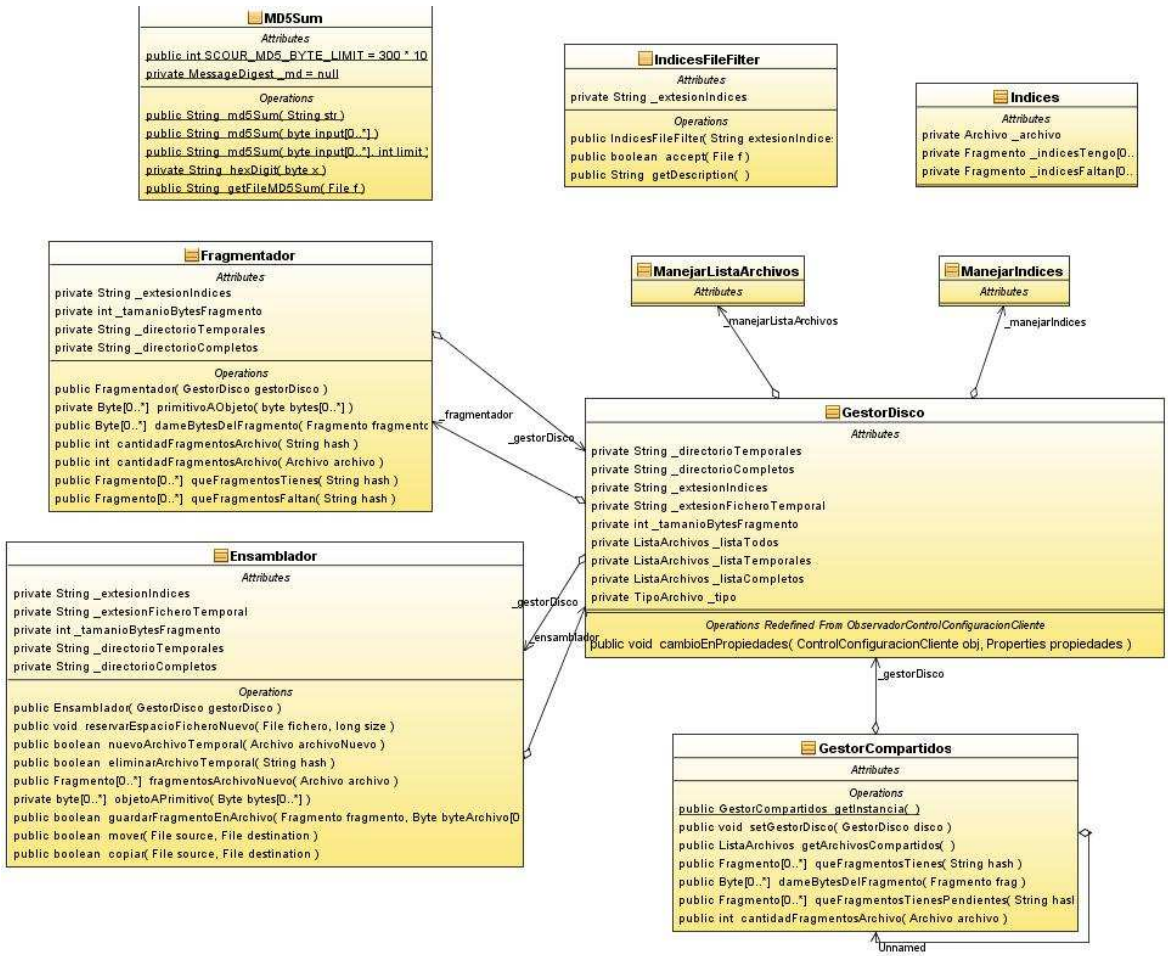


Ilustración – Diagrama de clases para el *GestorDeFicheros*.

También se pensó en el uso del patrón *Decorador* debido a la necesidad de añadir funcionalidad al ensamblador y fragmentador de forma dinámica, para aumentar la flexibilidad que la herencia proporciona en Java. La motivación de esto fue debido a que se podrían necesitar un ensamblador o un fragmentador que pudiera manejar diferentes tipos de índices o listas de archivos, lo que hubiera supuesto crear un *EnsambladorIndicesA, EnsambladorIndicesAListasArchivosA*, *EnsambladorIndicesBListasArchivosA*, *etcétera* y lo mismo con el *Fragmentador*, teniendo un motón de clases para satisfacer todas estas combinaciones. Pero como finalmente se decidió en utilizar un sólo tipo de índices y la mismas listas de archivos no se ha incluido tampoco en el diseño.

A continuación se muestran los diagramas de secuencia que ilustran algunas de las posibilidades de uso de esta parte.

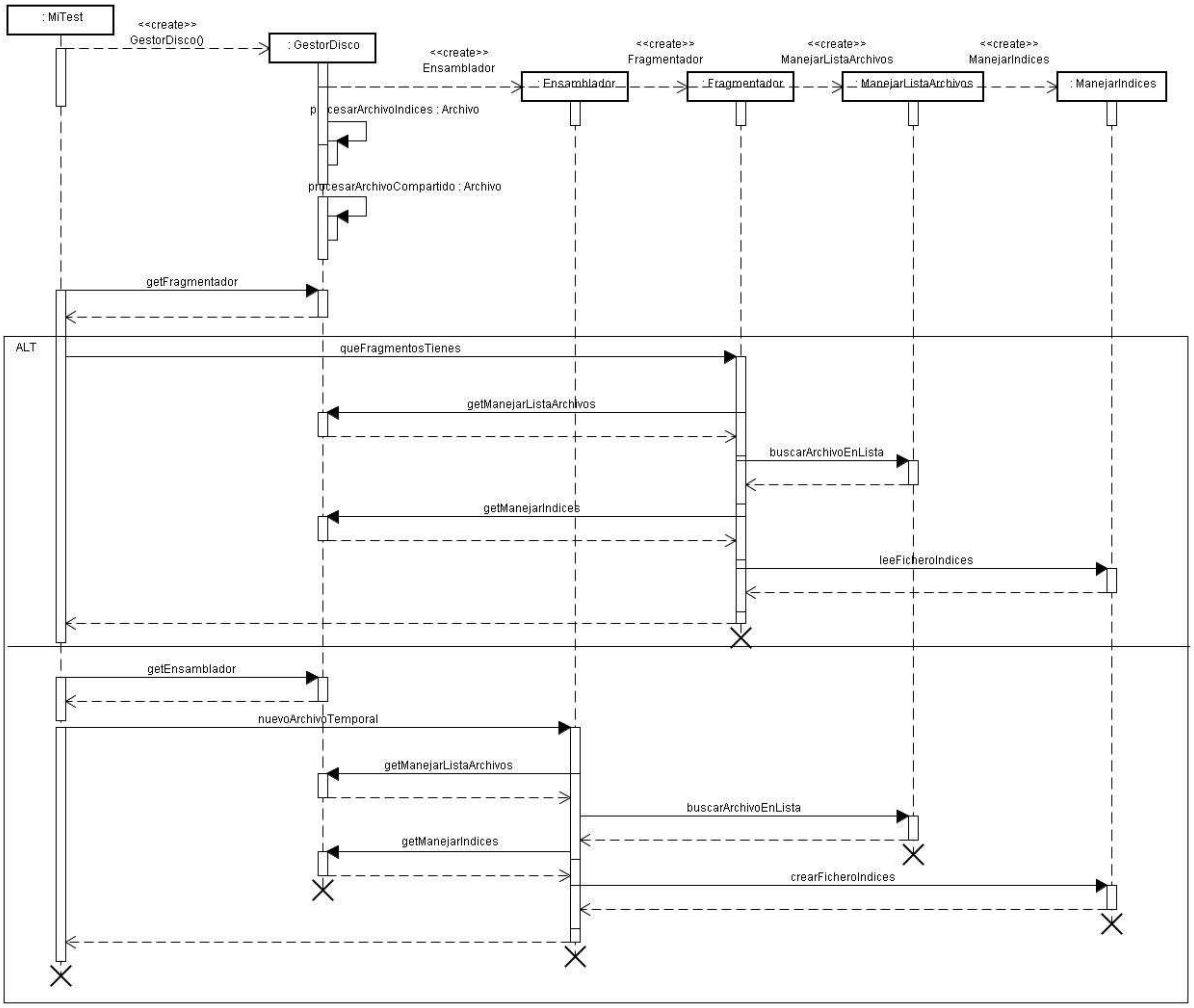


Ilustración – Diagrama de secuencia para el *GestorDeFicheros*.

4.2. Diagramas de Estados

5. Diagrama de comunicación entre módulos