

Universidad de las Ciencias Informáticas

Facultad 1



**Título: Sistema para la gestión de navegación por tiempo y tráfico en Smart Keeper
2.0**

Trabajo de Diploma para optar por el Título de Ingeniería en Ciencias Informáticas

Autor

Eduardo Díaz Vázquez

Tutor

Ing. Luis Enrique Sánchez Arce

La Habana, Junio 2012

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Eduardo Díaz Vázquez

Firma del Autor

Luis Enrique Sánchez Arce

Firma del Tutor

DEDICATORIA

A mis abuelos Samuel y Dictinio, que no pudieron verme convertido de Medio Técnico a Ingeniero.

Eduardo Díaz Vázquez

AGRADECIMIENTOS

Agradezco infinitamente:

A mis padres, por darme siempre su apoyo incondicional y confiar en mí.

A mi hermanita linda, por estar a mi lado desde el vientre.

A mi hermano, por ser mi faro y guía, cuando sea grande quiero ser como tú.

A mis abuelas, por estar siempre pendientes de mí.

A mis primos y tíos, por preocuparse de mi carrera en todo momento.

A mi novia, por su amor y comprensión y darme la fuerza en los momentos difíciles, te quiero.

A Eli y Luisi, por darme una sobrinita tan bella, ahora sí puedo ir con Vero al parque.

A mi cuñada, por su estar siempre atenta a mi desempeño.

A mi familia de Ranchuelo, por acogerme como un hijo.

A mi tutor, por su entrega, dedicación e interés por ayudar siempre, por ser como un compañero de tesis.

A mi amigo de la infancia Wilo, a los del preuniversitario Obel y Daniel, a los de la universidad Jose, Pepo, Juanky y Elier.

A mis colegas del proyecto Eddy, Yasmani, Novoa, Enier y Rubén, y las chicas Maidelys, Fanny, María y Llanela, por trabajar juntos tantas noches.

A Yurisleidy y Dailenis, por sus consejos oportunos.

A todos los profesores que contribuyeron con mi formación en todos estos años de estudio.

A los socios de la beca y de las brigadas, por tantos buenos momentos.

A todo el que de una manera u otra ha hecho posible que este trabajo se realizase, muchas gracias.

RESUMEN

El surgimiento de nuevas tecnologías de desarrollo web y la creación de *hardware*, que permiten alcanzar altas velocidades de conexión, ha permitido que la web evolucione, trayendo consigo un amplio tráfico en la red, lo que afecta en gran medida la transferencia de datos desde Internet. Estas tecnologías, desarrolladas principalmente en el primer mundo, son excesivamente caras. Las instituciones con bajos recursos se ven obligadas a implementar mecanismos para brindar los servicios de navegación con los recursos que disponen. Para este fin se implementan restricciones tecnológicas que regulen el uso del canal de Internet por tiempo de navegación y por cantidad de datos informáticos descargados. El *software* de filtrado Smart Keeper posee mecanismos para regular el ancho de banda que no cumple con las premisas antes mencionadas. El objetivo de este trabajo se enmarca en el desarrollo de un sistema de cuotas que contemple, de forma eficiente, las restricciones de tiempo de navegación y cantidad de datos descargados. Para lograr este objetivo se prevé aprovechar la rapidez en el acceso a datos de la base de datos no relacional MongoDB y la metodología RUP para guiar el proceso de desarrollo del *software*. Con la implementación del sistema de cuotas se logró extender las características de Smart Keeper y hacer más configurables sus mecanismos de control del ancho de banda.

Palabras clave:

Ancho de banda, control, MongoDB, sistema de cuotas, Smart Keeper

ÍNDICE GENERAL

INTRODUCCIÓN.....	1
CAPÍTULO I: FUNDAMENTO TEÓRICO.....	6
1.1 Definición de ancho de banda	6
1.2 Sistemas de Control del Ancho de Banda.....	7
1.2.1 Trabajos similares	7
1.2.2 Comparación de soluciones.....	13
1.3 Evolución de los contenidos en la web	14
1.3.1 Implicación en el ancho de banda.....	17
1.4 Sistema de cuotas para la navegación en Internet.....	18
1.5 Metodologías y herramientas a utilizar.....	19
1.5.1 Metodología de desarrollo	19
1.5.2 Lenguajes de programación	21
1.5.3 Entorno de desarrollo	22
1.5.4 Herramienta CASE	23
1.5.5 Selección de la solución para el almacenamiento de los datos.....	24
1.6 Conclusiones	30
CAPÍTULO II: PROPUESTA DE SOLUCIÓN.....	31
2.1 Descripción del problema	31
2.2 Modelo de Dominio.....	32
2.3 Solución propuesta	34
2.4 Requisitos funcionales y no funcionales	37
2.4.1 Requisitos funcionales.....	37
2.4.2 Requisitos no funcionales	38
2.5 Diagrama de Casos de Uso del Sistema	39
2.5.1 Definición de los actores.....	39
2.5.2 Listado de casos de uso	39

2.5.3	Diagramas de casos de uso	42
2.5.4	Casos de uso expandidos.....	43
2.6	Conclusiones	52
CAPÍTULO III: ANÁLISIS Y DISEÑO		53
3.1	Análisis	53
3.1.1	Modelo del Análisis.....	53
3.2	Diseño	55
3.2.1	Definición de la arquitectura	55
3.2.2	Patrones de diseño.....	57
3.2.3	Diagrama de clases del diseño	58
3.2.4	Diagramas de secuencia	60
3.2.5	Diseño de la base de datos	63
3.2.6	Diagrama de despliegue.....	64
3.3	Conclusiones	65
CAPÍTULO IV: IMPLEMENTACIÓN Y PRUEBAS		67
4.1	Implementación	67
4.1.1	Diagrama de componentes.....	67
4.1.2	Pantallas principales del sistema de cuotas.....	68
4.2	Pruebas	69
4.2.1	Tipos de prueba realizadas.....	70
4.2.2	Resultados de las pruebas realizadas	71
4.3	Conclusiones	71
CONCLUSIONES GENERALES		72
RECOMENDACIONES.....		73
GLOSARIO DE TÉRMINOS		74
REFERENCIAS BIBLIOGRÁFICAS.....		76
ANEXO A: FIGURAS RELACIONADAS		80

ANEXO B: DIAGRAMAS DE CLASES DEL ANÁLISIS	83
ANEXO C: DIAGRAMAS DE CLASES DEL DISEÑO	85
ANEXO D: DIAGRAMAS DE SECUENCIA	90
ANEXO E: DISEÑOS DE CASOS DE PRUEBA BASADOS EN CASOS DE USO	96
Caso de Uso Consultar la cuota consumida	96
Caso de Uso Adicionar valores de consumo	97
Caso de Uso Cambiar estado de la cuota	98
Caso de Uso Reiniciar cuota total a usuario	99
Caso de Uso Reiniciar cuota diaria	100
Caso de Uso Reiniciar cuota semanal	101
Caso de Uso Reiniciar cuota mensual	102
Caso de Uso Reiniciar cuota anual	104
Caso de Uso Reiniciar cuota a un usuario según parámetro	105

ÍNDICE DE FIGURAS

Figura 1: Modelo de dominio.....	33
Figura 2: Diagrama de Casos de Uso del Sistema (Usuario y Administrador).....	42
Figura 3: Diagrama de Casos de Uso del Sistema (Syslog-ng).....	43
Figura 4: Diagrama de Casos de Uso del Sistema (Crontab).....	43
Figura 5: Diagrama de Clases del Análisis: CU Cambiar el estado de la cuota.....	54
Figura 6: Diagrama de Clases del Análisis: CU Reiniciar cuota diaria.....	54
Figura 7: Diagrama de Clases del Análisis: CU Reiniciar cuota a un usuario según parámetro.	54
Figura 8: Diagrama de Clases del Diseño: CU Cambiare el estado de la cuota.	59
Figura 9: Diagrama de Clases del Diseño: CU Reiniciar cuota diaria.....	59
Figura 10: Diagrama de Clases del Diseño: CU Reiniciar cuota a un usuario según parámetro.	60
Figura 11: Diagrama de Secuencia: CU Cambiar el estado de la cuota.....	61
Figura 12: Diagrama de Secuencia: CU Reiniciar cuota diaria.....	62
Figura 13: Diagrama de Secuencia: CU Reiniciar cuota a un usuario según parámetro. .	62
Figura 14: Colección Quota.	64
Figura 15: Diagrama de despliegue.	65
Figura 16: Diagrama de componentes.	68
Figura 17: Vista de administración de la cuota consumida por un usuario y funciones de reinicio.	69
Figura 18: Funcionalidad Reiniciar cuota desde la vista de administración.....	69
Figura 19: Interfaz de administración de la herramienta Antamedia Bandwidth Manager.	80
Figura 20: Interfaz de administración de la herramienta Bandwidth Controller.....	81
Figura 21: Interfaz de la herramienta Squish.	82
Figura 22: Diagrama de Clases del Análisis: CU Consultar la cuota consumida.	83
Figura 23: Diagrama de Clases del Análisis: CU Adicionar valores de consumo	83
Figura 24: Diagrama de Clases del Análisis: CU Reiniciar cuota total a usurario.....	84
Figura 25: Diagrama de Clases del Análisis: CU Reiniciar cuota semanal.....	84
Figura 26: Diagrama de Clases del Análisis: CU Reiniciar cuota mensual.....	84
Figura 27: Diagrama de Clases del Análisis: CU Reiniciar cuota anual.....	84
Figura 28: Diagrama de Clases del Diseño: CU Consultar la cuota consumida.	85

Figura 29: Diagrama de Clases del Diseño: CU Adicionar valores de consumo..... 86

Figura 30: Diagrama de Clases del Diseño: CU Reiniciar cuota total a usuario. 87

Figura 31: Diagrama de Clases del Diseño: CU Reiniciar cuota semanal. 88

Figura 32: Diagrama de Clases del Diseño: CU Reiniciar cuota mensual. 88

Figura 33: Diagrama de Clases del Diseño: CU Reiniciar cuota anual..... 89

Figura 34: Diagrama de Secuencia: CU Consultar la cuota consumida. 90

Figura 35: Diagrama de Secuencia: CU Adicionar valores de consumo..... 91

Figura 36: Diagrama de Secuencia: CU Reiniciar cuota total a usuario. 92

Figura 37: Diagrama de Secuencia: CU Reiniciar cuota semanal. 93

Figura 38: Diagrama de Secuencia: CU Reiniciar cuota mensual. 94

Figura 39: Diagrama de Secuencia: CU Reiniciar cuota anual..... 95

ÍNDICE DE TABLAS

Tabla 1: Comparación de soluciones.	13
Tabla 2: Definición de los actores del sistema.	39
Tabla 3: Definición de CU: Consultar la cuota consumida.	39
Tabla 4: Definición de CU: Adicionar valores de consumo.	40
Tabla 5: Definición de CU: Cambiar el estado de la cuota.	40
Tabla 6: Definición de CU: Reiniciar cuota total a usuario.	40
Tabla 7: Definición de CU: Reiniciar cuota diaria.	40
Tabla 8: Definición de CU: Reiniciar cuota semanal.	41
Tabla 9: Definición de CU: Reiniciar cuota mensual.	41
Tabla 10: Definición de CU: Reiniciar cuota anual.	41
Tabla 11: Definición de CU: Reiniciar cuota un usuario según parámetro.	41
Tabla 12: Descripción CU: Consultar la cuota consumida.	44
Tabla 13: Descripción CU: Adicionar valores de consumo.	45
Tabla 14: Descripción CU: Cambiar el estado de la cuota.	46
Tabla 15: Descripción CU: Reiniciar cuota total a usuario.	46
Tabla 16: Descripción CU: Reiniciar cuota diaria.	47
Tabla 17: Descripción CU: Reiniciar cuota semanal.	48
Tabla 18: Descripción CU: Reiniciar cuota mensual.	49
Tabla 19: Descripción CU: Reiniciar cuota anual.	50
Tabla 20: Descripción CU: Reiniciar cuota a un usuario según parámetro.	51
Tabla 21: Caso de Prueba: CU Consultar la cuota consumida.	97
Tabla 22: Casos de Prueba: CU Adicionar valores de consumo.	97
Tabla 23: Casos de Prueba: CU Cambiar estado de la cuota.	98
Tabla 24: Casos de Prueba: CU Reiniciar cuota total a usuario.	99
Tabla 25: Casos de Prueba: Reiniciar cuota diaria.	101
Tabla 26: Casos de Prueba: CU Reiniciar cuota semanal.	102
Tabla 27: Casos de Prueba: CU Reiniciar cuota mensual.	104
Tabla 28: Casos de Prueba: CU Reiniciar cuota anual.	105
Tabla 29: Casos de Prueba: Reiniciar cuota a un usuario según parámetro.	111

INTRODUCCIÓN

El uso de Internet, o Red de redes como comúnmente se le conoce, se torna indispensable para el desempeño de diversos centros e instituciones tanto a nivel nacional como internacional. El cúmulo de aplicaciones informáticas en Internet, que son visitadas por miles de usuarios, asciende diariamente. Entre los factores que sustentan este fenómeno en torno a Internet destaca su propia naturaleza, caracterizada por su disponibilidad global y permanente, así como por el acceso a gran cantidad de información en forma de páginas web.

La web fue diseñada originalmente como un medio puramente informativo, evolucionando hasta convertirse en un medio donde coexisten información, aplicaciones y gran variedad de servicios. En el período 1992 hasta la actualidad, se desarrollaron nuevas tecnologías, que permitieron el desarrollo de páginas dinámicas, la incorporación de elementos gráficos al diseño web, tecnologías multimedia, hojas de estilo y el uso de lenguajes *scripts* del lado del cliente, como Javascript. El estudio de estas transformaciones hace posible percibir cómo el volumen de carga de las páginas web ha ido en aumento, trayendo consigo un amplio tráfico en la red, lo que afecta en gran medida la transferencia de datos desde Internet.

El incremento en el volumen de datos que contienen las páginas web ha tenido su sustento en el desarrollo de nuevas tecnologías de *hardware* que permiten obtener altas velocidades de conexión. Estas tecnologías, desarrolladas principalmente por transnacionales del primer mundo, son excesivamente caras. Los precios elevados de estos equipos anulan la posibilidad de ser adquiridos por las instituciones de países pobres o empresas con bajos ingresos, que se ven obligadas a adoptar mecanismos para brindar servicios de conectividad a Internet con los recursos que disponen.

En las instituciones con bajas velocidades de conexión, donde se brindan servicios de Internet, se realizan campañas para concientizar a los usuarios y lograr un mejor aprovechamiento de los recursos de la red, adicionalmente se toman medidas tecnológicas que ayuden a eliminar o, en última instancia, disminuir considerablemente las consecuencias del problema antes mencionado.

Centrando la atención en el caso particular de las medidas tecnológicas, se pueden

encontrar dos formas principales para obtener un mejor aprovechamiento del ancho de banda, una desde el punto de vista del tiempo de conexión y otra relacionada con la descarga de componentes web accedidos por los usuarios, siendo la primera de estas variantes la más utilizada en la actualidad a nivel internacional por las compañías proveedoras de servicio de Internet.

En la Universidad de las Ciencias Informáticas (UCI) se desarrolla desde 2005 Smart Keeper, un filtro de contenidos web que permite regular (admitir o denegar) el acceso de los usuarios a determinados contenidos disponibles en Internet. Adicionalmente dicho sistema incorpora un conjunto de características que complementan su objetivo principal, entre las que se encuentran procedimientos que contribuyen a un mejor aprovechamiento del ancho de banda. Sin embargo, dichos procedimientos reproducen los mecanismos de control del ancho de banda que se implementan en la UCI, enfocados en la restricción de la cantidad de datos que los usuarios pueden descargar de la web en el período de un mes.

Estos mecanismos, al estar centrados en el entorno de la UCI, son poco flexibles en ambientes con características diferentes, debido a que no brindan la posibilidad de definir diversos períodos de asignación para la navegación. El hecho antes mencionado, le impone a las instituciones que hagan uso del sistema, la adaptación a los paradigmas de control del ancho de banda de la UCI, sin considerar características del servicio que ofrecen como la cantidad de usuarios y la velocidad de conexión que poseen, que permiten variar, en mayor o menor medida, el nivel de rigurosidad de las reglas que se apliquen.

A estos inconvenientes se suma la imposibilidad de gestionar la navegación teniendo en cuenta el período de tiempo que los usuarios hacen uso del canal de Internet, así como combinar ambas medidas tecnológicas, tecnológicas para obtener un mayor beneficio de los recursos de la red, reflejando que los procedimientos de control del ancho de banda de Smart Keeper no se ajusten a las normas estándares para este tipo de herramientas.

A partir de los planteamientos anteriores y con el propósito de extender y fortalecer los mecanismos de control del ancho de banda en Smart Keeper, se plantea el siguiente **problema a resolver**: ¿Cómo gestionar la navegación de los usuarios en Smart Keeper, teniendo en cuenta el tráfico de datos y el tiempo de navegación?

Por lo que se formula la siguiente **idea a defender**: el desarrollo de una herramienta que permita el control del tráfico de red generado por la descarga de componentes web y el tiempo de conexión de los usuarios en Smart Keeper, ampliará los mecanismos de control de ancho de banda de este sistema y permitirá un mejor aprovechamiento del canal de Internet.

En consecuencia con el problema a resolver y la idea a defender previamente planteadas, se define como **objeto de estudio** los mecanismos de control del ancho de banda y como **campo de acción** los mecanismos de control del ancho de banda para la navegación en Internet. El **objetivo general** derivado de la idea a defender es el siguiente: desarrollar una aplicación para Smart Keeper que permita el control del tráfico de red generado por la descarga de componentes web y el tiempo de conexión de los usuarios, para contribuir al aprovechamiento del ancho de banda así como a su estandarización con las formas más comunes del control del ancho de banda. De esta forma se derivan los siguientes **objetivos específicos**:

- ✓ Sistematizar fundamentos, técnicas y procedimientos de las principales herramientas desarrolladas para el control de la navegación de los usuarios en Internet.
- ✓ Diseñar las funcionalidades de la aplicación para el control de la navegación de los usuarios.
- ✓ Implementar la aplicación para el control de la navegación de los usuarios.
- ✓ Integrar la aplicación a Smart Keeper.
- ✓ Validar las funcionalidades de la aplicación realizada.

Para materializar los objetivos propuestos se especifican las siguientes tareas de investigación:

- ✓ Realización de un sistema de conocimientos con las principales herramientas desarrolladas para el control de la navegación de los usuarios en Internet.
- ✓ Realización del análisis y diseño de las funcionalidades requeridas.
- ✓ Codificación de las funcionalidades diseñadas para la aplicación.
- ✓ Integración de la aplicación con Smart Keeper.
- ✓ Realización de pruebas de funcionalidad e integración de la aplicación desarrollada.

Se pretende obtener como **posible resultado**: una aplicación que permitirá a Smart

Keeper llevar el control del tráfico de red, generado por la descarga de componentes web, y el tiempo de conexión de los usuarios durante la navegación de los mismos por Internet.

Para lograr el cumplimiento de las tareas se utilizaron los siguientes métodos científicos:

- ✓ **Analítico-sintético:** permitió descomponer en diversas partes el conocimiento durante la investigación sobre los mecanismos de control del tráfico de red, para un posterior estudio y arribo de conclusiones según los resultados.
- ✓ **Histórico-lógico:** valió para comprender la evolución de los contenidos en la web, así como la repercusión de su desarrollo en el ancho de banda.
- ✓ **Inducción y deducción:** se hizo uso de estos métodos científicos cuando se analizaron las herramientas que se utilizan para tratar el problema a resolver, tanto a nivel internacional como nacional, arribándose a conclusiones de cuales pudieran o no ser utilizadas.
- ✓ **Comparación y analogía:** al comparar herramientas similares de control del ancho de banda en cuanto a diferentes métricas.

La presente investigación está estructurada en introducción, cuatro capítulos, conclusiones generales, recomendaciones y un apartado dedicado a las referencias bibliográficas. La estructura de los capítulos obedece a la lógica que se describe a continuación:

El primer capítulo se centra en el estudio del tema relacionado con el objeto de estudio definido. Se analiza la existencia de soluciones en el ámbito nacional e internacional que puedan ser reutilizables y de manera general se abordan temas referentes a los mecanismos de control del ancho de banda.

En el segundo capítulo se profundiza en el problema a resolver a través de su descripción. También se describe una propuesta de solución así como los diagramas que se generan en la medida en que se detallan las funcionalidades a implementar.

En el tercer capítulo se muestran los artefactos que son generados durante el análisis y diseño de la solución, así como algunas características de la implementación.

Por último, el cuarto capítulo se centra en la descripción de la implementación, así como las pruebas realizadas al sistema luego de estar en estado funcional e integrado a Smart

Keeper.

CAPÍTULO I: FUNDAMENTO TEÓRICO

En este capítulo se precisan elementos teóricos que sustentan la investigación y el desarrollo del tema propuesto, a través del estudio y análisis de soluciones existentes que puedan ser reutilizadas e integradas a Smart Keeper. Se arriban a conclusiones sobre el uso o no de estas, se detalla la evolución de los contenidos en la web y la implicación de su desarrollo en el ancho de banda. Se definen además la metodología y herramientas para el desarrollo de la solución al problema planteado.

1.1 Definición de ancho de banda

En el área de las comunicaciones en la capa física, el término ancho de banda se relaciona con el ancho espectral de señales electromagnéticas o con las características de propagación de los sistemas de comunicación [\[1\]](#).

Del mismo modo, dentro de las redes de computadoras existe el término de ancho de banda, que es la cantidad de información o de datos que se puede enviar a través de una conexión de red en un período de tiempo determinado [\[1\]](#). El uso del término en este trabajo se hace centrado en el contexto de las redes de computadoras.

El ancho de banda depende de las tecnologías que se estén implementando en la red y juega un papel fundamental en el desempeño de la red, esto se debe a que los servicios requieren el uso de este recurso en mayor o menor medida; donde la competencia por este recurso se mide en términos de muchos factores como el número de usuarios en la red, la capacidad de los dispositivos de red para manejar el flujo de la información y el tipo de servicios que se solicitan en la red.

La importancia del ancho de banda radica en que es un recurso finito. En un momento dado la saturación de usuarios en la red y la competencia por este recurso puede generar que la transferencia de información se ejecute a velocidades por debajo de la capacidad de la red, provocando que la disponibilidad de los servicios se vea afectada, disminuyendo su nivel de calidad o la pérdida total del mismo.

1.2 Sistemas de Control del Ancho de Banda

1.2.1 Trabajos similares

La regulación del ancho de banda es un requisito indispensable en un servicio de Internet, por cuanto es un recurso limitado y costoso. Desde el punto de vista económico es necesario poder repartir los costos del enlace de Internet entre los usuarios finales del servicio y desde el punto de vista técnico, es necesario repartir la capacidad de los equipos para evitar congestión en los equipos de transmisión. La regulación se logra mayormente utilizando una técnica llamada *traffic shapping* (recorte de tráfico) que retrasa la entrega de los paquetes desde y hacia Internet.

De este modo se evita que el usuario sobrepase un límite conocido como “umbral” a partir del cual se comienza a hacer la regulación. Es posible definir diferentes umbrales dependiendo de la ocupación, pero esto también depende de los equipos disponibles para hacer esa regulación. La opción más justa sería asignar a cada usuario el ancho de banda disponible de forma equitativa, lo cual requiere saber cuántos usuarios están conectados en determinado instante y dividir el ancho de banda contratado entre el número de usuarios.

Resulta un poco complejo de implementar y existen pocos dispositivos disponibles en el mercado para realizar esta operación. Sin embargo, se pueden hacer cálculos estadísticos basados en el número de usuarios conectados contra las horas del día en las que se tomaron las muestras y se programa un perfil que se active en un instante de tiempo determinado. Esta característica sí está disponible en la mayoría de equipos de regulación.

Se consideran trabajos similares aquellos que fueron materializados para resolver la esencia del problema que origina este trabajo. Luego de realizada una investigación, respecto a programas que hagan uso de algún mecanismo para mejorar el rendimiento de la navegación web, los resultados que no se corresponden con sistemas de cuotas, se centran en programas que son construidos bajo la premisa de monitorear y controlar el ancho de banda que usan las aplicaciones y usuarios de Internet. A continuación se presentan evidencias de las soluciones.

1.2.1.1 Ámbito Internacional

1.2.1.1.1 *Hardware*

Existe *hardware* dedicado a la administración del ancho de banda para redes de gran tamaño, lo que simboliza más de 1000 conexiones activas en la red corporativa. Algunas de estas compañías son Cisco¹ y Allot Communications². Las soluciones basadas en *hardware* se han desarrollado hasta el momento para redes cableadas principalmente, aunque existen equipos para ambientes inalámbricos, siendo estos últimos equipos muy costosos para la implementación en oficinas o instituciones de menos de 100 usuarios.

Cisco Systems: es una empresa multinacional principalmente dedicada a la fabricación, venta, mantenimiento y consultoría de equipos de telecomunicaciones tales como: dispositivos de conexión para redes informáticas, dispositivos de seguridad y *software* de gestión de red. Actualmente, *Cisco Systems* es líder mundial en soluciones de red e infraestructuras para Internet. Además de desarrollar el *hardware* de sus equipos, *Cisco Systems* también se ocupa de desarrollar su propio *software* de gestión y configuración de los mismos. Dicho *software* es conocido como IOS³ de código totalmente propietario [2].

Allot Communications: es un proveedor de *hardware* dedicado que permite regular el ancho de banda, provee visibilidad granular y control dinámico, necesarios para que los operadores de la red sean capaces de optimizar la disponibilidad, el rendimiento y la rentabilidad de las redes WAN⁴ y de los servicios de banda ancha. Los dispositivos de la serie *NetEnforcer* son utilizados en miles de instalaciones alrededor del mundo, donde aplica un avanzado sistema de inspección de paquetes para ayudar a los operadores a identificar, clasificar, priorizar y manipular el tráfico en las redes para cada usuario. A través del monitoreo en tiempo real, la aplicación de políticas y el direccionamiento de tráfico de servicios, estos dispositivos permiten a las empresas y proveedores de servicios controlar la utilización del ancho de banda y los costos, lo que propicia la protección y mejora de la calidad del servicio para todos los usuarios de la red [3].

¹ Más información disponible en: <http://www.cisco.com>

² Más información disponible en: <http://www.allot.com/>

³ Siglas de *Internetwork Operating System*, sistema operativo de interconexión de redes. Posee funciones de enrutamiento, conmutamiento y telecomunicaciones que se integra estrechamente con un sistema operativo multitarea.

⁴ Siglas de *Wide Area Network*, red de área amplia. Es un tipo de red de computadoras capaz de cubrir distancias desde unos 100 hasta unos 1000 km, proveyendo de servicio a un país o un continente.

1.2.1.1.2 *Software*

Para el caso de las soluciones basadas en *software* se hace notar que el punto central es el uso de un servidor que permite la regulación del ancho de banda en la red.

Antamedia Bandwidth Manager: es un *software* de puerta de enlace para computadoras, que permite controlar y limitar el uso de Internet, incluida la tasa de carga y descarga, de cada equipo de la red. El *software* posibilita definir tiempos específicos de conexión a Internet así como cuotas para las tasas de descarga de datos por sesiones para cada estación de trabajo [4].

Este *software* es adecuado para diferentes propósitos:

- ✓ Hogares o complejos residenciales con servicio de Internet compartido, limitación de conexión por usuarios y bloqueos de URLs⁵ basado en palabras claves.
- ✓ Instituciones y centros educacionales, limitación de tráfico y el seguimiento del contenido que se visita, bloqueo de contenido sensible o sitios web no deseados.
- ✓ Entidades que brindan servicio de Internet en el que cada equipo puede tener diferentes tasas de carga y descarga y tiempos de conexión.

Las principales características de esta herramienta son las siguientes:

- ✓ Control del ancho de banda real por carga y descarga, más allá de reglas de restricción del canal.
- ✓ Cada equipo puede tener diferentes tasas de carga y descarga, y tiempos de conexión.
- ✓ Controla tanto el ancho de banda como la cantidad de datos (por ejemplo, 60 Kps, 10 Mb, o ambos).
- ✓ Monitoreo en tiempo real de los equipos activos con panel de control fácil de usar y centralizado.
- ✓ Fácil implementación en cualquier PC estándar con dos tarjetas de red.
- ✓ Basado en IP.

⁵ Siglas de *Uniform Resource Locator*, localizador de recursos uniforme. Es una secuencia de caracteres, fijada a un formato estándar, que se usa para nombrar recursos en Internet para su localización o identificación.

- ✓ Solamente soportado en Sistema Operativo Windows [4].

Bandwidth Controller: es un *software* de administración de ancho de banda para redes basado en IP. Provee administración de redes con capacidad para realizar recortes de tráfico y control de flujo, incluyendo componentes automatizados para la asignación de Internet [5].

Entre sus principales características se puede destacar:

- ✓ Administración de ancho de banda Capa 3.
- ✓ Establecimiento de límites y prioridades en flujo TCP/IP⁶.
- ✓ Monitoreo de tráfico.
- ✓ Administración remota.
- ✓ Soporte para redes inalámbricas.
- ✓ Hasta 2.000 usuarios concurrentes.
- ✓ Solamente soportado en Sistema Operativo Windows [5].

Squid⁷: Es un popular programa de *software* libre que implementa un servidor *proxy* y un dominio para caché de páginas web, publicado bajo licencia GPL⁸. Tiene una amplia variedad de utilidades, desde acelerar un servidor web, guardando en caché peticiones repetidas a DNS y otras búsquedas para un grupo de usuarios que comparten recursos de la red, hasta caché de web. Aunque está especialmente diseñado para ejecutarse bajo entornos tipo Unix, es compatible con la mayoría de los sistemas operativos incluyendo Windows. Ha sido desarrollado durante muchos años y se le considera muy completo y sólido. Aunque orientado principalmente a HTTP y FTP es compatible con otros protocolos como Internet Gopher⁹. Implementa varias modalidades de cifrado como TLS, SSL y HTTPS [6].

Realiza una reducción del ancho de banda y mejora los tiempos de respuesta cacheando y reutilizando las peticiones más frecuentes de páginas web. Sin embargo, la caché reduce el uso del canal pero no soluciona el problema del ancho de banda si los usuarios

⁶ Denominación común que se utiliza para designar al conjunto de protocolos de red en los que está basada Internet.

⁷ Más información disponible en: <http://www.squid-cache.org>.

⁸ Siglas de *General Public Licence*, Licencia Pública General. Es una licencia orientada a proteger la libre distribución, modificación y uso de *software*.

⁹ Protocolo de Internet consistente en el acceso a la información a través de menús.

deciden descargar archivos muy “pesados” que generalmente no se guardan en caché.

A través de Squid se puede establecer límites al ancho de banda mediante una de sus características de configuración denominada "*delay pools*". Se implementa una definición de regulación de ancho de banda, se define una ACL¹⁰ y se asocia la regulación a la ACL.

La ventaja de su uso radica en que controla el ancho de banda, sin causar penalidades sobre los objetos traídos desde la caché. En lenguaje técnico de *proxy*, los *delay pools* afectan los *cache misses*, no los *cache hits*.

En Squid 3.x existen cinco clases de *delay pool*:

- ✓ **Clase 1:** Define una única estructura de control. Este limita el uso del canal de manera global sin importar cómo lo usan los clientes internamente o cómo está definida lógicamente la LAN.
- ✓ **Clase 2:** Se manifiesta como un *delay* clase 1, con 256 *delay pools* clase 1 subordinados.
- ✓ **Clase 3:** Este es un *delay pool* clase 1 con 256 *delay pools* clase 2 subordinados.
- ✓ **Clase 4:** Incluye lo relacionado con los *delay pools* clase 3, adicionando un límite sobre usuarios del sistema. Solo es efectivo si el *proxy* tiene implementado reglas de autenticación.
- ✓ **Clase 5:** Se definen reglas de limitación de ancho de banda según los *tags* devueltos por ACLs externas [7].

A pesar de ser mecanismos bastante eficaces en la regulación del ancho de banda, poseen limitaciones que deben ser tomadas en consideración:

- ✓ No comparten ancho de banda. Si existe un solo cliente demandando ancho banda de Internet la cantidad máxima de canal que recibe está limitado por el *delay* más restrictivo que lo cobija sin importar que él sea el único haciendo peticiones al servidor.
- ✓ No hay garantía de asignación equitativa de canal entre los clientes.
- ✓ Si se desea manejar valores de ancho de banda en horarios laborales y otros valores

¹⁰ Siglas de *Access Control List*, lista de control de acceso. Es un concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto.

en horarios no laborales, en el momento de hacer los cambios puede provocar la interrupción de las descargas [8].

Squish: Es una solución de *Software* Libre bajo licencia GPL, actualmente en desarrollo, que analiza los registros de navegación de Squid y permite racionalizar el ancho de banda según las necesidades de los usuarios. Mediante su uso, los administradores de red pueden especificar la cantidad permitida en términos de tiempo de navegación o descarga de datos que puede consumir cada usuario del ancho de banda que posee la entidad. Estas especificidades se toman por día, semana, mes o año [9].

Las características más significativas de esta herramienta se exponen a continuación:

- ✓ Racionaliza el ancho de banda.
- ✓ Genera gráficos del consumo del ancho de banda por usuarios.
- ✓ *Software* de código abierto y reutilizable.
- ✓ Permite ser integrado a Squid.
- ✓ Soportado por sistemas GNU/Linux [9].

1.2.1.2 Ámbito Nacional

De manera progresiva Cuba ha ido desarrollando su incipiente industria de *software* informatizando algunas esferas de manera más visible. La UCI forma parte imprescindible en esta tarea. A pesar de esto, los niveles de informatización que muestra la Isla son considerablemente bajos. La industria cubana del *software* progresa, pero dificultades como la carencia de recursos, la poca preparación del personal y la falta de compromiso de quienes laboran en la rama, atentan contra el buen desenvolvimiento de la misma.

Con respecto al tema tratado en este punto de la investigación, no existen referencias bibliográficas que indiquen el desarrollo de sistemas de control del ancho de banda de factura nacional. De cualquier modo no se asume que no haya implementaciones de los mismos, sólo que, al no haber evidencias al alcance del autor de su existencia dejan de tener peso en el marco de la presente investigación. Sin embargo, cabe destacar que sí se conoce del uso herramientas como Squish en redes nacionales como InfoMed¹¹.

En la UCI, debido a la magnitud de su red y el escaso ancho de banda que posee, se

¹¹ Red telemática del Ministerio de Salud Pública de Cuba.

implementa un mecanismo de control del ancho de banda por usuarios, aunque no se cuenta con documentación que abunde sobre su funcionamiento no debía dejar de mencionarse en el marco de esta investigación.

1.2.2 Comparación de soluciones

En el proceso de comparación, para la definición de las posibles herramientas a utilizar, fueron desechadas las soluciones de *hardware*, ya que, a pesar de que su estudio aporta conocimientos acerca del control del ancho de banda, no se ajustan a las exigencias del problema que presenta el producto Smart Keeper. La solución que se requiere debe ser a nivel de *software*, lo que supone analizar en profundidad el resto de los sistemas estudiados.

La siguiente tabla muestra el comportamiento de las soluciones antes señaladas según diferentes métricas:

	Antamedia Bandwidth Manager	Bandwidth Controller	Squid	Squish
Sistema Operativo	Windows	Windows	GNU/Linux, MacOS X, Windows	Linux
Licencia	Privativa	Privativa	Libre	Libre
Control de datos	Si	Si	No	Si
Control de Tiempo	Si	No	No	Si
Acceso al código fuente	No	No	Si	Si

Tabla 1: Comparación de soluciones.

Como se puede apreciar en la tabla comparativa, Antamedia Bandwidth Manager y Bandwidth Controller son *software* privados que imposibilitan el estudio de su código fuente para obtener detalles de su funcionamiento, especialmente de Antamedia Bandwidth Manager pues permite el control del tiempo de conexión de los usuarios. Estos sistemas figuran como *software* de puerta de enlace que regulan todo el flujo de datos desde y hacia Internet sin diferenciar entre los protocolos de red como el HTTP, elemento imprescindible para controlar el ancho de banda utilizado al navegar por la web. Estos

inconvenientes impiden que se tenga en cuenta el uso de estas aplicaciones aunque la forma en que estos muestran la información puede ser tenida en cuenta para futuras implementaciones.

Squid, a pesar de ser una herramienta potente y muy utilizada en entornos de red, no cuenta con mecanismos óptimos que logren el control del ancho de banda más allá de implementar *traffic shapping*, razón para no tener en cuenta el uso de los mecanismos que provee este *software*.

La herramienta Squish a primera vista pudiese ser la solución perfecta para el problema planteado. Es una herramienta libre, que permite tanto el control de datos como el control del tiempo de conexión de los usuarios. No obstante, su integración con Smart Keeper es nula, utiliza bases de datos diseñadas para ser accedidas, exclusivamente, mediante el uso de bibliotecas del lenguaje Perl, además, luego de haber examinado su código fuente con rigurosidad se detectaron errores en algunos algoritmos.

Pese a estas insuficiencias, la gran mayoría de las funcionalidades de Squish pueden ser tenidas en cuenta en el desarrollo de la solución definitiva, argumento que se maximiza por el hecho de que esta herramienta presenta cierta similitud con los paradigmas de los actuales mecanismos de control del ancho de banda en Smart Keeper, mediante el análisis de los registros de navegación del servidor Squid.

1.3 Evolución de los contenidos en la web

Como ya se hacía alusión en la introducción de la investigación, la evolución de la web puede enmarcar su desarrollo en cuatro grandes etapas o generaciones y aunque en la actualidad las cuatro conviven, ya casi no existen páginas con las características de aquellas generaciones.

La **primera generación** está enmarcada desde el propio nacimiento de la web en 1992 hasta mediados de 1994. En esta etapa la web era empleada como un medio de comunicación tradicional, una simple colección de documentos estáticos. Esta web incipiente puede definirse como la Web 1.0 y las páginas de esta etapa poseen las características que se muestran a continuación:

- ✓ Uso de textos, pocas imágenes y ningún recurso multimedia.

- ✓ Tiempo de carga rápido.
- ✓ Navegación poco estructurada.
- ✓ Páginas que poseen un contenido educativo o científico.
- ✓ Empleo de listas para organizar la información.
- ✓ Se pueden visualizar prácticamente en cualquier navegador.

La llegada de la **segunda generación** está estrechamente relacionada con el desarrollo de diferentes tecnologías entre las que destaca CGI¹², que permitió la creación de las primeras páginas dinámicas.

Las ventajas sobre la primera generación están dadas por la incorporación de elementos gráficos:

- ✓ Los iconos sustituyen las palabras.
- ✓ El color de fondo se sustituye por una imagen.
- ✓ Los banners sustituyen los encabezados de las páginas.

A partir de ese momento comenzarían el proceso de interacción con bases de datos para la creación de aplicaciones de entretenimiento y educación, entre otras. De esta forma la web evoluciona a una etapa superior, ya se hablaba de la Web 1.5. Las principales características de esta etapa son:

- ✓ Tiempo de carga más lento. Empleo excesivo de imágenes y animaciones debido a la novedad de su uso.
- ✓ Empleo de tablas, aunque no con el propósito del diseño, sino para mostrar datos tabulados.
- ✓ Las páginas poseen una estructura de arriba hacia abajo.
- ✓ Prima el uso de tecnologías multimedia (imágenes y sonido), aunque no puedan ser visualizadas por el público correctamente.
- ✓ Aún no existe una filosofía de navegación, aunque existe una navegación jerárquica a partir de la página principal.

A pesar del poco tiempo, en 1996 hace aparición la **tercera generación**. Durante esta etapa el desarrollo de herramientas relacionadas con la web tiene un incremento. Aunque

¹² Siglas de *Common Gateway Interface*, interfaz común de puerta de enlace. Es una tecnología que permite a un cliente solicitar datos de un programa ejecutado en un servidor web.

se extendió el uso de CGI, sus limitantes dieron paso al surgimiento de tecnologías como IDC¹³ y ASP¹⁴. Además emergen tecnologías como *ColdFusion*, PHP y JSP¹⁵ basada en Java.

Las páginas pertenecientes a esta generación son las más comunes en la actualidad. Se caracterizan por:

- ✓ Uso de CSS y optimización del código HTML.
- ✓ Tiempo de carga rápido.
- ✓ Las páginas se limitan para que se puedan visualizar completamente en una pantalla sin tener que realizar desplazamiento (*scroll*).
- ✓ Se crean pensando en el usuario final, teniendo como objetivos ofrecer servicios, informar, vender, etc.
- ✓ Se tienen en cuenta principios tipográficos y de organización visual de la información.
- ✓ Se emplean de forma coherente los colores, las imágenes, tipos de letras, los símbolos e iconos, etc.

La etapa anterior sirve como cimiento de la **cuarta generación** que comenzó en 1999. Gran parte de las páginas web en este período se crean a partir de información almacenada en bases de datos. Representa un salto en la forma de ver Internet con la aparición del concepto Web 2.0.

Las aplicaciones web alcanzan su máximo esplendor y son utilizadas tanto en la gestión de la información de las empresas como en sistemas de transferencia de voz y vídeo. Las características de las páginas de esta etapa son:

- ✓ Se emplean nuevamente los recursos gráficos.
- ✓ HTML evoluciona: se extiende el uso de tecnologías poco empleadas hasta ese momento, como CSS y aparecen nuevas como *Dynamic HTML* (DHTML). Estas tecnologías generan incompatibilidad entre distintos navegadores, a pesar de

¹³ Siglas de *Internet Database Connector*, conector de bases de datos de Internet. Ofrece un mecanismo directo de alto rendimiento para la integración del contenido de una base de datos dentro de una página web.

¹⁴ Siglas de *Active Server Pages*, páginas activas del servidor. Tecnología de Microsoft del lado del servidor para páginas web generadas dinámicamente.

¹⁵ Siglas de *Java Server Pages*, páginas servidoras de Java. Tecnología Java que permite generar contenido web dinámico.

introducir considerables ventajas.

- ✓ Uso de nuevas tecnologías multimedia: Se puede crear un sitio web sin emplear HTML.
- ✓ Crean a los usuarios una experiencia agradable desde que visita la primera página hasta que abandona el sitio [\[10\]](#).

1.3.1 Implicación en el ancho de banda

El tamaño o “peso” promedio de las páginas web de los 1000 sitios web más frecuentados de Internet se ha septuplicado desde 2003. De 2003 a 2011, el peso promedio de las páginas web tuvo un aumento 7,2 veces, de 93,7K a más de 679K. Durante el mismo período de ocho años, el número promedio de referencia a objetos (imágenes, videos flash, programas externos) en las páginas web se incrementó en más de 3 veces, de 26 hasta 85 objetos por página. Las estadísticas a largo plazo muestran que desde 1995, el tamaño de la página web promedio se ha incrementado en 48 veces y el número de objetos por página ha crecido en casi 37 veces [\[11\]](#).

Según un estudio realizado por un equipo de medición de la web de Google, el peso promedio de todas las páginas de la web es de 320K haciendo referencia a un promedio de 44 objetos. La página web promedio utilizada 7,01 hosts por página y 6,26 objetos por host [\[11\]](#).

Un estudio realizado en el 2007 arrojó los siguientes resultados con respecto al uso de objetos externos en páginas web [\[11\]](#):

Uso de lenguajes *script* en la página web media: el 84,8% de las páginas web utilizan elementos de *script*. El tamaño medio de *scripts* externos fue de 8,845 bytes sin compresión y comprimidos 6,302 bytes. El tamaño total de *scripts* por páginas fue de 68,812 bytes sin comprimir y 49,738 bytes comprimidos. El número medio de *scripts* externos fue de 7.

Uso de CSS en la página web media: el 82,4% utiliza la etiqueta de vínculo, y el 54,5% utiliza la etiqueta de estilo (con un promedio de 2,27 etiquetas de estilo de uso interno). El tamaño medio de las hojas de estilo externas fue de 6,575 bytes y 4,457 bytes comprimidos. El tamaño promedio total en hojas de estilo fue de 15,175 de bytes sin comprimir y comprimido 10,347 bytes.

Uso de imágenes en la página web media: las imágenes son utilizadas en el 91,6% de las páginas web. La extensión GIF se utiliza en el 84,6% de las páginas web, frente a un 77,9% en 2006. JPEG se utiliza en el 64,5% de las páginas web, frente a 55,8% en 2006. La extensión PNG se usa en el 32,2% de las páginas web, un incremento significativo con respecto a su uso de 7,2% en 2006.

El uso de medios de *streaming* en la web se ha incrementado en más del 100% cada año. Del año 2000 al 2005 el volumen total de *streaming* de archivos multimedia almacenados en la web creció más del 600%. Más del 87% de todos los medios de transmisión es abandonada por los usuarios en los primeros 10 segundos, sin embargo, esto repercute en la pérdida de hasta el 20% del ancho de banda del servidor. Mientras que sólo el 3% de las respuestas del servidor son los vídeos, que representan más del 98,6% de los bytes transferidos.

1.4 Sistema de cuotas para la navegación en Internet

Generalmente los mecanismos de regulación de ancho de banda, realizan su función mediante la limitación del canal de red, lo que repercute en la calidad del servicio de Internet que recibe el usuario final, ya que aunque haya sólo un usuario navegando, la velocidad máxima estará en función de la restricción a la que esté sujeta la red.

El ancho de banda, matemáticamente expresado, es la razón entre la cantidad de datos que recorren una red y el lapso de tiempo que tardan en recorrerla. Sus magnitudes, enfocadas a la navegación en Internet, se definen como los datos cargados o descargados desde la red de redes y el tiempo que demora en responderse la petición. Los mecanismos de cuotas para la navegación en Internet se basan en estos preceptos matemáticos del ancho de banda para su funcionamiento.

Las cuotas, en ciencias económicas se definen como: asignaciones a los individuos, empresas o países de cantidades máximas permitidas [12]. En el área de dirección estratégica y mercadotecnia, cuota de mercado es la fracción o porcentaje que se tendrá del total de mercado disponible o del segmento del mercado que está siendo suministrado por la compañía [13]. En informática existe el término cuota de disco, este determina el límite establecido por el administrador del sistema que restringe ciertos aspectos del uso del sistema de archivos en los sistemas operativos modernos [14].

Durante la investigación, no se encontraron referencias que reflejaran el uso de las cuotas vinculado a la navegación en Internet, a falta de un concepto de cuota para la navegación en Internet, el autor propone como tal el siguiente: cuota para la navegación en Internet es el límite de uso del ancho de banda establecido por los proveedores del servicio de Internet a los usuarios.

Un mecanismo de cuotas para la navegación en Internet, es una herramienta que va más allá de una restricción del ancho de banda. El uso de estos evita que la red se sature pero sin comprometer las tasas de velocidad. Son ideales en ambientes donde el ancho de banda es muy reducido y fragmentarlo sería penalizar en demasía los recursos hasta hacer casi imposible la navegación. Su efectividad radica en acotar alguna de las dos magnitudes que componen el ancho de banda: los datos o el tiempo. De esta manera, se asigna un límite de carga o descarga de datos o un límite de tiempo de uso del canal. Si además, estos mecanismos vinculan ambas restricciones se obtendrá una herramienta robusta que tribute al aprovechamiento de los recursos de conectividad a Internet en los entornos donde se empleen.

1.5 Metodologías y herramientas a utilizar

Es importante definir desde los primeros momentos, las tecnologías que se utilizarán, tomar una decisión correcta respecto a los lenguajes de programación, herramientas y metodologías que permitirán darle cumplimiento a los objetivos trazados.

1.5.1 Metodología de desarrollo

Las metodologías de desarrollo de *software* son un conjunto de procedimientos, técnicas y estándares para el desarrollo de productos de *software*. Existen metodologías llamadas “tradicionales” como es el caso de RUP¹⁶, que ha sido durante mucho tiempo la metodología de desarrollo más usada. Sin embargo, debido al volumen de documentación que esta genera, ha ganado el rechazo de muchos que han cambiado de paradigma hacia la utilización de metodologías ágiles. El principal acierto de las metodologías ágiles es que enfatizan las comunicaciones cara a cara con el cliente, convirtiéndolo en parte indispensable del proceso de desarrollo, lo que permite acelerar el lanzamiento del producto y disminuir la amplia documentación que se genera con las metodologías

¹⁶ Siglas de *Rational Unified Process*, Proceso Racional Unificado.

tradicionales.

Dentro de las metodologías ágiles de desarrollo de *software* más comúnmente usadas se encuentran XP¹⁷, *Scrum*, *Cristal Methodologies*, *Dynamic Systems Development Method* (DSDM), *Adaptive Software Development* (ASD) y *Feature-Driven Development* (FDD), dentro de este grupo las más usadas son XP y *Scrum*. A diferencia de RUP, donde un cambio en alguna de las etapas definidas, se traduce en un cambio que va a repercutir en el costo, en las metodologías ágiles este impacto se reduce. Sin embargo, RUP brinda una mayor cantidad de ventajas respecto al énfasis que le da a los requisitos y al diseño. Teniendo estas dos etapas del *software* bien definidas, se obtendrá al final un *software* mucho más correcto respecto a la idea original. Otra ventaja que muestra RUP, es que se basa en todas las mejores prácticas que hasta el día de hoy se han probado, por lo cual el desarrollador que implementa bajo esta metodología se encuentra de alguna manera con un soporte mayor que cuando lo realiza con una metodología ágil [15].

El grupo de desarrollo del Smart Keeper posee una gran inestabilidad de permanencia entre sus integrantes, dado que está compuesto en su gran mayoría por estudiantes que cursan años terminales de la carrera. Esta inestabilidad obliga a que todo cambio que se realice sobre el producto sea bien documentado para que el nuevo conocimiento esté accesible y permanezca, a pesar de la ausencia de quien lo haya generado.

Por las razones anteriormente expresadas y teniendo en cuenta que al utilizar RUP se guarda concordancia con la documentación de Smart Keeper se opta por el uso de esta metodología de desarrollo de *software*.

Características principales de RUP:

- ✓ Dirigido por casos de uso: RUP tiene a los Casos de Uso como el enlace guía que orienta las actividades de desarrollo. Se basa en buscar que el sistema a implementar cumpla las necesidades de quien interactúa con el mismo.
- ✓ Centrado en la arquitectura: RUP propone una estructura similar a la de una edificación. Va llevando la construcción del *software* similar al modo en que se realiza la construcción de un edificio desde el comienzo.

¹⁷ Siglas de *eXtreme Programming*, es el más destacado de los procesos ágiles de desarrollo de *software*.

- ✓ Iterativo e incremental: En proyectos grandes, RUP propone dividirlos en mini-proyectos, y cada uno de ellos será una iteración, cada una de las mismas deberá ser correctamente gestionada y tratar un grupo de casos de uso [16].

1.5.2 Lenguajes de programación

El programa *redirector*, eje de Smart Keeper, está construido en C++, además, la solución actual al problema del ancho de banda en Smart Keeper está escrito usando este lenguaje, reutilizar parte de la codificación de estos programas sería una gran ventaja que ahorraría tiempo en la futura implementación de la solución. Estas razones sientan las bases para que, manteniendo una coherencia con lo que ya está implementado en el sistema, y sin ánimos de minimizar la efectividad de otros lenguajes de programación, se enuncie C++ como lenguaje para desarrollar la aplicación propuesta. Algunas de las características que distinguen este lenguaje de programación se describen a continuación:

- ✓ C++ presenta gran funcionalidad en proyectos grandes como Smart Keeper, debido a su estructura orientada a objetos, lo que favorece en gran medida el desarrollo.
- ✓ A su vez presenta una gran ventaja respecto a otros lenguajes como Perl (lenguaje en el que estaba primeramente implementado el *redirector*), PHP o Python debido a ser compilado y la necesidad de búsqueda de un mayor rendimiento para satisfacer las necesidades de los usuarios.
- ✓ Permite dividir en varias partes la solución que se desarrolla, por lo cual varios grupos pueden implementar diversas funcionalidades de un mismo proyecto.
- ✓ La reutilización de código es otra de las potencialidades que ofrece C++, permitiendo disminuir en ocasiones el tiempo de desarrollo.

Smart Keeper cuenta con una interfaz gráfica de usuario que permite el acceso de forma sencilla y amigable a las funcionalidades involucradas en el control y configuración del sistema. Esta interfaz está desarrollada utilizando el *framework* Symfony y el lenguaje PHP. La implementación de un nuevo sistema de cuotas traerá consigo, irremediablemente, cambios en el diseño de la interfaz de administración, debido a su carácter invasivo. De esta forma se justifica el uso del lenguaje PHP para llevar a cabo estas modificaciones. Algunas características de este lenguaje son:

- ✓ Simplicidad: usuarios con experiencia en Perl, C o C++ podrán utilizarlo rápidamente.
- ✓ Velocidad: alta velocidad de ejecución sin introducir demoras en la máquina, bajo consumo de recursos.
- ✓ Estabilidad: posee una amplia comunidad de programadores y usuarios para corregir los errores o bugs que pueden existir.
- ✓ Posee un sofisticado manejo de variables que lo hace muy flexible y estable.
- ✓ Seguridad: permite la protección contra diversos ataques a través de diferentes niveles de seguridad [\[17\]](#).

1.5.3 Entorno de desarrollo

Seleccionar el Entorno Integrado de Desarrollo (IDE, por sus siglas en inglés), es un paso importante en un proyecto de desarrollo. Un IDE puede simplificar mucho la tarea del programador ya que ofrece varias funcionalidades como completamiento de código, chequeo de sintaxis y depuración, contribuyendo a evitar pérdidas de tiempo.

La selección del IDE tuvo en cuenta el enfoque hacia el uso de *software* libre que mantiene Smart Keeper. Además debía cumplir las características de presentar un alto rendimiento con los lenguajes de programación previamente seleccionados, C++ y PHP. Dos de los más importantes entornos integrados de desarrollo que cumplen las características descritas anteriormente son Eclipse y NetBeans.

Optar por el último estuvo avalado por las siguientes características:

- ✓ Chequeo de sintaxis.
- ✓ Presenta soporte para el Sistema de Control de Versiones.
- ✓ Permite el reformato de código, basado en estilos de código que son definidos por el usuario [\[18\]](#).
- ✓ Experiencia de los desarrolladores en el trabajo con la herramienta.
- ✓ Soporte incorporado para el desarrollo con Symfony.

Como desventaja se puede mencionar su uso elevado de memoria, pero no en un nivel que deje de ser aceptable para el usuario, respondiendo también a las características de los ordenadores que se utilizan para desarrollar.

1.5.3.1 Marco de trabajo

Un marco de trabajo o *framework*, como también se le conoce, permite reutilizar un diseño para un dominio específico de *software* rigiendo la arquitectura del mismo. Además define la estructura global de un sistema teniendo en cuenta las clases y objetos, sus responsabilidades claves y como colaboran entre sí, de modo que los diseñadores e implementadores puedan concentrarse en los elementos específicos del mismo [19]. Por último hace la programación más fácil encapsulando operaciones complejas en instrucciones sencillas.

La interfaz de administración de Smart Keeper, fue desarrollada siguiendo las pautas que dicta el *framework* Symfony, elemento preponderante, que obliga a la utilización de este *framework* en el desarrollo de la solución que enmarca esta investigación.

Varias de las características que ostenta este *framework* se mencionan a continuación:

- ✓ Está escrito en PHP5 basándose en el estilo arquitectónico MVC¹⁸.
- ✓ Pueden usarse fácilmente componentes de otros *frameworks*, para proyectos con requisitos cuya solución no está presente en Symfony, lo que brinda junto a otros mecanismos de extensión una gran flexibilidad.
- ✓ Cuenta con una excelente documentación que explica todo lo relativo al trabajo con las diferentes versiones
- ✓ Numerosos *plugins*¹⁹ son creados y actualizados constantemente ante las necesidades de los desarrolladores [20].

1.5.4 Herramienta CASE²⁰

Las herramientas CASE son utilizadas para automatizar o apoyar una o más fases del proceso de desarrollo de *software*. Proporcionan un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño,

¹⁸ Siglas de *Model-View-Control*, Modelo-Vista-Controlador, es un estilo de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos.

¹⁹ Un *plugin* es un módulo de *hardware* o *software* que añade una característica o un servicio específico a un sistema más grande.

²⁰ Siglas de *Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Computadora.

hasta la generación del código fuente de los programas y la documentación. Entre las más utilizadas se encuentran Rational Rose y Visual Paradigm.

Visual Paradigm al igual que Rational Rose son herramientas multiplataforma que utilizan UML²¹ como lenguaje de modelado. Dichas herramientas soportan todos los diagramas UML y la generación de código en diversos lenguajes, permiten desarrollo multiusuario y control de versiones. El hecho de que ambos sean *software* bajo licencias propietarias, plantea una desventaja a tener en cuenta en la elección entre estas herramientas CASE.

Finalmente, optar por Visual Paradigm está dado por el conocimiento previo que los autores poseen del trabajo con esta herramienta, además de que se cuenta con la licencia que este *software* requiere para su instalación, unido a estas razones está la posibilidad de integración de esta herramienta directamente con el IDE seleccionado, lo que permite una perfecta sincronización entre los diagramas y el código fuente de la aplicación [\[21\]](#).

1.5.5 Selección de la solución para el almacenamiento de los datos

El sistema de cuotas que se pretende desarrollar estará sustentado sobre la base del procesamiento en tiempo real de los registros de Squid. Esta herramienta, antes descrita, archiva en sus registros de navegación gran cantidad de líneas que deben ser analizadas. Para que se tenga una idea del volumen de procesamiento, una petición a un sitio web puede generar hasta 100 líneas en un log de navegación de Squid. La selección de la herramienta de almacenamiento debe estar respaldada en el rendimiento como elemento primario.

Smart Keeper guarda los datos que necesita para su funcionamiento en una base de datos PostgreSQL, pero su uso para las funciones que requiere el sistema de cuotas previamente citado ha demostrado no ser la opción adecuada.

Actualmente las bases de datos relacionales no constituyen la más viable o única solución para darle respuesta a cualquier problema que requiere el almacenamiento de información.

Después de realizar una búsqueda exhaustiva se llegó a un consenso respecto a cuatro

²¹ Siglas de *Unified Modeling Language*, lenguaje unificado de modelado. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

soluciones, que al final resultaron las más sugerentes y que a su vez estuvieran más cerca del tipo de solución que se desea ofrecer. Estas fueron Cassandra, Memcached, MongoDB y Redis.

Las soluciones anteriormente relacionadas constituyen ejemplos de bases de datos no relacionales, es decir, transitan en muchos aspectos en sentido opuesto al paradigma propuesto por el modelo relacional. Al movimiento que contiene este tipo de bases de datos se le conoce como NoSQL [\[22\]](#).

Las bases de datos del movimiento NoSQL proponen una conceptualización que se entrelaza con las necesidades de un sistema de cuotas para Smart Keeper. Estos sistemas favorecen una mayor velocidad y rendimiento en el proceso de consulta e inserción de la información. Esto se debe a que la mayoría almacena los datos en la memoria interna del ordenador, lo que hace el acceso a los mismos mucho más rápido. La memoria interna de un ordenador está constituida por la memoria RAM²² y la memoria ROM²³, a diferencia de la memoria secundaria o externa que se compone de la memoria de almacenamiento masivo y permanente, posee una menor capacidad de almacenamiento pero una velocidad de acceso muy superior, es por esta razón que se logra el acceso en un tiempo mucho menor en las bases de datos pertenecientes al movimiento NoSQL.

Este tipo de bases de datos se vislumbra como el próximo nivel en lo que a bases de datos se refiere. Buscan por definición la velocidad, la escalabilidad, buen rendimiento en un entorno distribuido y de código abierto. La escalabilidad en este tipo de base de datos es horizontal, o sea, va linealmente proporcional al número de ordenadores inmersos en la solución.

A continuación se muestran algunas características que presentan de manera general las bases de datos no relacionales o pertenecientes al movimiento NoSQL.

²² Siglas de *Random Access Memory*, memoria de acceso aleatorio. Es la memoria del ordenador que contiene de forma temporal el programa, los datos y los resultados que están siendo usados por el usuario del ordenador. Se denominan "de acceso aleatorio" porque se puede leer o escribir en una posición de memoria con un tiempo de espera igual para cualquier posición, no siendo necesario seguir un orden para acceder a la información de la manera más rápida posible. En general es volátil, pierde su contenido cuando se apaga el ordenador.

²³ Siglas de *Read Only Memory*, memoria de sólo lectura. Viene grabada de fábrica en el ordenador, con una serie de programas.

- ✓ Algunas soluciones como Memcached constituyen base de datos en-memoria, que son aquellas que realizan el almacenamiento en la memoria principal de la computadora, haciendo su lectura y funcionamiento mucho más rápido.
- ✓ Las bases de datos no relacionales proveen de una mayor escalabilidad que las relacionales.
- ✓ El esquema de datos en este tipo de soluciones no es predefinido, o bien es flexible y sencillo de tal manera que se puedan realizar ajustes.
- ✓ Hacen un uso eficiente de la memoria RAM para el almacenamiento.
- ✓ Este tipo de programas por lo general son de código abierto, lo que es de suma importancia para todo desarrollador que no cuenta con los recursos financieros para acometer sus soluciones.
- ✓ En el entorno actual, es indispensable que los sistemas presenten una arquitectura distribuida, es decir, que se apoye en varios ordenadores para brindar su solución, en las bases de datos relacionales, esto ha sido un problema durante su desarrollo, sin embargo el concepto es mucho más aplicable en las bases de datos del movimiento NoSQL porque está presente desde su definición.

Tipos de bases de datos NoSQL:

- ✓ Almacenamiento llave-valor: Este tipo de almacenamiento provee de un índice distribuido para almacenar los objetos. Estos últimos por lo general no son interpretados por el sistema, son manejados por la aplicación como objetos binarios de gran tamaño. Los sistemas que utilizan el almacenamiento llave-valor generalmente proveen la replicación de objetos en caso de que sea necesaria la recuperación por un posible daño al sistema o a la base de datos en específico, mantener los datos en varios ordenadores y la persistencia de objetos. Ejemplos: Redis, Memcached y Riak.
- ✓ Almacenamiento de documentos: Este tipo de sistemas se encuentran en un nivel superior al evidenciado en las bases de datos de tipo llave-valor. En estos se reconoce la estructura de los objetos almacenados. Los mismos son tratados como documentos, por eso comúnmente también, aparte de su denominación de bases de datos orientadas a objetos, se les conoce como orientadas a documentos. Los objetos pueden ser almacenados en colecciones, a las que se le pueden realizar consultas definidas con un mecanismo simple, que en casos como MongoDB no

distan mucho de la sintaxis que se ve en el SQL²⁴ desde el punto de vista de la sencillez. Permiten al igual que los sistemas de tipo llave-valor el almacenamiento de los datos en varios ordenadores, replicar los mismos en caso de que se requiera una recuperación automática pero con un avance en cuanto al manejo de la persistencia de los datos, el cual es mucho más consistente en este tipo de sistemas. Ejemplos de bases de datos NoSQL que usan el almacenamiento de documentos: MongoDB, Dynamo y CouchDB.

- ✓ Almacenamiento extensible: Este tipo de soluciones, a veces es visto como una continuación de lo propuesto por las bases de datos relacionales, debido a que proveen un modelo de datos. Lo que presentan de nuevo este tipo de soluciones es que proveen de un número dinámico de atributos y cómo mismo realizan las bases de datos orientadas a documentos, brindan una alta escalabilidad y disponibilidad con el correspondiente almacenamiento de la base de datos en varios ordenadores. Ejemplos de este tipo de almacenamiento: BigTable, Hbase y Cassandra [\[23\]](#).

Desventajas de las bases de datos NoSQL.

- ✓ Relativamente el tiempo de uso de este tipo de soluciones es corto. Como todo lo nuevo en el ámbito informático, suscita cierto escepticismo con el rendimiento y su consistencia.
- ✓ Mientras que las bases de datos relacionales, generalmente comparten ciertas normas, siguen una manera común de implementarse, las bases de datos no relacionales tienen su propia interfaz de programación, cada una presenta una forma propia de manejar las consultas, lo que convierte la posibilidad de migrar a una base de datos de similar corte desde el punto de vista conceptual en un problema de compatibilidad [\[23\]](#).

Se relacionaba anteriormente a Cassandra, Memcached, Redis y MongoDB como las soluciones que tras la investigación realizada se ajustaban más a la solución que se pretende desarrollar, véase a continuación las principales características de las mismas:

Cassandra:

²⁴ Siglas de *Structured Query Language*, lenguaje estructurado de consultas. Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas.

- ✓ Escrito en Java y con licencia Apache.
- ✓ Presenta lo mejor de las soluciones propietarias Big Table y Dynamo.
- ✓ Escribe más rápido de lo que puede leer [\[24\]](#).

Al ser escrito en Java, si cada componente del sistema es programado en dicho lenguaje, se obtiene un excelente rendimiento. Esta ventaja no podría ser explotada por el sistema de cuotas ya que el núcleo de Smart Keeper está desarrollado en C++, hecho que atenta contra la selección de esta solución de base de datos, a lo que se adiciona su baja velocidad en la lectura de datos lo que la convierte en una herramienta ideal para el registro de grandes cúmulos de información pero poco eficaz cuando se precisa además de una constante lectura de datos para ser procesados.

Memcached:

- ✓ Es libre y a su vez de código abierto, de alto rendimiento y constituye un sistema de caché de objetos en memoria distribuida.
- ✓ Es apropiada para aplicaciones web dinámicas favoreciendo su velocidad, representando un soporte en el que se apoya la base de datos de las mismas para evitar la sobrecarga.
- ✓ Almacena en la memoria interna, de manera que favorece la rapidez con la que el sistema que la usa responde a las solicitudes.
- ✓ Diseño simple que promueve a su vez un rápido despliegue, fácil de implementar y resuelve muchos problemas pudiéndose enfrentar a elevados volúmenes de datos en caché.
- ✓ Presenta una interfaz de programación para los lenguajes de programación más populares como C++, Java entre otros.
- ✓ Su rendimiento le avala para ser usado por varios sitios web de renombre como el canal de videos más popular de Internet Youtube, la red social Twitter y Wikipedia [\[25\]](#).

Entre los aspectos negativos que supondría el uso de Memcached está el hecho de que al almacenar los datos en la memoria interna de la PC, estos no persisten una vez que se reinicia el sistema. La persistencia de los datos es un punto vital en un sistema de cuotas, por lo que el uso de esta base de datos se descarta.

Redis:

- ✓ Al igual que Memcached almacena los datos en la memoria interna y en la forma llave-valor. Se destaca en Redis la persistencia de los datos, que son cargados en la memoria interna cuando es reiniciado el sistema.
- ✓ Soporta varios tipos de datos que lo hacen una variante muy atractiva dentro de toda la gama de soluciones similares, aparte de *strings*, puede soportar también listas y *hashes*.
- ✓ Se ajusta perfectamente a los programas que necesitan implementar una caché, aunque puede funcionar como la base de datos principal [\[24\]](#).

A pesar de ser Redis una solución fiable, pruebas realizadas por el equipo de desarrollo de Smart Keeper para medir el rendimiento de Redis y MongoDB arrojaron como resultado que esta última base de datos funcionaba más rápido, aprovechando el multiprocesamiento, característica que necesita de configuraciones un tanto complejas para ser logradas con Redis. La elección de **MongoDB** está sustentada, al mismo tiempo, sobre la base de las siguientes premisas:

- ✓ Presenta interfaz para varios lenguajes programación como C++, Java, PHP, C# entre otros.
- ✓ Utiliza un fichero asignado en memoria (ocupa una porción de la memoria virtual) como mecanismo de almacenamiento y consulta de la información. Lo anterior hace que delegue gran parte del manejo de la memoria al sistema operativo donde se ejecuta, por lo que el código de MongoDB para gestionar la memoria es pequeño y claro. Las operaciones de entrada y salida son mucho más rápidas con esta variante utilizada por MongoDB.
- ✓ Cuenta con una comunidad estable y amplia, de experiencia en el desarrollo con la herramienta que brinda un espacio para consultar y solucionar los problemas que se presentan en el desarrollo con MongoDB.
- ✓ Mantiene algunas características de SQL que permiten que los usuarios que desarrollan con MongoDB luego de haberlo hecho con SQL perciban de un modo mucho más sencillo el cambio de paradigma [\[26\]](#).

Una de las desventajas que puede presentar MongoDB es el hecho de que para instalarlo y mantenerlo funcional presenta muchas dependencias de librerías, sin embargo el

resultado final es una base de datos muy rápida y potente, lo cual reduce el costo que trae consigo todas las dependencias de las cuales requiere.

1.6 Conclusiones

Después de analizar los inconvenientes de diversos trabajos similares existentes en los ámbitos nacional e internacional, se evidencia la necesidad de concebir una herramienta propia, que extienda las características actuales de los mecanismos para el aprovechamiento del ancho de banda en Smart Keeper. Para ello, se debe tener en cuenta, en primera instancia, características presentes en el sistema de cuotas para la navegación en Internet Squish, ya que su funcionamiento sigue los mismos principios que los implementados hasta el momento en Smart Keeper. Por otra parte, debido a la necesidad de almacenar y consultar datos de forma rápida y eficiente se considera apropiado el uso de una base de datos no relacional, como herramienta para la persistencia de datos. Luego de tratar aspectos de RUP, como una metodología muy práctica para equipos de desarrollo inestables, que propone el predominio de la documentación, se torna ideal su utilización para regir el desarrollo futuro del sistema de cuotas para Smart Keeper.

Llegado este punto se cuenta con las bases teóricas que servirán de sustento para plantear una propuesta de solución contando con las tecnologías y métodos de desarrollo definidas previamente.

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

El desarrollo de un *software* parte de comprender la problemática a la que se quiere dar solución. En RUP, la fase de elaboración permite lograr este fin a través del levantamiento de requisitos y la realización del modelo del negocio o dominio. En el presente capítulo se describe detalladamente el funcionamiento actual del sistema y se presenta una propuesta de solución que permitirá fortalecer y flexibilizar los mecanismos de control del ancho de banda de Smart Keeper. Se detallan los requisitos de *software*, tanto funcionales como no funcionales, se modela el diagrama de casos de uso del sistema y se describen textualmente dichos casos de uso.

2.1 Descripción del problema

En el capítulo anterior se hacía mención a la necesidad de implementar un sistema de cuotas para el Filtro de Contenido Web Smart Keeper que permita el control del tráfico de datos y el tiempo de conexión de los usuarios. Actualmente, el sistema dispone, exclusivamente, de una base de datos PostgreSQL, que contiene todos los elementos que requiere para su correcto funcionamiento. Dentro de toda esta información se encuentran los datos relacionados con los usuarios que navegan a través del sistema y los correspondientes a los recursos solicitados por estos.

Ante la solicitud de un recurso de Internet por parte de un usuario, el servidor *proxy* Squid, que es parte inseparable del sistema, genera una entrada por cada componente web descargado, en su archivo de registros. Por cada entrada, se obtiene el usuario que realizó la petición y el tamaño en bytes del componente descargado. El programa *syslog2db*, que está construido en C++, es el encargado de utilizar los valores anteriormente mencionados e interactuar con la base de datos. Este programa ejecuta de forma constante una serie de consultas de escritura a la base de datos para adicionar el tamaño de los componentes descargados al usuario correspondiente. El proceso se realiza analizando una entrada del archivo de registros a la vez, por cuanto las entradas son generadas a medida que el servidor *proxy* Squid le da solución a la petición. Las consultas de escrituras, en los gestores de bases de datos, son las que más penalizan el desempeño del sistema. El proceso de análisis de las peticiones es directamente proporcional a la cantidad de usuarios que utilizan el sistema, por lo que en entornos de alta demanda el sistema vería afectado su rendimiento.

El programa *redirector*, núcleo del sistema, se nutre de la respuesta independiente que dan una serie de *plugins*, cada uno con diferente funcionalidad, para tomar la decisión de satisfacer o denegar la solicitud del usuario. Uno de los *plugins* de los que el *redirector* espera respuesta, es el que determina el estado de uso de la cuota de los usuarios. Este *plugin* se conecta a la base de datos para solicitar los datos que requiere y realiza una lógica interna para gestionar su respuesta. Esto incorpora un conjunto de consultas de lectura a las base de datos que se suman al proceso final de la respuesta.

La realidad es que estos procesos retrasan la respuesta final del sistema a las peticiones de los usuarios, lo cual puede ocasionar molestias por la repercusión que tienen en la velocidad con que se brinda el servicio de navegación. Se suma a esto la carencia en el sistema de algún mecanismo para determinar el tiempo que el usuario permanece navegando. Por tanto, la solución debe enfocarse principalmente a favorecer la rapidez en la gestión de la cuota, a la disminución del tiempo de respuesta del *plugin* que determina el estado de uso la cuota de los usuarios así como la implementación de un mecanismo que determine el tiempo de conexión de los usuarios, que no sería otra el sistema de cuotas a implementar.

2.2 Modelo de Dominio

Debido a la relativa simplicidad del entorno donde está enmarcado el sistema y la comprensión que se posee de su funcionamiento es posible prescindir de la realización de un Modelo de Negocio para interpretar el problema que ha de resolverse, siendo un Modelo de Dominio o Conceptual suficiente. Este se define como la representación visual de los conceptos u objetos del mundo real significativos en el contexto del problema o área de interés. Es importante destacar que en él se señalan las clases conceptuales del dominio del problema, conceptos del mundo real, no de componentes de *software*. La siguiente imagen muestra el Modelo de Dominio realizado teniendo en cuenta lo planteado en la descripción del problema.

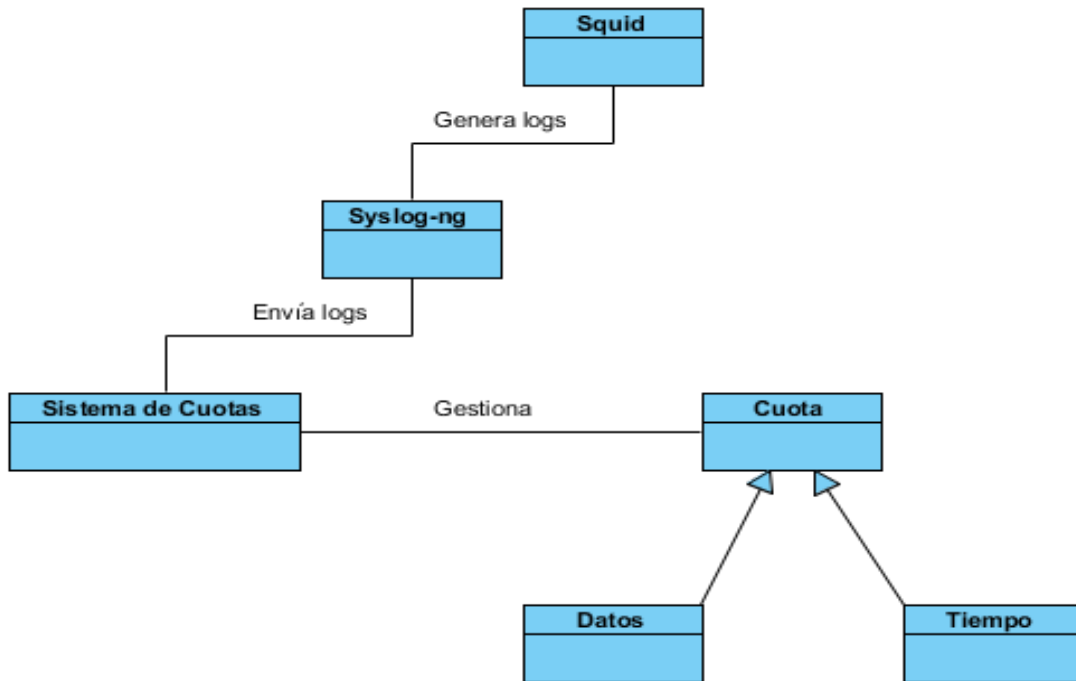


Figura 1: Modelo de dominio.

Squid: Representa al servidor *proxy-caché* Squid que genera registros por cada solicitud de los usuarios. Los registros son capturados y enviados para ser analizados.

Syslog-ng: Representa el sistema externo que envía al Sistema de Cuotas, por la entrada estándar, los datos que recibe de Squid. Sirve como una especie de interfaz entre los dos componentes.

Sistema de Cuotas: Representa el sistema de cuotas que será desarrollado, este recibirá datos por la entrada estándar y ejecutará una lógica interna realizando consultas en una colección de MongoDB.

Cuota: Representa, de forma genérica, los datos de la cuota que serán almacenados usando la base de datos MongoDB, donde estarán todos los elementos que el sistema de cuotas necesitará para su funcionamiento.

Datos: Representa, de forma específica, los datos descargados por los usuarios en su navegación a través del sistema.

Tiempo: Representa, de forma específica, el tiempo que los usuarios se mantienen en línea durante su navegación a través del sistema.

2.3 Solución propuesta

La propuesta para dar solución al problema descrito con anterioridad consiste en la implementación de una serie de mecanismos que propicien la obtención del tiempo de conexión de los usuarios así como las tasas de descarga de componentes web de estos a través del sistema de forma eficiente, para este fin se prevé aprovechar la rapidez en el acceso a datos de la base de datos MongoDB.

La premisa que posibilita controlar el tiempo de navegación de los usuarios es considerar que un usuario se mantiene navegando mientras el intervalo de tiempo entre cada una de sus peticiones no exceda un período de inactividad de cinco minutos, que puede ser configurado según las necesidades de los clientes.

Los detalles de cada petición serán obtenidos de los registros de navegación del servidor *proxy* Squid y enviados al sistema de cuotas por el servidor de registros Syslog-ng. De estos registros serán ignorados aquellos que incumplan con el patrón definido por Syslog-ng, los que tengan códigos inválidos²⁵ de Squid y los que no ofrezcan datos como: el usuario y la política de navegación.

El algoritmo para contar este tiempo de navegación consiste en verificar el intervalo de tiempo transcurrido entre el instante en que se realizó la última petición servida al usuario y el instante de la petición recién resuelta, si este tiempo excede el período de inactividad, solo se acumulará el tiempo que tomó resolver la petición, de otra manera se acumula el tiempo entre las peticiones, no sin antes comprobar que el tiempo de la transacción sea menor que el intervalo de tiempo entre las peticiones, lo que supondría que la transacción estaba pendiente y se acumularía solamente la diferencia de tiempo entre el tiempo que tomó resolver la petición recién resuelta y el intervalo de tiempo entre las peticiones, de ser dicho intervalo de tiempo menor que el período de inactividad, de lo contrario tendría que acumularse la diferencia entre el tiempo que tomó resolver la petición recién resuelta y el último período de navegación ininterrumpido. El algoritmo también se encargará de almacenar el tamaño en bytes de cada componente web descargado al usuario correspondiente.

²⁵ Los códigos de respuesta HTTP de Squid para un objeto transferido, son un criterio importante para determinar si la transferencia se realizó con éxito o de lo contrario conocer el error. Los códigos diferentes de 200 suponen algún tipo de error en la transferencia del objeto.

Otro algoritmo se encargará de chequear el estado de consumo de la cuota, analizando la cuota asignada y deshabilitando la posibilidad de navegación al usuario que haya excedido su consumo. En este caso se almacenará en un valor lógico la inhabilitación, o no, de la navegación y en caso de estar deshabilitada, el período de asignación afectado para su posterior reinicio.

Para realizar todo este proceso se hace necesaria la persistencia de un conjunto de datos. Los datos que sean necesarios consultar estarán estructurados en una “colección” de MongoDB. Esta colección debe contar con datos que identifiquen al usuario, además de los valores de tiempo de navegación y datos descargados para los diferentes períodos de asignación (día, semana, mes y año), el instante de la última petición resuelta, el último período de navegación ininterrumpida, un valor lógico que indique el estado de uso de la cuota y en caso de exceder la cuota asignada el período de asignación afectado.

Es importante señalar tres puntos significativos:

1. Los datos de los usuarios de la colección estarán sincronizados con los almacenados en la base de datos PostgreSQL, que no dejará de ser el soporte principal del núcleo del sistema, para no crear inconsistencias y redundancias en el funcionamiento.
2. La definición de la cuota tendrá cambios significativos, dejará de ser un valor numérico almacenado en la tabla usuario en la base de datos PostgreSQL y pasará a ser una cadena de caracteres que se almacenará en la base de datos MongoDB, de la forma: valor numérico seguido de la unidad de medida, que podrá ser de datos o de tiempo, entre (/) el período de asignación.
3. Se podrán asignar diferentes restricciones, separadas por espacio, y combinarlas para diferentes períodos de asignación, sin repetir para el mismo período de asignación restricciones con el mismo tipo de unidad de medida.

Luego de cumplido cada período de asignación se reiniciarán los valores de tiempo de navegación y datos descargados correspondientes a estos. Los usuarios que hayan excedido en más de cinco megabytes o cinco minutos, para los datos y el tiempo respectivamente, el consumo de su cuota serán penalizados y verán afectado el reinicio de su consumo con el valor de la diferencia entre lo consumido en el período de asignación recién finalizado y el valor asignado. En caso de que al reiniciarse los valores de consumo un usuario siga excediendo el valor asignado, la navegación permanecerá

deshabilitada hasta pasado nuevamente el período de asignación.

Para comprender de una manera más sencilla algunos elementos de la solución que se pretende brindar, se deben conocer los conceptos relativos a la base de datos MongoDB como el de “colección”. En MongoDB, una colección es un grupo de documentos y un documento no es más que un grupo ordenado de llaves con valores asociados. La representación de un documento difiere en dependencia del lenguaje de programación, sin embargo, la mayoría de los lenguajes tienen estructuras de datos que se ajustan naturalmente a este concepto como pueden ser los mapas, arreglos asociativos o diccionarios [\[27\]](#). En Javascript, por ejemplo, un documento se puede representar como objeto de la forma que se muestra a continuación:

```
{“saludo”: “Hola Mundo!”}
```

Para implementar la solución, se usarán las interfaces de aplicación para el manejo de datos que MongoDB provee para C++ y PHP, como también la presenta para otros lenguajes conocidos como Java y C#. La interfaz de PHP representa un documento de MongoDB en forma de arreglo asociativo mientras que en la interfaz de C++, un documento de MongoDB queda representado como un objeto BSON de la siguiente manera:

```
BSONObj obj = new BSONObj(“saludo” << “Hola Mundo!”);
```

BSON es la forma abreviada para hacer referencia a Binary JSON (*Javascript Object Notation*). Como JSON, un ligero formato de intercambio de datos, fácil de leer y escribir, BSON ofrece la posibilidad de embeber documentos y arreglos en otros documentos y arreglos. Sin embargo, BSON presenta, teniendo en cuenta algunas características, mayor complejidad que JSON, una de ellas es la presencia de algunas representaciones de tipos de dato que no están presentes en la definición de JSON, como Date y BinData, para fecha y datos binarios respectivamente [\[28\]](#).

Los requisitos funcionales y no funcionales que se presentan a continuación brindarán más claridad de la solución que se propone.

2.4 Requisitos funcionales y no funcionales

2.4.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. A continuación se relacionan los requisitos funcionales del sistema de cuotas para Smart Keeper:

R1. Consultar la cuota consumida

R2. Adicionar valores de consumo

R3. Cambiar el estado de la cuota

R4. Reiniciar la cuota total a un usuario

R5. Reiniciar la cuota diaria

R6. Reiniciar la cuota semanal

R7. Reiniciar la cuota mensual

R8. Reiniciar la cuota anual

R9. Reiniciar el consumo de datos de un usuario en un día

R10. Reiniciar el consumo de datos de un usuario en una semana

R11. Reiniciar el consumo de datos de un usuario en un mes

R12. Reiniciar el consumo de datos de un usuario en un año

R13. Reiniciar el tiempo en línea de un usuario en un día

R14. Reiniciar el tiempo en línea de un usuario en una semana

R15. Reiniciar el tiempo en línea de un usuario en un mes

R16. Reiniciar el tiempo en línea de un usuario en un año

2.4.2 Requisitos no funcionales

Los requisitos no funcionales especifican propiedades del sistema y, generalmente, se asocian a ciertos casos de uso. A continuación se relacionan los requisitos no funcionales del sistema de cuotas para Smart Keeper:

Fiabilidad

- ✓ El sistema debe estar disponible las 24 horas del día cada uno de los 7 días de la semana toda vez que está intrínsecamente unido a Smart Keeper.
- ✓ Si el sistema de cuotas falla, Smart Keeper debe continuar trabajando.
- ✓ Los datos en la base de datos MongoDB deben tener correspondencia con los que se encuentran en la base de datos en PostgreSQL.

Disponibilidad

Para que el sistema funcione de manera correcta y no tenga problemas de disponibilidad deben estar creadas las estructuras en el sistema de cuotas para que pueda ser consultado.

Seguridad

La seguridad es un tema de suma importancia para cualquier *software* que se desarrolle, en el caso que se presenta, donde se implementará un sistema para un *software* ya existente, este debe contar con un mecanismo de seguridad por cuanto puede decidir en la habilitación o no de la navegación de los usuarios.

- ✓ El acceso a los datos de las cuotas estará protegido por contraseña utilizando la seguridad que brinda MongoDB.

Soporte

- ✓ Se seguirán los estándares de codificación utilizados por las bibliotecas y *frameworks* que se utilicen en la solución.

Restricciones de diseño

- ✓ Se utilizará C++ para la lógica del sistema de cuotas y se almacenarán los datos usando MongoDB.
- ✓ Deben utilizarse herramientas de desarrollo libre, de manera que se corresponda todo el ambiente de trabajo con el perfil que mantiene Smart Keeper.
- ✓ Se utilizará la versión 1.8.4 de MongoDB, última versión estable disponible en el momento de la investigación.

2.5 Diagrama de Casos de Uso del Sistema

2.5.1 Definición de los actores

Actor	Descripción
Usuario	Persona que tendrá acceso al sistema como un usuario básico.
Administrador	Persona que podrá ejecutar todas las funcionalidades disponibles relacionadas con el rol de administración.
Syslog-ng	Syslog-ng envía los registros que genera Squid al sistema para ser procesados.
Crontab	Crontab ordena la ejecución de las tareas programadas en el sistema operativo.

Tabla 2: Definición de los actores del sistema.

2.5.2 Listado de casos de uso

Caso de Uso	Consultar la cuota consumida
Actores	Administrador, Usuario
Resumen	El Caso de Uso se inicia cuando el actor consulta los detalles y el sistema muestra los datos de la cuota consumida por día, semana, mes y año.
Referencia	R1

Tabla 3: Definición de CU: Consultar la cuota consumida.

Caso de Uso	Adicionar valores de consumo
Actores	Syslog-ng

Resumen	El Caso de Uso se inicia cuando Syslog-ng envía un registro de navegación al sistema y se acumula el consumo al usuario correspondiente.
Referencia	R2

Tabla 4: Definición de CU: Adicionar valores de consumo.

Caso de Uso	Cambiar el estado de la cuota
Actores	Syslog-ng
Resumen	El Caso de Uso se inicia cuando Syslog-ng envía un registro de navegación al sistema y se verifica que el usuario no haya excedido el uso de su cuota.
Referencia	R3

Tabla 5: Definición de CU: Cambiar el estado de la cuota.

Caso de Uso	Reiniciar cuota total a usuario
Actores	Administrador
Resumen	El Caso de Uso inicia cuando el administrador selecciona la opción de reiniciar la cuota del período afectado.
Referencia	R4

Tabla 6: Definición de CU: Reiniciar cuota total a usuario.

Caso de Uso	Reiniciar cuota diaria
Actores	Crontab
Resumen	El Caso de Uso inicia cuando el Crontab ejecuta una tarea programada que reinicia la cuota diaria de los usuarios.
Referencia	R5

Tabla 7: Definición de CU: Reiniciar cuota diaria.

Caso de Uso	Reiniciar cuota semanal
Actores	Crontab

Resumen	El Caso de Uso inicia cuando el Crontab ejecuta una tarea programada que reinicia la cuota semanal de los usuarios.
Referencia	R6

Tabla 8: Definición de CU: Reiniciar cuota semanal.

Caso de Uso	Reiniciar cuota mensual
Actores	Crontab
Resumen	El Caso de Uso inicia cuando el Crontab ejecuta una tarea programada que reinicia la cuota mensual de los usuarios.
Referencia	R7

Tabla 9: Definición de CU: Reiniciar cuota mensual.

Caso de Uso	Reiniciar cuota anual
Actores	Crontab
Resumen	El Caso de Uso inicia cuando el Crontab ejecuta una tarea programada que reinicia la cuota anual de los usuarios.
Referencia	R8

Tabla 10: Definición de CU: Reiniciar cuota anual.

Caso de Uso	Reiniciar cuota a un usuario según parámetro
Actores	Administrador
Resumen	El Caso de Uso inicia cuando el administrador introduce los datos del usuario y el parámetro para reiniciar su cuota.
Referencia	R9, R10, R11, R12, R13, R14, R15, R16

Tabla 11: Definición de CU: Reiniciar cuota un usuario según parámetro.

2.5.3 Diagramas de casos de uso

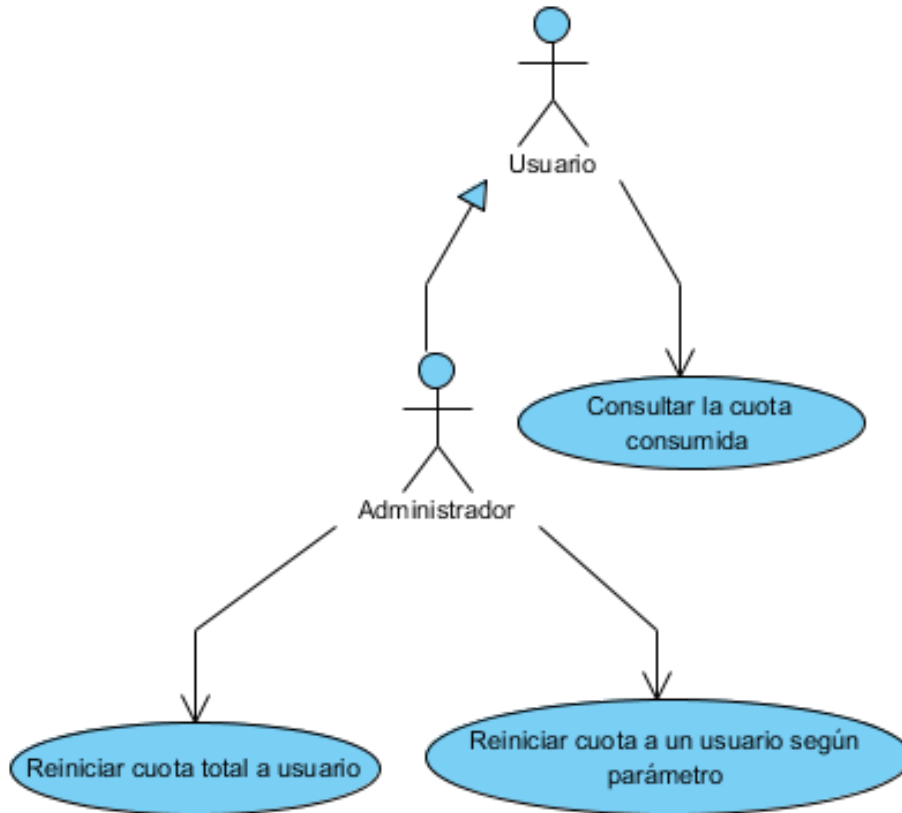


Figura 2: Diagrama de Casos de Uso del Sistema (Usuario y Administrador).

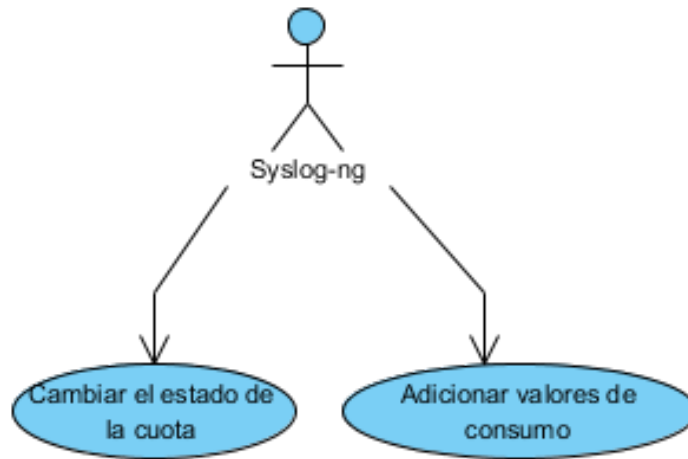


Figura 3: Diagrama de Casos de Uso del Sistema (Syslog-ng).

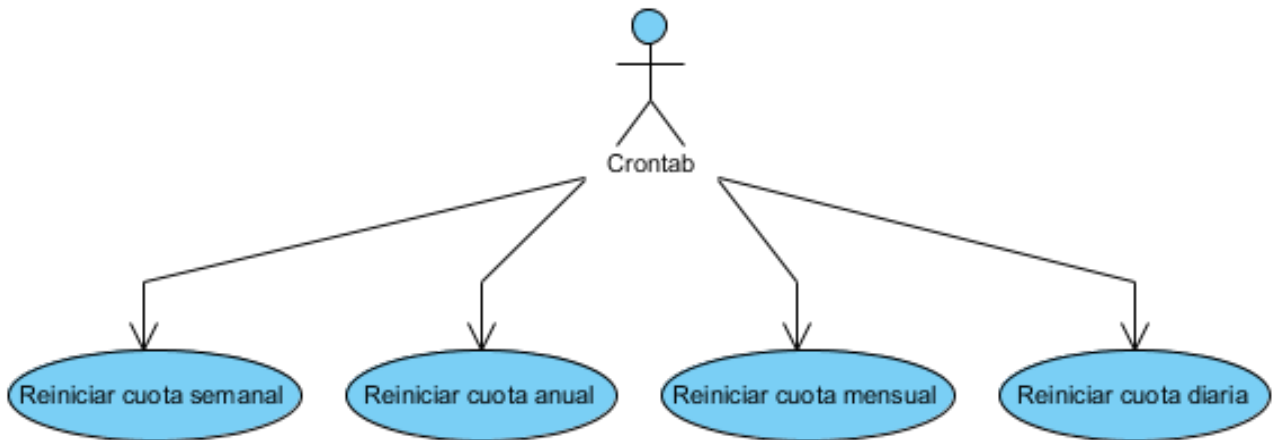


Figura 4: Diagrama de Casos de Uso del Sistema (Crontab).

2.5.4 Casos de uso expandidos

Consultar la cuota consumida

Caso de Uso:	Consultar la cuota consumida
Actores:	Administrador, Usuario

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

Resumen:	El Caso de Uso se inicia cuando el actor consulta los detalles y el sistema muestra los datos de la cuota consumida por día, semana, mes y año.	
Precondiciones:	Que el actor se haya autenticado como usuario del sistema.	
Referencias	R1	
Complejidad	Media	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
1. El usuario ejecuta la opción de ver detalles.	1.1. El sistema muestra los detalles del consumo de la cuota del usuario.	
Flujo Alternativo		
Acción del Actor	Respuesta del sistema	
	1.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema muestra un mensaje indicando el fallo.	
Postcondiciones		

Tabla 12: Descripción CU: Consultar la cuota consumida.

Adicionar valores de consumo

Caso de Uso:	Adicionar valores de consumo	
Actores:	Syslog-ng	
Resumen:	El Caso de Uso se inicia cuando Syslog-ng envía un registro de navegación al sistema y se acumula el consumo al usuario correspondiente.	
Precondiciones:	El servidor Syslog-ng debe estar funcionando.	
Referencias	R2	
Complejidad	Alta	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
1. Syslog-ng envía un registro de	1.1 El sistema recibe los registros por la entrada	

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

navegación al sistema.	estándar. 1.2 Se procesan los registros y se adiciona los valores de la cuota al usuario indicado.
Flujo Alterno	
Acción del Actor	Respuesta del sistema
	1.2 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro cuota.log.
Postcondiciones	Se acumula la cuota consumida.

Tabla 13: Descripción CU: Adicionar valores de consumo.

Cambiar el estado de la cuota

Caso de Uso:	Cambiar el estado de la cuota
Actores:	Syslog-ng
Resumen:	El Caso de Uso se inicia cuando Syslog-ng envía un registro de navegación al sistema y se verifica que el usuario no haya excedido el uso de su cuota.
Precondiciones:	El servidor Syslog-ng debe estar funcionando.
Referencias	R3
Complejidad	Alta
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. Syslog-ng envía un registro de navegación al sistema.	1.1 El sistema recibe los registros por la entrada estándar. 1.2 Se analizan los registros y se verifica si la cuota excede los valores asignados. 1.3 En caso de que exceda, se actualiza el estado y se almacena el período.
Flujo Alterno	
Acción del Actor	Respuesta del sistema
	1.2 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

	registro de error en el archivo de registro quota.log. 1.3 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro quota.log.
Postcondiciones	El estado de la cuota del usuario correspondiente queda actualizado

Tabla 14: Descripción CU: Cambiar el estado de la cuota.

Reiniciar cuota total a usuario

Caso de Uso:	Reiniciar cuota total a usuario	
Actores:	Administrador	
Resumen:	El Caso de Uso inicia cuando el administrador selecciona la opción de reiniciar la cuota total consumida por un usuario.	
Precondiciones:	Que el actor se haya autenticado como administrador del sistema.	
Referencias	R4	
Complejidad	Alta	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
1. El Actor selecciona la opción Reiniciar Cuota de un usuario	1.1. El sistema reinicia todos los valores de la cuota consumida del usuario.	
Flujo Alternativo		
Acción del Actor	Respuesta del sistema	
	1.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema muestra un mensaje indicando el fallo.	
Postcondiciones	La cuota consumida queda reiniciada.	

Tabla 15: Descripción CU: Reiniciar cuota total a usuario.

Reiniciar cuota diaria

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

Caso de Uso:	Reiniciar cuota diaria
Actores:	Crontab
Resumen:	El Caso de Uso inicia cuando el Crontab ejecuta una tarea programada que reinicia la cuota diaria de los usuarios.
Precondiciones:	El servicio cron de Unix debe estar funcionando.
Referencias	R5
Complejidad	Alta
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Actor ejecuta una tarea programada para reiniciar la cuota diaria.	<p>1.1. El sistema consulta todos los usuarios.</p> <p>1.2 El sistema verifica la cuota consumida por cada usuario en el día.</p> <p>1.3 El sistema reinicia la cuota diaria de cada usuario.</p>
Flujo Alterno	
Acción del Actor	Respuesta del sistema
	<p>1.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro cuota.log.</p> <p>1.2 A cada usuario que excedió su cuota en más de 5 megabytes o 5 minutos, para los datos y el tiempo respectivamente, se le penaliza el reinicio de la cuota con el valor del exceso.</p> <p>1.3 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro cuota.log.</p>
Postcondiciones	La cuota diaria de todos los usuarios queda reiniciada.

Tabla 16: Descripción CU: Reiniciar cuota diaria.

Reiniciar cuota semanal

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

Caso de Uso:	Reiniciar cuota semanal
Actores:	Crontab
Resumen:	El Caso de Uso inicia cuando el Crontab ejecuta una tarea programada que reinicia la cuota semanal de los usuarios.
Precondiciones:	El servicio cron de Unix debe estar funcionando.
Referencias	R6
Complejidad	Media
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Actor ejecuta una tarea programada para reiniciar la cuota semanal.	<p>1.1. El sistema consulta todos los usuarios.</p> <p>1.2 El sistema verifica la cuota consumida por cada usuario en la semana.</p> <p>1.3 El sistema reinicia la cuota semanal de cada usuario.</p>
Flujo Alternativo	
Acción del Actor	Respuesta del sistema
	<p>1.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro quota.log.</p> <p>1.2 Si un usuario excedió su cuota, el sistema penaliza el reinicio de la cuota con el valor del exceso.</p> <p>1.3 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro quota.log.</p>
Postcondiciones	La cuota semanal de todos los usuarios queda reiniciada.

Tabla 17: Descripción CU: Reiniciar cuota semanal.

Reiniciar cuota mensual

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

Caso de Uso:	Reiniciar cuota mensual
Actores:	Crontab
Resumen:	El Caso de Uso inicia cuando el Crontab ejecuta una tarea programada que reinicia la cuota mensual de los usuarios.
Precondiciones:	El servicio cron de Unix debe estar funcionando.
Referencias	R7
Complejidad	Media
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Actor ejecuta una tarea programada para reiniciar la cuota mensual.	<p>1.1. El sistema consulta todos los usuarios.</p> <p>1.2 El sistema verifica la cuota consumida por cada usuario en el mes.</p> <p>1.3 El sistema reinicia la cuota mensual de cada usuario.</p>
Flujo Alterno	
Acción del Actor	Respuesta del sistema
	<p>1.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro cuota.log.</p> <p>1.2 Si un usuario excedió su cuota, el sistema penaliza el reinicio de la cuota con el valor del exceso.</p> <p>1.3 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro cuota.log.</p>
Postcondiciones	La cuota mensual de todos los usuarios queda reiniciada.

Tabla 18: Descripción CU: Reiniciar cuota mensual.

Reiniciar cuota anual

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

Caso de Uso:	Reiniciar cuota anual
Actores:	Crontab
Resumen:	El Caso de Uso inicia cuando el Crontab ejecuta una tarea programada que reinicia la cuota anual de los usuarios.
Precondiciones:	El servicio cron de Unix debe estar funcionando.
Referencias	R8
Complejidad	Baja
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Actor ejecuta una tarea programada para reiniciar la cuota anual.	<p>1.1. El sistema consulta todos los usuarios.</p> <p>1.2 El sistema verifica la cuota consumida por cada usuario en el año.</p> <p>1.3 El sistema reinicia la cuota anual de cada usuario.</p>
Flujo Alterno	
Acción del Actor	Respuesta del sistema
	<p>1.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro cuota.log.</p> <p>1.2 Si un usuario excedió su cuota, el sistema penaliza el reinicio de la cuota con el valor del exceso.</p> <p>1.3 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema guarda un registro de error en el archivo de registro cuota.log.</p>
Postcondiciones	La cuota anual de todos los usuarios queda reiniciada.

Tabla 19: Descripción CU: Reiniciar cuota anual.

Reiniciar cuota a un usuario según parámetro

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

Caso de Uso:	Reiniciar cuota a un usuario según parámetro	
Actores:	Administrador	
Resumen:	El Caso de Uso inicia cuando el administrador introduce los datos del usuario y el parámetro para reiniciar su cuota.	
Precondiciones:	El actor debe haberse autenticado como Administrador del sistema.	
Referencias	R9, R10, R11, R12, R13, R14, R15, R16	
Complejidad	Alta	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
1. El Actor selecciona la opción ver detalles de un usuario.	1.1. El sistema muestra los detalles del consumo de la cuota del usuario.	
2. El Actor selecciona el parámetro para reiniciar la cuota consumida (datos o tiempo) del usuario en un período.	2.1. El sistema reinicia la cuota consumida según el parámetro (datos o tiempo en un período determinado). 2.2 El sistema muestra un mensaje de confirmación.	
Flujo Alternativo		
Acción del Actor	Respuesta del sistema	
	1.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema muestra un mensaje de error al mostrar detalles. 2.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema muestra un mensaje indicando el fallo. 2.1 Si la base de datos MongoDB presenta problemas de disponibilidad, el sistema muestra un mensaje indicando el fallo.	
Postcondiciones	La cuota consumida (datos descargados o tiempo en línea) del usuario es reiniciada en el período elegido.	

Tabla 20: Descripción CU: Reiniciar cuota a un usuario según parámetro.

2.6 Conclusiones

A partir de la descripción del problema fue posible identificar y enunciar los principales requerimientos que debe cumplir el sistema de cuotas y generar consecuentemente los diagramas de Casos de Uso del sistema. Las descripciones de Casos de Uso propuestas servirán como base para fases posteriores por las que transitará la solución, describiendo cómo se realiza el flujo de actividades. Después de definir las características del sistema de cuotas se está en condiciones de efectuar el Análisis y Diseño a partir de las mismas. Además, se concluye que el sistema propuesto podrá cumplir satisfactoriamente su labor dentro de Smart Keeper.

CAPÍTULO III: ANÁLISIS Y DISEÑO

Aunque en la metodología RUP el análisis y diseño de una aplicación se encuentra presente en sus cuatro fases, es en la fase de elaboración donde cobran mayor relevancia. En este sentido, el uso de UML como lenguaje de modelado ofrece la notación gráfica precisa para representar los sucesivos modelos que se obtienen en el proceso de refinamiento de la solución. Asociados a los flujos de trabajo de la metodología RUP, existen varios artefactos que debido a su importancia constituyen la base para las fases posteriores, aunque la generación de estos está motivada por las necesidades y objetivos finales de cada proyecto. En este capítulo, de la solución propuesta, se presentan: sus diagramas de clases del análisis, mostrando las clases participantes y relaciones entre ellas; sus clases del diseño y sus diagramas de secuencia, evidenciando el orden de las acciones en los casos de uso cuando son invocados.

3.1 Análisis

3.1.1 Modelo del Análisis

El Modelo de Análisis es la primera representación técnica de un sistema y ofrece una descripción más precisa de los requisitos con el objetivo de refinarlos y estructurarlos. Es importante señalar, entre los aspectos fundamentales del Modelo de Análisis, que se describe en el lenguaje de los desarrolladores, aportando una visión general del sistema mediante los diagramas de Clases del Análisis.

Los diagramas de clases de análisis representan un modelo conceptual temprano que describe las características y comportamientos comunes de un conjunto de elementos que existen en el sistema. Se expresa que es conceptual pues pospone todos los elementos de diseño al no considerar posibles tecnologías a emplear en el desarrollo del *software*. Las clases del análisis están siempre identificadas con uno de los tres estereotipos existentes, los cuales son:

Interfaz: Se encargan de la modelación de toda la interacción que puede existir entre los actores y el sistema.

Control: Representan la coordinación, secuenciación, transacciones y a veces la lógica

del negocio; se emplean a menudo para encapsular el control referido a un Caso de Uso.

Entidad: Representa la información de larga duración y a menudo persistente que se maneja en el sistema [29].

A continuación se muestran algunos de los diagramas de clases del análisis que componen el modelo de análisis del sistema (para consultar los restantes diagramas de clases del análisis ver [Anexo B](#)):

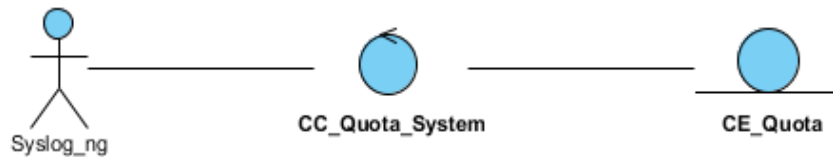


Figura 5: Diagrama de Clases del Análisis: CU Cambiar el estado de la cuota.



Figura 6: Diagrama de Clases del Análisis: CU Reiniciar cuota diaria.

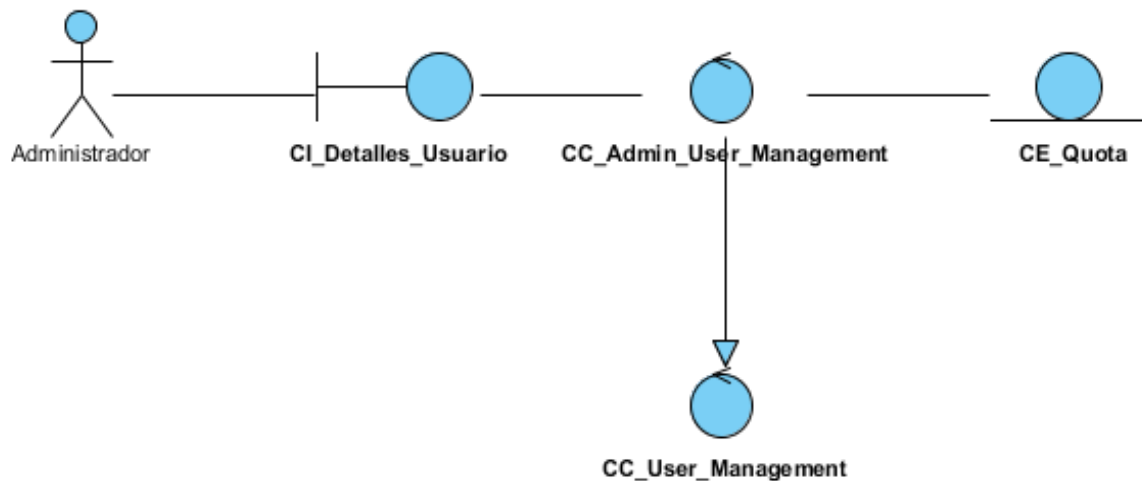


Figura 7: Diagrama de Clases del Análisis: CU Reiniciar cuota a un usuario según parámetro.

3.2 Diseño

En la metodología RUP el diseño cobra relevancia al final de la fase de elaboración y el comienzo de las iteraciones de construcción. El Modelo de Diseño, se define como el modelo de objetos que detalla la realización física de los Casos de Uso. Su importancia se sustenta en que hace notar el impacto que tendrán tanto los requisitos funcionales como los no funcionales en el sistema.

3.2.1 Definición de la arquitectura

La arquitectura de *software* es, a grandes rasgos, una vista del sistema que incluye los principales componentes del mismo, la conducta de esos componentes desde la visión del resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar los objetivos del mismo. La arquitectura de un sistema consiste en la vista conceptual de toda su estructura [\[30\]](#).

Teniendo en cuenta que la implementación de un nuevo sistema de cuotas para Smart Keeper no invade en forma alguna los paradigmas que este sistema propone, sino que se integra en toda medida a estos, es posible establecer, de forma explícita, que la arquitectura que presenta es una arquitectura Cliente-Servidor.

La arquitectura Cliente-Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

3.2.1.1 Estilos arquitectónicos

Las soluciones arquitectónicas que son comunes y reusables a lo largo de años de

experiencia se han ido agrupando en lo que más tarde se les llamó estilos. El éxito del diseño de la arquitectura de *software* depende de los estilos que se decidan utilizar para el desarrollo de la misma. Los estilos expresan la arquitectura en el sentido más formal y teórico, describen entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Una vez que se han identificado los estilos, es lógico y natural pensar en reutilizarlos en situaciones semejantes que se presenten en el futuro. [\[31\]](#).

Durante el diseño de la presente investigación se decidió el uso de dos estilos arquitectónicos, el estilo centrado en datos y el estilo MVC.

En el centro de las arquitecturas centradas en datos se encuentra un almacén de datos (por ejemplo un documento o una base de datos) al que otros componentes acceden con frecuencia para actualizar, añadir, borrar o bien modificar los datos del almacén. El *software* cliente accede a un almacén central. En algunos casos, el almacén de datos es pasivo, esto significa que el *software* cliente accede a los datos independientemente de cualquier cambio en los datos o de las acciones de otro agente [\[31\]](#).

El sistema de cuotas a implementar, en su integración a Smart Keeper supone cambios en la interfaz de administración de dicho producto, por cuanto la asignación de cuotas y posterior consulta del consumo de las mismas, por mencionar algunas acciones del dominio de este nuevo sistema, deben resultar tareas de fácil realización para aquellos que utilicen el sistema de filtrado. El hecho de que la interfaz de administración de Smart Keeper este implementada usando Symfony, como marco de trabajo, explica el uso del estilo MVC.

Symfony ha sido escrito completamente en lenguaje PHP con el objetivo de aprovechar todas las ventajas del lenguaje y se basa en el estilo arquitectónico MVC, implementándolo de modo que el desarrollo de aplicaciones web sea rápido y sencillo.

En esta variante, el controlador que contiene el código que asocia la lógica de negocio con la presentación está compuesto por el controlador frontal, que es el único punto de entrada a la aplicación y a las acciones que se encargan de verificar la validez de las peticiones y preparar los datos para la vista. Esta última está formada por el *layout* que posee el código común a todas las acciones, las plantillas que contienen el código

asociado a cada acción en particular y la lógica que se puede manipular a través de un fichero de configuración sencillo sin necesidad de programarla. Por último, el modelo está formado por la capa de acceso a datos cuyas clases son generadas automáticamente por el ORM²⁶ Propel en función de la estructura de datos de la aplicación y por la capa de abstracción de la base de datos PDO²⁷, que es completamente transparente al programador; así si se cambia el sistema gestor de bases de datos en cualquier momento, no se debe reescribir ni una línea de código, siendo necesario solamente modificar un parámetro en un archivo de configuración.

3.2.2 Patrones de diseño

Sin dudas los patrones de diseño son una herramienta muy potente para el desarrollo de *software*. Su objetivo fundamental es que sean reutilizados en el contexto donde puedan presentarse, puesto que cada patrón tiene un objetivo específico a resolver. La reutilización permite: reducción de tiempo, disminución del esfuerzo de mantenimiento, eficiencia, consistencia, fiabilidad y protección de la inversión en desarrollo [\[32\]](#). A continuación se hará referencia a las principales definiciones de los patrones, así como se analizan los patrones de diseño que puedan servir de ayuda durante el diseño de la arquitectura y posteriormente en la implementación de la solución propuesta.

3.2.2.1 Patrones GRASP²⁸

Creador: en las clases controladoras *actions* se encuentran las acciones definidas y se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que las clases controladoras *actions* son "creadoras" de dichas entidades.

Experto: este es uno de los más utilizados, puesto que Propel es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, encapsula toda la

²⁶ Siglas de *Object-Relational mapping*, mapeo de objeto-relacionales, una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

²⁷ Siglas de PHP *Data Objects*, objetos de datos de PHP, es una extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.

²⁸ Siglas de *General Responsibility Assignment Software Patterns*, patrones generales de *software* para asignación de responsabilidades.

lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

3.2.2.2 Patrones GOF²⁹

Instancia única: Es usado en Symfony para permitir el acceso desde cualquier lugar de la aplicación a los objetos relacionados con el núcleo del *framework*. Esto ocurre a través de la llamada a la función `sfContext::getInstance()` desde cualquier parte del código. Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Decorador: Añade funcionalidad a una clase, dinámicamente. El *layout*, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. La plantilla sólo se encarga de visualizar las variables definidas en el controlador, esto se puede ver como que el *layout* decora la plantilla.

3.2.3 Diagrama de clases del diseño

En el modelo de diseño se confeccionan los diagramas de clases del diseño. Los elementos básicos que se pueden encontrar en este diagrama son las clases y las relaciones que existen entre las mismas. El lenguaje utilizado para especificar las clases de este diagrama tiene que ser un lenguaje de programación. Consecuentemente, las operaciones, parámetros, atributos, tipos y demás elementos son especificados utilizando las sintaxis del lenguaje de programación elegido [\[33\]](#).

A continuación se muestran algunos de los diagramas de clases del diseño que componen el modelo de diseño del sistema (para consultar los restantes diagramas de clases del diseño ver [Anexo C](#)):

²⁹ Siglas de *Gan of Four*, banda de cuatro, en honor a sus cuatro autores. Patrones de diseño orientado a objetos.

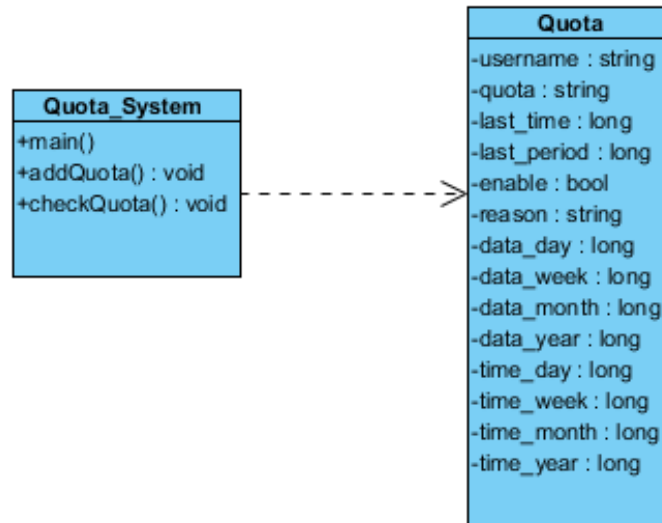


Figura 8: Diagrama de Clases del Diseño: CU Cambiare el estado de la cuota.

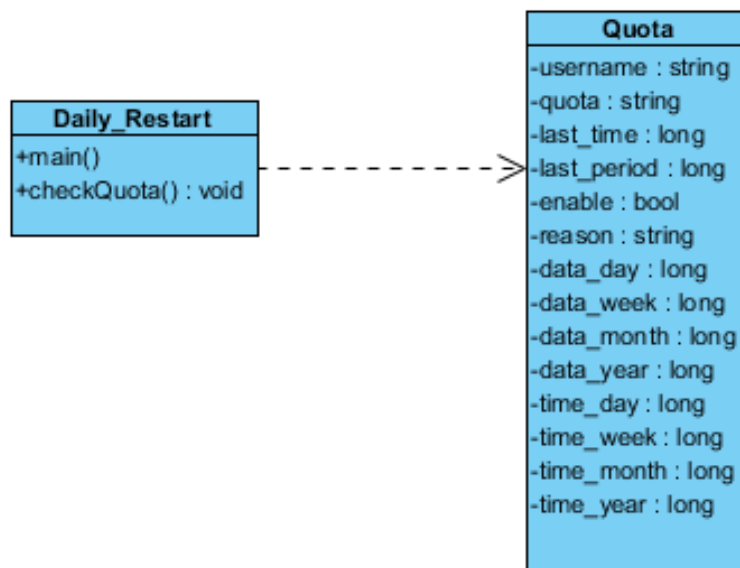


Figura 9: Diagrama de Clases del Diseño: CU Reiniciar cuota diaria.

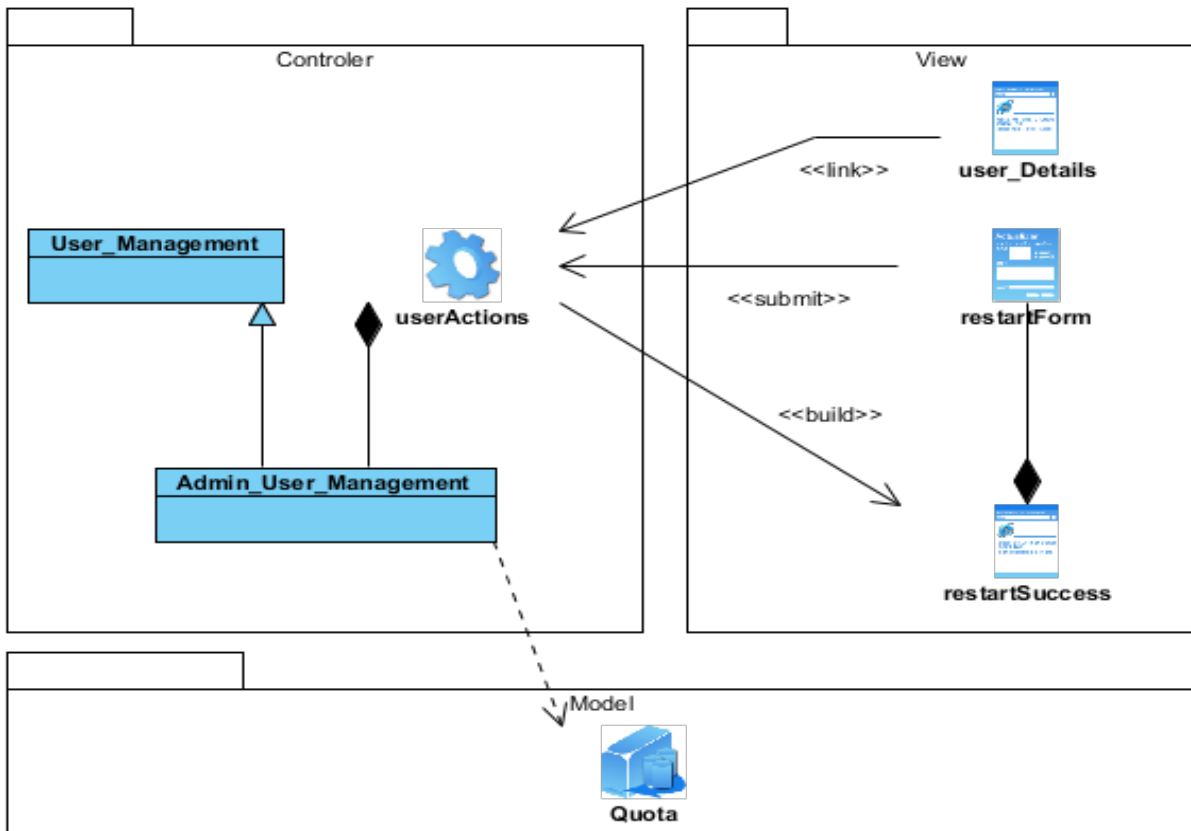


Figura 10: Diagrama de Clases del Diseño: CU Reiniciar cuota a un usuario según parámetro.

3.2.4 Diagramas de secuencia

Los diagramas de secuencia y los diagramas de colaboración (ambos llamados de interacción) constituyen diagramas UML que se utilizan para la modelación de los aspectos dinámicos de un sistema. Un diagrama de este tipo muestra una interacción que consiste en un conjunto de objetos y sus relaciones, teniendo como valor añadido el hecho de mostrar los mensajes que se envían entre dichos objetos. Los diagramas de secuencia destacan el orden cronológico de los mensajes mientras que los diagramas de colaboración destacan la organización estructural de los objetos que envían y reciben mensajes.

Específicamente a los diagramas de secuencia los distinguen una línea de vida que pertenece a cada objeto y que representa su existencia a lo largo de un período de tiempo. La mayoría de estos objetos perdurarán en la medida que dure la interacción.

A continuación se proponen algunos de los diagramas de secuencia generados (para

consultar los restantes diagramas de secuencia ver [Anexo D](#)):

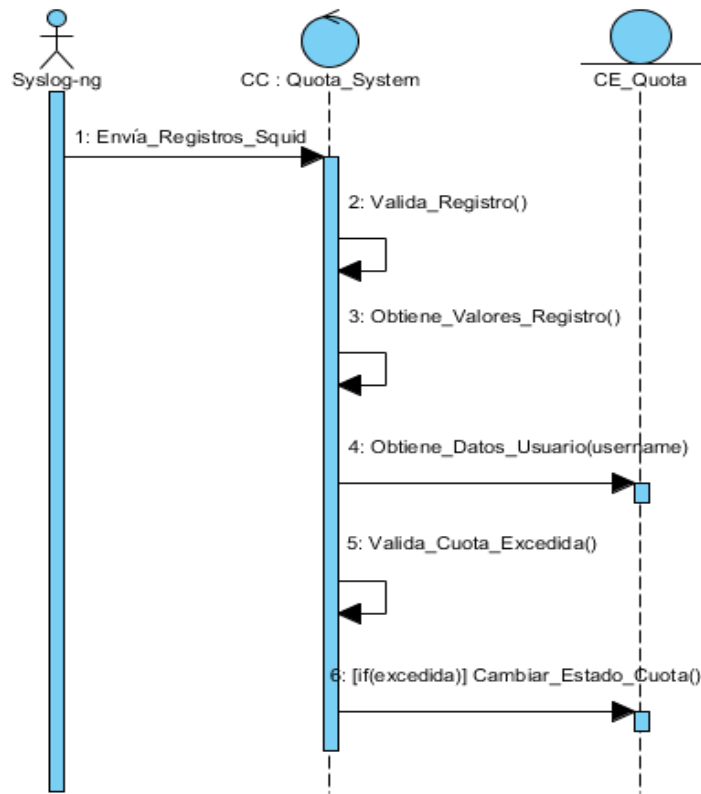


Figura 11: Diagrama de Secuencia: CU Cambiar el estado de la cuota.

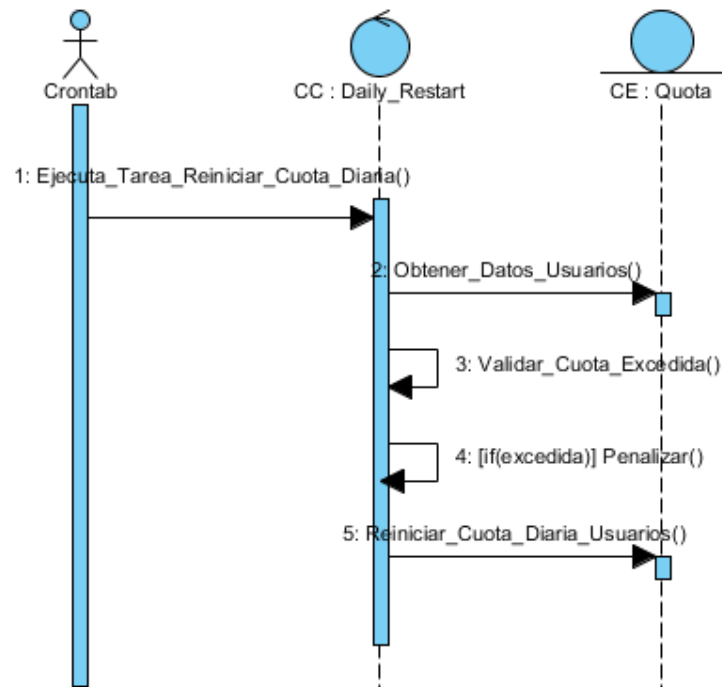


Figura 12: Diagrama de Secuencia: CU Reiniciar cuota diaria.

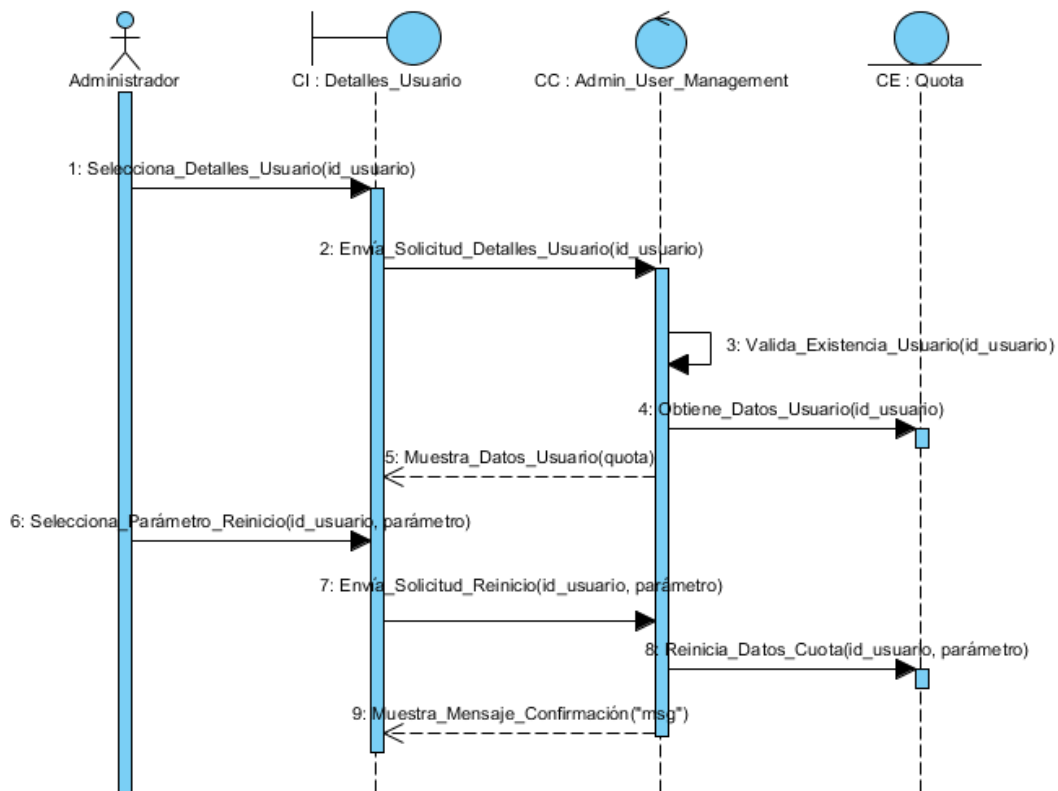


Figura 13: Diagrama de Secuencia: CU Reiniciar cuota a un usuario según parámetro.

3.2.5 Diseño de la base de datos

La persistencia de los datos, para su posterior procesamiento, es un elemento crítico en la implementación del sistema que compete a esta investigación. Como se apuntó anteriormente el sistema estará sustentado por el almacenamiento de los datos en MongoDB. Aunque las entidades en este tipo de base de datos no guardan relación, es necesario realizar una modelación del modo en que se conservarán los datos. MongoDB tiene su propio “lenguaje” a la hora de nombrar ciertos términos que constituyen paradigmas de las bases de datos que funcionan bajo la influencia del SQL. En MongoDB, un documento es la unidad básica de datos, comparable a una tupla en bases de datos SQL, pero mucho más expresiva. Una colección de documentos puede ser vista como el equivalente a una tabla de los sistemas relacionales.

Teniendo en cuenta que las colecciones en MongoDB no guardan relación entre sí, al menos no de forma intencional, se optó por el almacenamiento de los datos del sistema de cuotas en una única colección que será detallada a continuación, almacenada a su vez en la base de datos “fcweb”.

3.2.5.1 Colección Quota

En esta colección se almacenarán los datos del consumo de cuota de los usuarios, de los cuales se servirá el sistema de cuota para su funcionamiento. La información que se conservará en esta colección estará constituida por el identificador del usuario, la cadena de definición de la cuota, un valor numérico para el último instante de conexión y otro para el último período de navegación ininterrumpida, un campo que almacena si la cuota ha sido excedida, una cadena que conserva el período en que fue excedida la cuota en caso de que el usuario supere la cuota asignada, así como los valores de datos descargados y tiempo en línea de los usuarios, diferenciados por día, semana, mes y año.

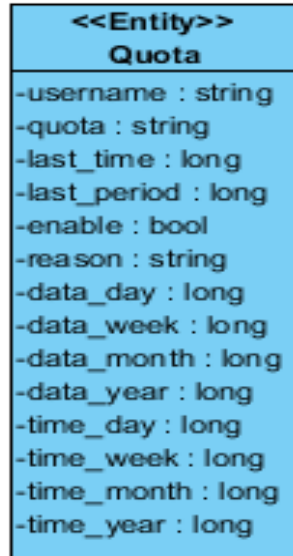


Figura 14: Colección Quota.

3.2.6 Diagrama de despliegue

El diagrama de Despliegue muestra las relaciones físicas entre los componentes *hardware* y *software* en el sistema final. Constituye en sí un grafo con nodos unidos por conexiones de comunicación donde cada uno de estos nodos puede contener instancias de componentes. De modo general, un nodo representa un recurso de cómputo, normalmente un procesador o recurso, un dispositivo de *hardware* similar, mientras que las instancias de componentes de *software* pueden estar unidas por relaciones de dependencia [\[34\]](#).

En la siguiente figura queda ilustrada la forma en que estará distribuida la aplicación a implementar:

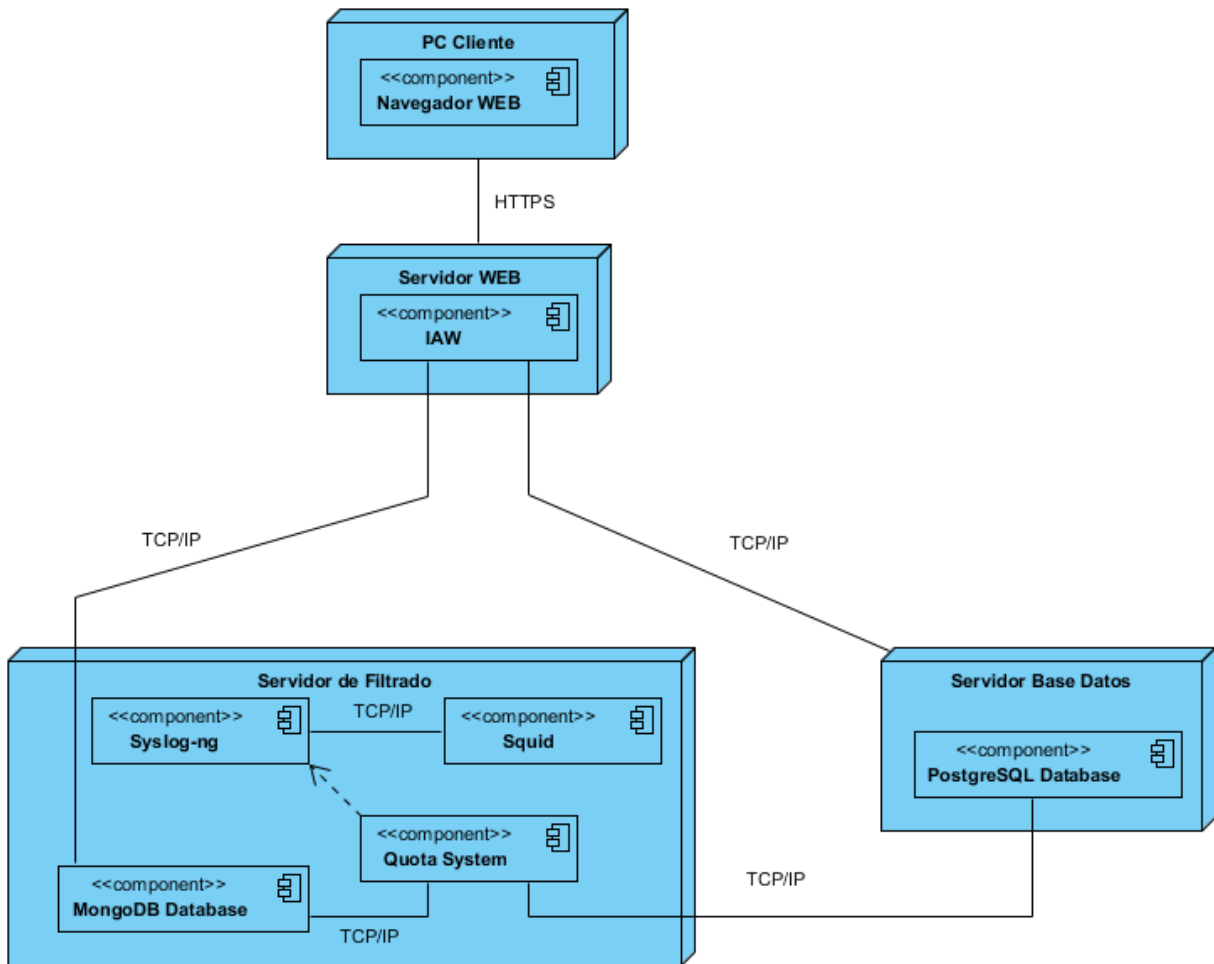


Figura 15: Diagrama de despliegue.

3.3 Conclusiones

A través del paso por los flujos de trabajo de Análisis y Diseño se pudo obtener una visión más clara de las características del sistema a implementar. Tras efectuar el análisis del sistema de cuotas se comprendieron los requisitos de manera más precisa, además, se logró preparar y simplificar las tareas del diseño. El diseño posibilitó definir la estructura del sistema y conseguir un sistema flexible a transformaciones en los requisitos. Mediante los diagramas de secuencia se reveló la forma en que los objetos interactúan entre sí al transcurrir el tiempo, contribuyendo de manera importante a entender el comportamiento del sistema.

Hasta este punto se ha logrado obtener un alto nivel de abstracción de diseño que deberá traducirse para obtener un alto nivel de abstracción operacional, se asignó de forma

eficiente las responsabilidades a los componentes del sistema. Se está en condiciones de afirmar que están creadas las bases para proceder a una implementación eficaz del sistema.

CAPÍTULO IV: IMPLEMENTACIÓN Y PRUEBAS

El desarrollo de los flujos de implementación y prueba constituye el punto de mayor relevancia durante la fase de construcción de RUP. Como resultado de estos flujos los modelos generados en las fases anteriores se traducen en componentes que forman parte de la aplicación final, ilustrando la relación entre los mismos. La realización de pruebas también reviste gran importancia, teniendo en cuenta que se puede detectar la existencia de errores en la implementación realizada. En este capítulo, se describe cómo los elementos del modelo de diseño son implementados en términos de componentes. Se exponen los resultados de las distintas pruebas realizadas al sistema de cuotas que da solución al problema planteado.

4.1 Implementación

La implementación es el centro en las iteraciones de construcción en la metodología RUP. El propósito general de la implementación es desarrollar la arquitectura y el sistema como un todo. Como resultado de este flujo, los modelos generados en las fases anteriores se traducen en los componentes que formarán parte de la aplicación final, reflejando la forma en que estos se relacionan entre sí para cumplir con los requisitos de *software* determinados con anterioridad.

4.1.1 Diagrama de componentes

Los diagramas de componentes representan todos los tipos de elementos de *software* que entran en la confección de aplicaciones, dígame ficheros de datos o código fuente y binarios o ejecutables así como las dependencias entre ellos. Los elementos de modelado que lo conforman son los componentes y paquetes y muestran la estructura del sistema en términos de implementación a un alto nivel [35]. El diagrama de componentes forma parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, proporcionando la oportunidad de establecer correspondencias entre las clases y los componentes de la implementación.

La figura muestra el diagrama de componentes del sistema propuesto:

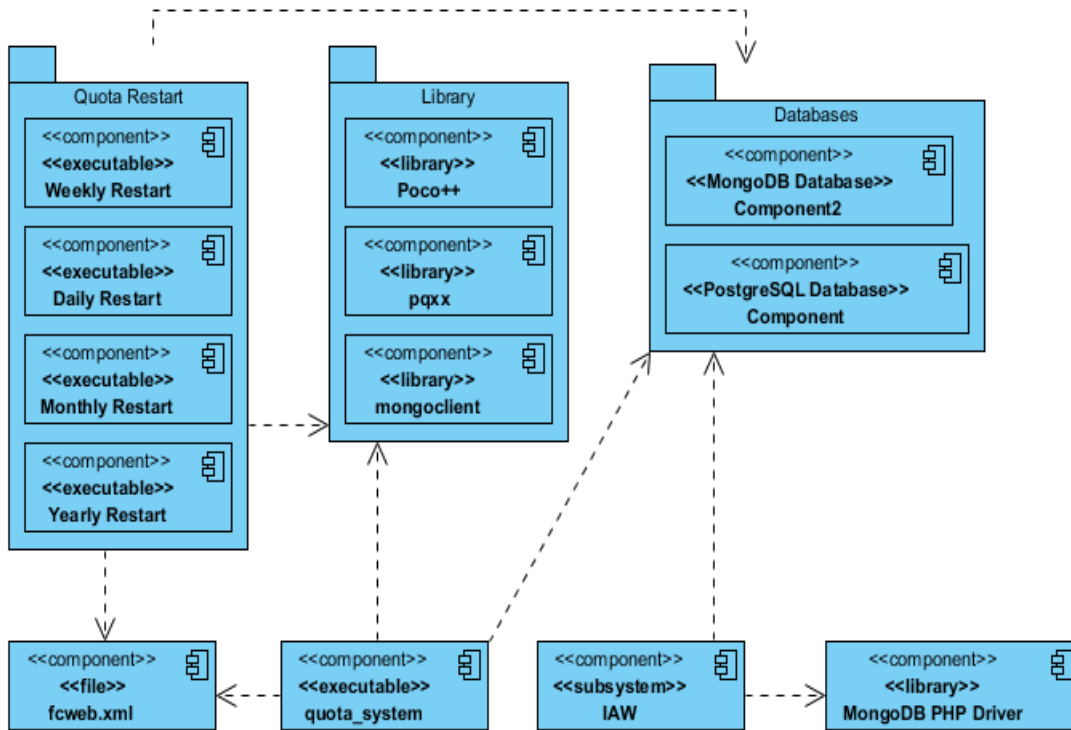


Figura 16: Diagrama de componentes.

4.1.2 Pantallas principales del sistema de cuotas

El sistema de cuotas desarrollado a pesar de que se ejecuta en un segundo plano, en interacción con diversos sistemas y no con usuarios a través de una interfaz, brinda la posibilidad de consultar las cuotas y realizar diversas operaciones para el reinicio del consumo los usuarios.

Las siguientes imágenes muestran las principales interfaces que permiten la interacción con el sistema de cuotas.

Detalles del Usuario "ediazv"

Datos básicos								
Nombre	Eduardo							
Apellido(s)	Diaz Vázquez							
Correo	ediazv@estudiantes.uci.cu							
Navegación								
Cuota	35mb/day 200mb/month 8h/day							
Cuota usada	Datos				Tiempo en línea			
	Día	Semana	Mes	Año	Día	Semana	Mes	Año
	36.56Mb	105.43Mb	752.43Mb	1.21Gb	42s	4:31m	19:24m	53:10m
	Reiniciar	Reiniciar	Reiniciar	Reiniciar	Reiniciar	Reiniciar	Reiniciar	Reiniciar



Figura 17: Vista de administración de la cuota consumida por un usuario y funciones de reinicio.

Usuarios

<input type="checkbox"/>	Usuario	Activo	Navegación	Política	Grupo de navegación	Acciones			
<input type="checkbox"/>	ediazv	✓	✓	default		Detalles	Reiniciar cuota	Editar	Borrar
<input type="checkbox"/>	administrador	✓	✓	default		Detalles	Reiniciar cuota	Editar	Borrar
<input type="checkbox"/>	acabreu	✓	✓	default		Detalles	Reiniciar cuota	Editar	Borrar
<input type="checkbox"/>	eypupo	✓	✓	default		Detalles	Reiniciar cuota	Editar	Borrar
<input type="checkbox"/>	mrodriguez	✓	✓	default		Detalles	Reiniciar cuota	Editar	Borrar
<input type="checkbox"/>	jclobaina	✓	✓	default		Detalles	Reiniciar cuota	Editar	Borrar

6 resultados Resultados por página 10

Selecciona una acción Adicionar

Figura 18: Funcionalidad Reiniciar cuota desde la vista de administración.

4.2 Pruebas

En el flujo de trabajo de prueba, según la metodología RUP, se verifica el resultado de la implementación, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregado a terceros. Las pruebas son un elemento

crítico para la garantizar la calidad del *software* y representan la revisión final de las especificaciones del diseño y de la codificación.

4.2.1 Tipos de prueba realizadas

Pruebas de Integración

Las pruebas de integración son técnicas sistemáticas para comprobar la arquitectura del *software*. Estas pruebas se realizan debido a que aunque las partes de un sistema funcionen bien por separado es necesario probarlos conjuntamente, ya que una parte puede tener un efecto adverso o inadvertido sobre otra. Por tanto, el objetivo de las pruebas de integración es probar el *software* como un todo, luego de probar sus partes por separado [\[36\]](#).

Pruebas Funcionales

Se denominan pruebas funcionales a las pruebas de *software* que tienen por objetivo probar que los sistemas desarrollados cumplan con los requisitos funcionales del *software*. A este tipo de pruebas se les denomina también pruebas de comportamiento y para realizarlas se emplea el método de caja negra, donde los probadores o analistas de pruebas no dirigen su atención en el cómo se generan las respuestas del sistema, sino en el funcionamiento de la interfaz del sistema. Básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas. El principal objetivo de este tipo de pruebas es demostrar que las funciones del *software* son operativas, que la entrada se produce de forma adecuada y la salida de forma correcta y que además se mantiene la integridad de la información [\[36\]](#).

Pruebas de Aceptación

Las pruebas de aceptación se realizan sobre el sistema completo, no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcional pactada previamente con el cliente. El objetivo de estas pruebas es comprobar si el cliente está satisfecho con el producto desarrollado y si este producto cumple con sus expectativas, permitiéndole al usuario que pueda determinar la aceptación del sistema.

Estas pruebas deben ser responsabilidad del cliente y son escritas en colaboración con el cliente y el equipo de desarrollo para que no exista la falta de entendimiento [\[37\]](#).

4.2.2 Resultados de las pruebas realizadas

La aplicación de las pruebas de integración confirmó que el sistema de cuotas se acopla a Smart Keeper de forma correcta, obteniendo los datos desde las diferentes interfaces de la manera esperada. Para la realización de las pruebas funcionales se definieron los casos de pruebas por cada Caso de Uso del sistema, así como los resultados esperados para cada uno de los posibles escenarios a probar. Para consultar los casos de pruebas ver [Anexo E](#).

Como resultado de la aplicación de las pruebas funcionales, según los casos de prueba previamente definidos, fueron detectados un total de cinco no conformidades, relacionadas con la internacionalización y validaciones, así como el tratamiento de excepciones. Con la realización de los cambios pertinentes se corrigió el flujo normal de eventos del sistema hacia el estado correcto.

4.3 Conclusiones

Durante el desarrollo del presente capítulo, estrechamente relacionado con los flujos de implementación y prueba, el sistema quedó desarrollado y validado. El diagrama de componentes ilustró la relación entre los principales componentes del sistema, integrado a Smart Keeper. Las pruebas realizadas confirmaron que el sistema cumple con los requerimientos de *software* definidos, después de haber sido corregidos varias no conformidades, de esta forma se garantiza la calidad del sistema de cuotas.

CONCLUSIONES GENERALES

Luego de realizada la investigación, enfocada en el desarrollo de un sistema de cuotas para la gestión de la navegación por tiempo y tráfico para Smart Keeper, se logró dar solución al problema planteado partiendo del marco investigativo elaborado. Tras culminar la investigación y el desarrollo generado a partir de la misma, es posible arribar a las siguientes conclusiones:

- ✓ Después de analizar los inconvenientes de diversos trabajos similares existentes en el ámbito nacional e internacional, se evidencia la necesidad de concebir una herramienta propia, que extienda las características actuales de los mecanismos para el aprovechamiento del ancho de banda en Smart Keeper.
- ✓ Una vez identificados los requisitos funcionales y no funcionales y elaborados los artefactos de especificación de los Casos de Usos del *software*, se pudo diseñar el sistema de cuotas, permitiendo establecer sus características y todo lo necesario para comenzar con su construcción.
- ✓ Tras efectuar el análisis y diseño, se obtuvo una entrada apropiada y un punto de partida para comenzar la etapa de implementación.
- ✓ Con la implementación de las funcionalidades del sistema de cuotas, se obtuvo una herramienta que fortalece y estandariza con normas internacionales los mecanismos de control del ancho de banda existentes en Smart Keeper.
- ✓ Con la ejecución de las pruebas de *software*, se comprobó que los requerimientos de *software* definidos, fueron satisfechos.

RECOMENDACIONES

A pesar de que las funcionalidades del sistema de cuotas fueron diseñadas e implementadas para integrarse al *software* de filtrado Smart Keeper el autor recomienda:

- ✓ Rediseñar el sistema de cuotas de forma tal que pueda ser utilizado en cualquier institución que utilice el *proxy* Squid para brindar el servicio de navegación, lo que permitiría el uso del mismo sin necesidad de instalar Smart Keeper.

GLOSARIO DE TÉRMINOS

CSS: Siglas de *Cascading Style Sheets*, Hojas de Estilo en Cascada. Tecnología empleada en la creación de páginas web, que permite un mayor control sobre el lenguaje HTML. Permite hojas de estilo que definen como cada elemento se tiene que mostrar.

DHTML: *Dynamic HTML*. Conjunto de extensiones a HTML que permiten modificar el contenido de una página web en el cliente sin necesidad de establecer una conexión con el servidor.

FTP: Siglas de *File Transfer Protocol*, Protocolo de Transferencia de Archivos, en informática, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red, basado en la arquitectura Cliente-Servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

Hardware: Corresponde a todas las partes físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado. Es el substrato físico en el cual existe el *software*, maquinaria real.

HTML: Siglas de *HyperText Markup Language*, Lenguaje de Marcado de Hipertexto. Lenguaje compuesto de una serie de etiquetas o marcas que permiten definir el contenido y la apariencia de las páginas Web. Aunque se basa en el estándar SGML, no se puede considerar que sea un subconjunto de él. Existen cientos de etiquetas con diferentes atributos.

HTTP: Siglas de *Hypertext Transfer Protocol*, Protocolo de Transferencia de Hipertexto, es el protocolo que emplea la WWW. Define como se tienen que crear y enviar los mensajes y que opciones debe tener el servidor y el navegador en respuesta a un comando.

HTTPS: Siglas de *Hypertext Transfer Protocol Secure*, Protocolo Seguro de Transferencia de Hipertexto, es la versión segura del protocolo HTTP y viene dada por el uso de

protocolos de seguridad, que crean un canal cifrado más apropiado para el tráfico de información sensible.

Puerta de enlace: Una puerta de enlace o pasarela (del inglés *gateway*) es un dispositivo, con frecuencia una computadora, que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red, al protocolo usado en la red de destino. La puerta de enlace es normalmente un equipo informático configurado para dotar a las máquinas de una red de área local conectadas a él de un acceso hacia una red exterior.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

SSL: Siglas de *Secure Sockets Layer*, capa de conexión segura, es un protocolo comúnmente utilizado para la gestión de la seguridad de una transmisión de mensajes en Internet, está incluido como parte de los navegadores web más utilizados y la mayoría de los productos de servicios web.

TLS: Siglas de *Transport Layer Security*, capa de transporte segura, es un protocolo que asegura la privacidad entre aplicaciones de comunicación y sus usuarios en Internet. Cuando un servidor y el cliente se comunican, TLS garantiza que ningún tercero puede interceptar o manipular cualquier mensaje. TLS es el sucesor del protocolo SSL.

Traffic shapping: El *traffic shaping* o recorte de tráfico es una técnica utilizada para controlar el tráfico en redes de ordenadores para así lograr optimizar o garantizar el rendimiento, baja latencia y un ancho de banda determinado, retrasando paquetes que cumplan con cierto criterio.

Tupla: Las tuplas encuentran cabida en el estudio teórico de las bases de datos sobre todo en el campo del cálculo relacional ya que proporcionan una notación básica para formular la definición de la relación en términos de las relaciones de la base de datos. En el cálculo relacional se emplea el cálculo orientado a tuplas, frente al orientado a dominio. Se emplea muy a menudo en la definición a gran nivel de las definiciones de los pares atributo-valor.

REFERENCIAS BIBLIOGRÁFICAS

- [1] **Carlos Vega Lebrún, Diego Arvizu Gutiérrez, Arturo García Santillán.** *Algoritmos para encriptación de Datos. Elementos técnicos de estudio*, p. 9. Edición electrónica, 2008.
- [2] **Cisco: Products & Services** [Internet]. Disponible en: <http://www.cisco.com/en/US/products/index.html> [Citado 21 Noviembre 2011].
- [3] **Allot NetEnforcer Bandwidth Management Device** [Internet]. Disponible en: http://www.allot.com/Bandwidth_Management_Devices.html [Citado 21 Noviembre 2011].
- [4] **Antamedia Bandwidth Manager** [Internet]. Disponible en: <http://www.antamediabandwidth.com/> [Citado 23 Noviembre 2011].
- [5] **Bandwidth Manager for Windows - Bandwidth Controller** [Internet]. Disponible en: <http://bandwidthcontroller.com/index.html> [Citado 23 Noviembre 2011].
- [6] **Squid: Optimising Web Delivery** [Internet]. Disponible en: <http://www.squid-cache.org/> [Citado 23 Noviembre 2011].
- [7] **Squid configuration directive delay_class** [Internet]. Disponible en: http://www.squid-cache.org/Doc/config/delay_class/ [Citado 25 Noviembre 2011].
- [8] **Carlos A. Martínez.** *Squid y los Delay Pools* [Internet]. Disponible en: <http://bulma.net/body.phtml?nIdNoticia=2284> [Citado 25 Noviembre 2011].
- [9] **Leading Edge Business Solutions** [Internet]. Disponible en: <http://www.ledge.co.za/software/squint/squish/> [Citado 25 Noviembre 2011].
- [10] **Sergio Luján Mora.** *Programación de aplicaciones Web: historia, principios básicos y clientes Web*, pp. 25–37. Editorial Club Universitario, 2002.
- [11] **Average Web Page Size Septuples Since 2003** [Internet]. Disponible en: <http://www.websiteoptimization.com/speed/tweak/average-web-page/> [Citado 25 Noviembre 2011].

- [12] CUOTAS-Enciclopedia de Economía [Internet]. Disponible en: <<http://www.economia48.com/spa/d/cuotas/cuotas.htm>> [Citado 26 Noviembre 2011].
- [13] Pontificia Universidad Católica del Ecuador - *Glosario de Términos* [Internet]. Disponible en: <<http://dc193.4shared.com/doc/CfRbTml/preview.html>> [Citado 26 Noviembre 2011].
- [14] **Carlos Raúl García, Fabricio Paul Guerra Galván, Maximiliano Vanzetti.** *Diseño e Implementación de Sistemas Operativos*, p. 9. Universidad Tecnológica Nacional. Facultad Regional Santa Fe, 2008.
- [15] *RUP vs XP* [Internet]. Disponible en: <<http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUPvs.XP.pdf>> [Citado 27 Noviembre 2011].
- [16] **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software*. ISBN: 84-7829-036-2, pp. 1-12, Addison Wesley, 2000.
- [17] **Agustín Dondo.** *¿Por qué elegir PHP?* [Internet]. Disponible en: <<http://www.programacion.com/php/articulo/porquephp/>> [Citado 27 Noviembre 2011].
- [18] *NetBeans IDE - Base IDE Features.* [Internet]. Disponible en: <<http://netbeans.org/features/ide/index.html>> [Citado 27 Noviembre 2011].
- [19] **Erich Gamma et al (1994).** *Design Patterns: Elements of Reusable Object-Oriented Software* [Internet]. Disponible en: <<http://www.exciton.cs.rice.edu/JavaResources/DesignPatterns/book/hires/chap1fso.htm>> [Citado 27 Noviembre 2011].
- [20] *The Definitive Guide to symphony. Chapter 1 - Introducing Symphony.* [Internet]. Disponible en: <http://www.symfony-project.org/book/1_0/01-Introducing-Symfony> [Citado 27 Noviembre 2011].
- [21] *What VP-UML Provides.* [Internet]. Disponible en: <<http://www.visual-paradigm.com/product/vpuml/provides/>> [Citado 27 Noviembre 2011].
- [22] *NoSQL* [Internet]. Disponible en: <<http://www.strozzi.it/cgi->

[bin/CSA/tw7//en_US/nosql/Home %20Page](#)> [Citado 27 Noviembre 2011].

[23] **Cattell Rick**. *Relational Databases, Object Databases, Key-Values Stores, Document Stores and Extensible Record Store: A Comparison* [Internet]. Disponible en: <<http://www.odbms.org/download/Cattell.Dec10.pdf>> [Citado 27 Noviembre 2011].

[24] *Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase Comparison* [Internet]. Disponible en: <<http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis>> [Citado 27 Noviembre 2011].

[25] *Memcached* [Internet] Disponible en: <<http://memcached.org>> [Citado 27 Noviembre 2011].

[26] **Kristina Chodorow, Michael Dirolf**. *MongoDB. The Definitive Guide*, ISBN: 978-1-449-38156-1, pp. 1-9. O'REILLY, 2010.

[27] **Kristina Chodorow, Michael Dirolf**. *MongoDB. The Definitive Guide*. ISBN: 978-1-449-38156-1, p. 5. O'REILLY, 2010.

[28] *BSON* [Internet] Disponible en: <<http://bsonspec.org>> [Citado 20 Enero 2012].

[29] **Ivar Jacobson, Grady Booch, James Rumbaugh**. *El Proceso Unificado de Desarrollo de Software*. ISBN: 84-7829-036-2, pp. 174-177, Addison Wesley, 2000.

[30] **Roger S. Pressman**. *Ingeniería del Software. Un enfoque práctico*. Quinta Edición. ISBN: 84-4813-214-9, pp. 238-239, McGraw-Hill, 2002.

[31] **Roger S. Pressman**. *Ingeniería del Software. Un enfoque práctico*. Quinta Edición. ISBN: 84-4813-214-9, pp. 241-243, McGraw-Hill, 2002.

[32] **Roger S. Pressman**. *Ingeniería del Software. Un enfoque práctico*. Quinta Edición. ISBN: 84-4813-214-9, pp. 390-393, McGraw-Hill, 2002.

[33] **Ivar Jacobson, Grady Booch, James Rumbaugh**. *El Proceso Unificado de Desarrollo de Software*. ISBN: 84-7829-036-2, pp. 208-209, Addison Wesley, 2000.

[34] **Ivar Jacobson, Grady Booch, James Rumbaugh**. *El Proceso Unificado de*

Desarrollo de Software. ISBN: 84-7829-036-2, pp. 217-218, Addison Wesley, 2000.

[35] **Ivar Jacobson, Grady Booch, James Rumbaugh**. *El Proceso Unificado de Desarrollo de Software*. ISBN: 84-7829-036-2, pp. 257-258, Addison Wesley, 2000.

[36] **Roger S. Pressman**. *Ingeniería del Software. Un enfoque práctico*. Quinta Edición. ISBN: 84-4813-214-9, pp. 294-318, McGraw-Hill, 2002.

[37] *La prueba de aceptación*. [Internet] Disponible en:
<http://pruebasdesoftware.com/pruebadeaceptacion.htm> [Citado 20 marzo 2012.].

ANEXOS

ANEXO A: FIGURAS RELACIONADAS

Bandwidth Manager

Please choose interface and start Bandwidth Manager in Setup - Network page.

ACCOUNTS Autologin Server Control

Computer

IP Address | MAC

IP Address	MAC
192.168.0.1	15-D5-F1-5D-6F-48

IP: 192.168.0.2
MAC: F8-35-A1-35-D4-D6

Download: 15 KB/sec
Upload: 15 KB/sec

Quota: 100.00 MB
 Enable unlimited quota

Time: 00:30
 Enable unlimited access time

Only allow access
From 00:00 To 00:00

Inactivity Timeout
 Active

Create Account Delete Account

Computers connected: 0
Bandwidth KB/sec
Download: 0.00
Upload: 0.00

Figura 19: Interfaz de administración de la herramienta Antamedia Bandwidth Manager.

Proxy usage

Select a user name to view details

User	Data				Time on-line			
	24 hours	Week	Month	Year	24 hours	Week	Month	Year
andrewm	75.20mb	81.76mb	89.59mb	254.69mb	2:55h	6:09h	9:44h	3d
michelle	25.15mb	106.17mb	248.03mb	684.26mb	3:53h	15:07h	2d	6d
10.0.0.7	2.90mb	8.37mb	29.44mb	1.331Gb	1:24h	3:12h	12:24h	6d
10.0.0.51	71.66kb	4.40mb	29.79mb	191.61mb	1:15m	3:21h	15:19h	3d
Total	103.31mb	200.70mb	396.86mb	2.435Gb	8:13h	1d	3d	19d

Interpretation:

- Data: the total amount of data transferred, excluding overheads
- Time on-line: time spent active on the internet, including any idle interval less than 5 minutes

[All users](#)

Figura 21: Interfaz de la herramienta Squish.

ANEXO B: DIAGRAMAS DE CLASES DEL ANÁLISIS

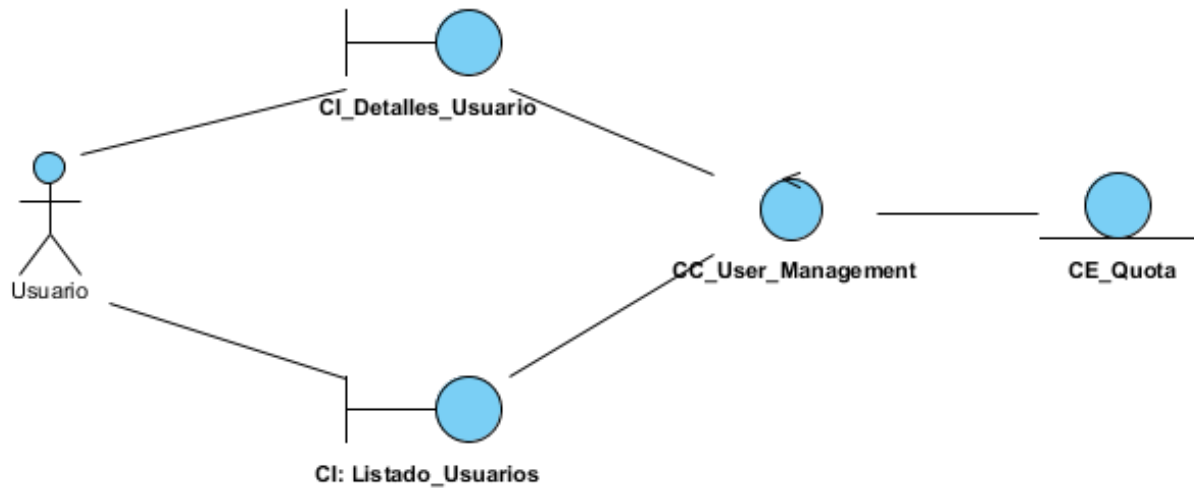


Figura 22: Diagrama de Clases del Análisis: CU Consultar la cuota consumida.

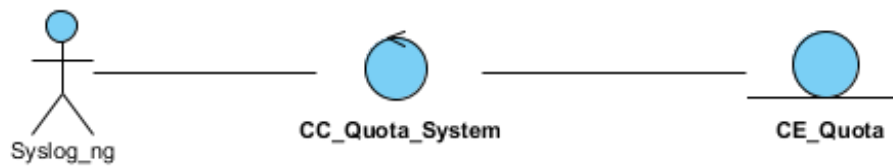


Figura 23: Diagrama de Clases del Análisis: CU Adicionar valores de consumo

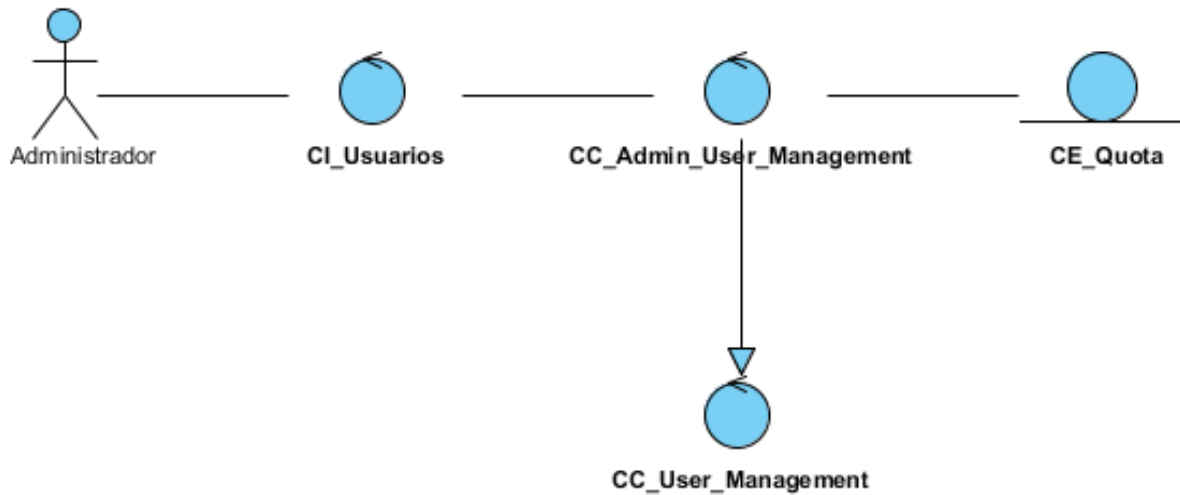


Figura 24: Diagrama de Clases del Análisis: CU Reiniciar cuota total a usuario.



Figura 25: Diagrama de Clases del Análisis: CU Reiniciar cuota semanal.



Figura 26: Diagrama de Clases del Análisis: CU Reiniciar cuota mensual.



Figura 27: Diagrama de Clases del Análisis: CU Reiniciar cuota anual.

ANEXO C: DIAGRAMAS DE CLASES DEL DISEÑO

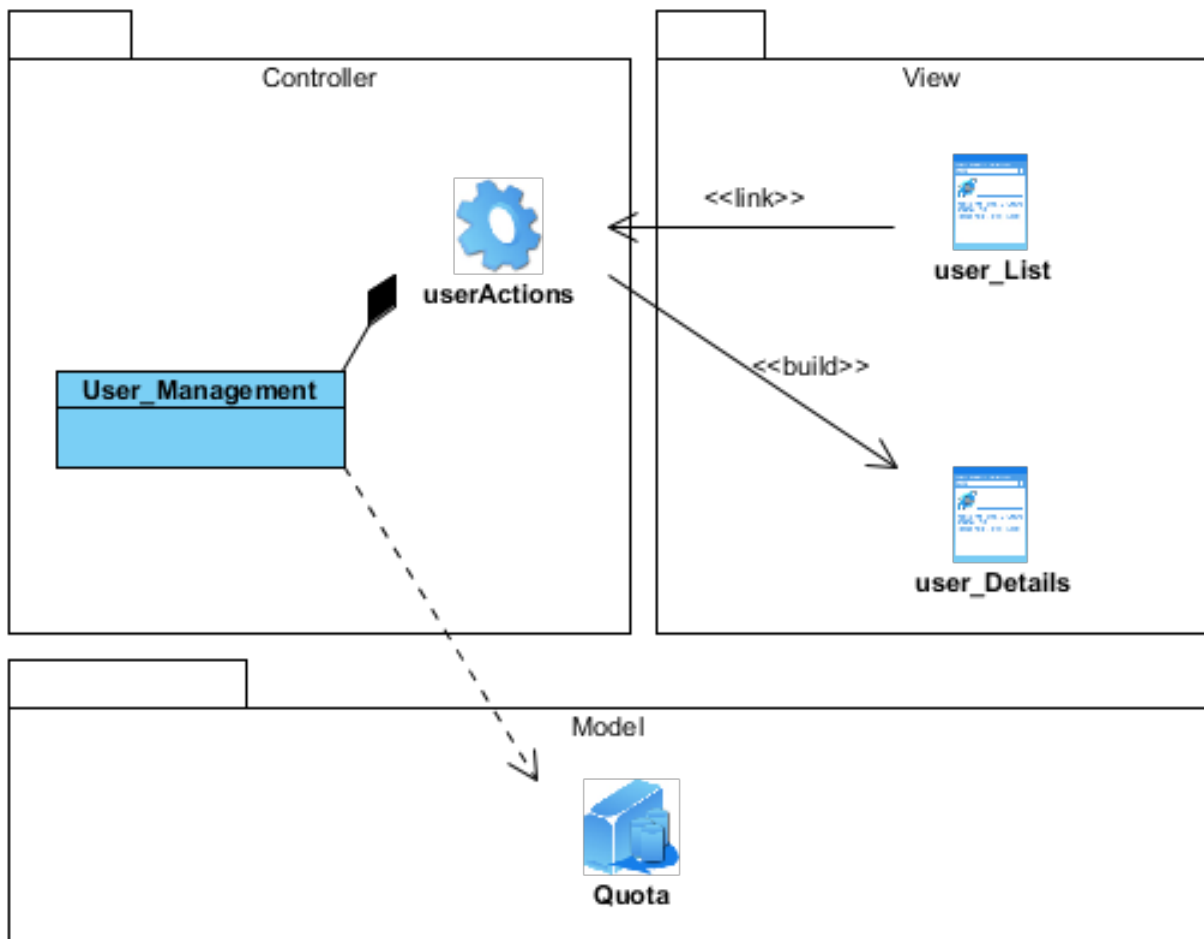


Figura 28: Diagrama de Clases del Diseño: CU Consultar la cuota consumida.

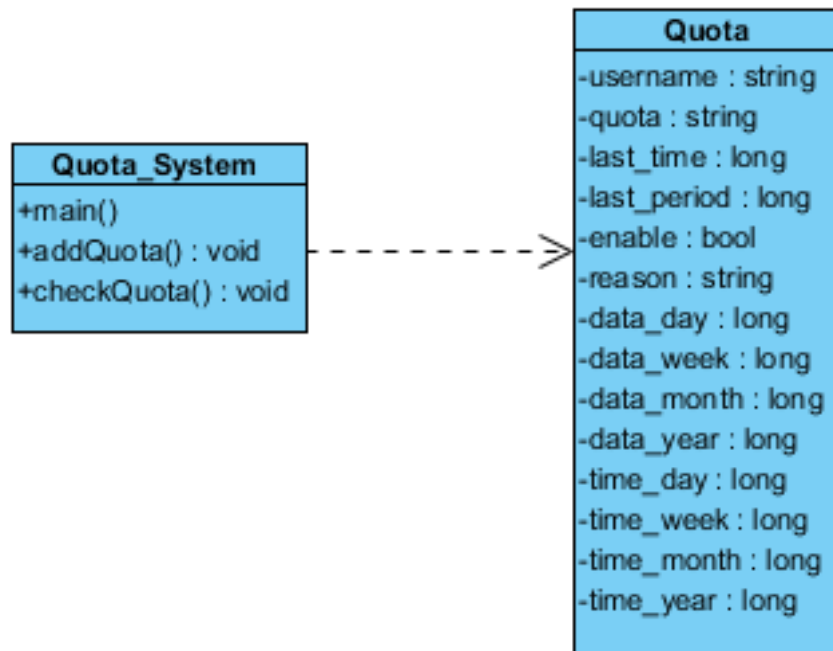


Figura 29: Diagrama de Clases del Diseño: CU Adicionar valores de consumo.

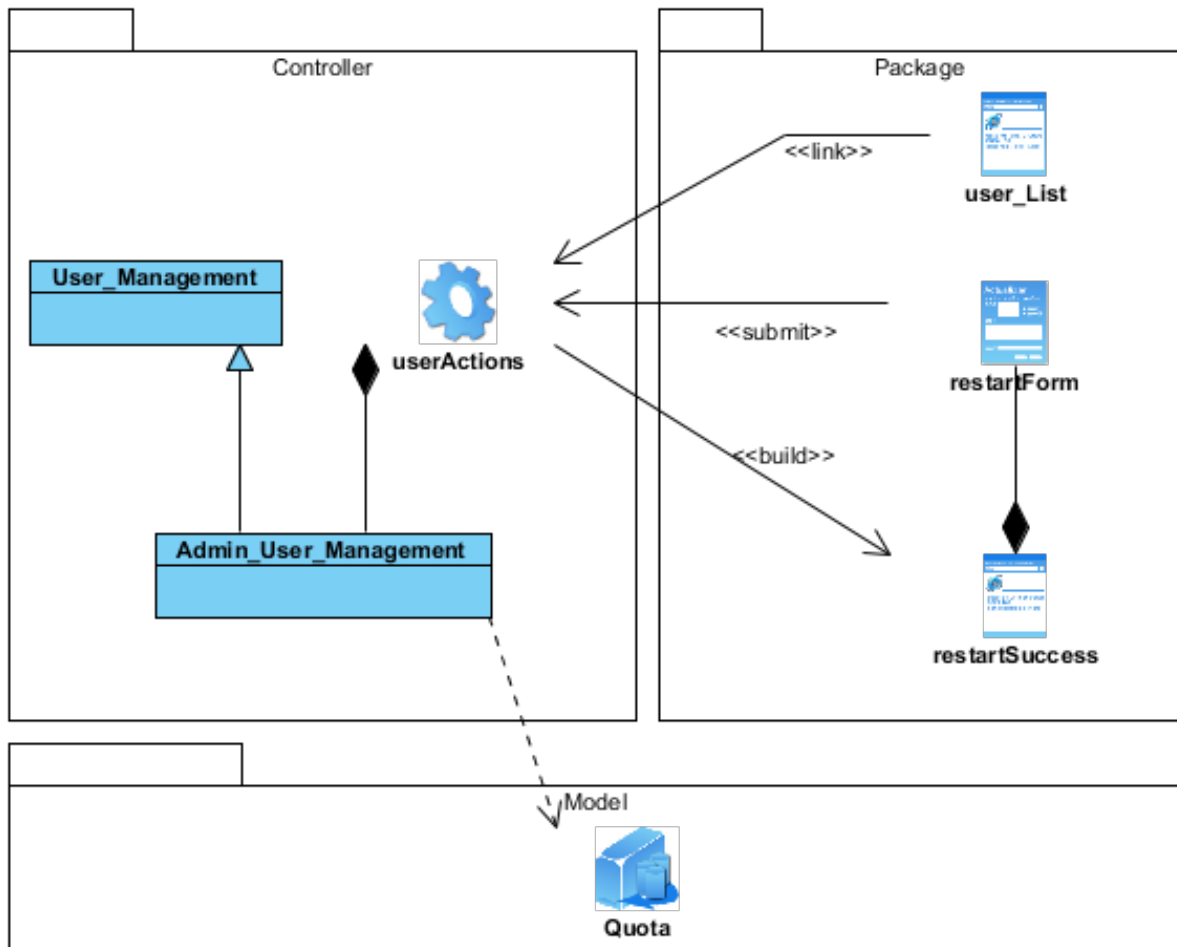


Figura 30: Diagrama de Clases del Diseño: CU Reiniciar cuota total a usuario.

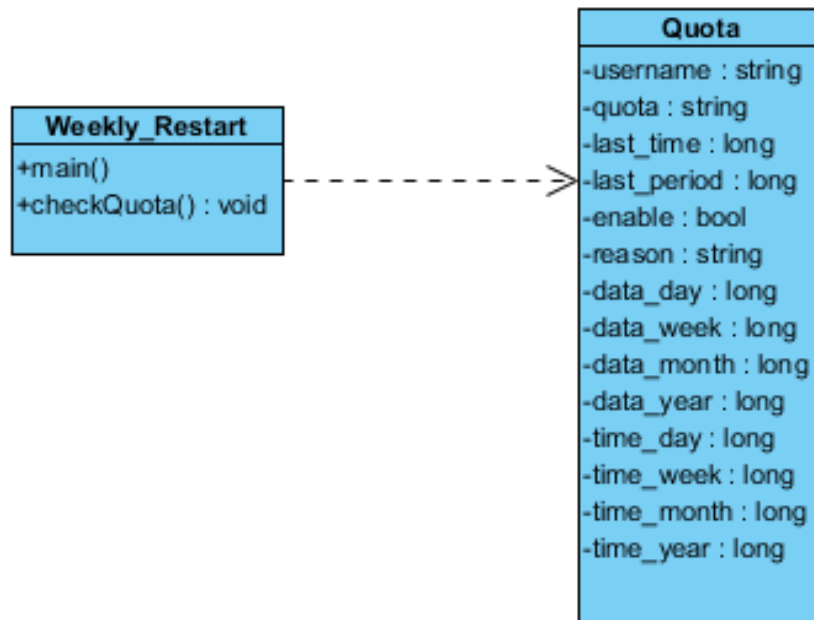


Figura 31: Diagrama de Clases del Diseño: CU Reiniciar cuota semanal.

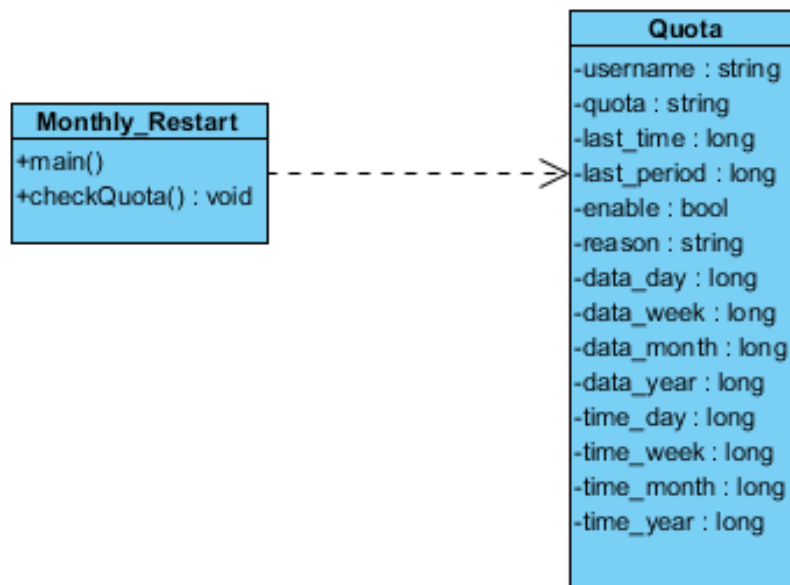


Figura 32: Diagrama de Clases del Diseño: CU Reiniciar cuota mensual.

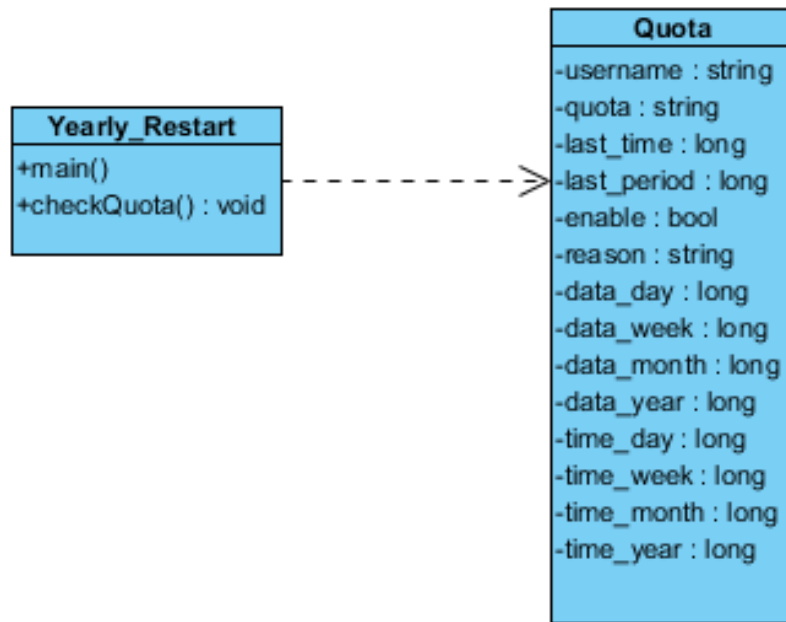


Figura 33: Diagrama de Clases del Diseño: CU Reiniciar cuota anual.

ANEXO D: DIAGRAMAS DE SECUENCIA

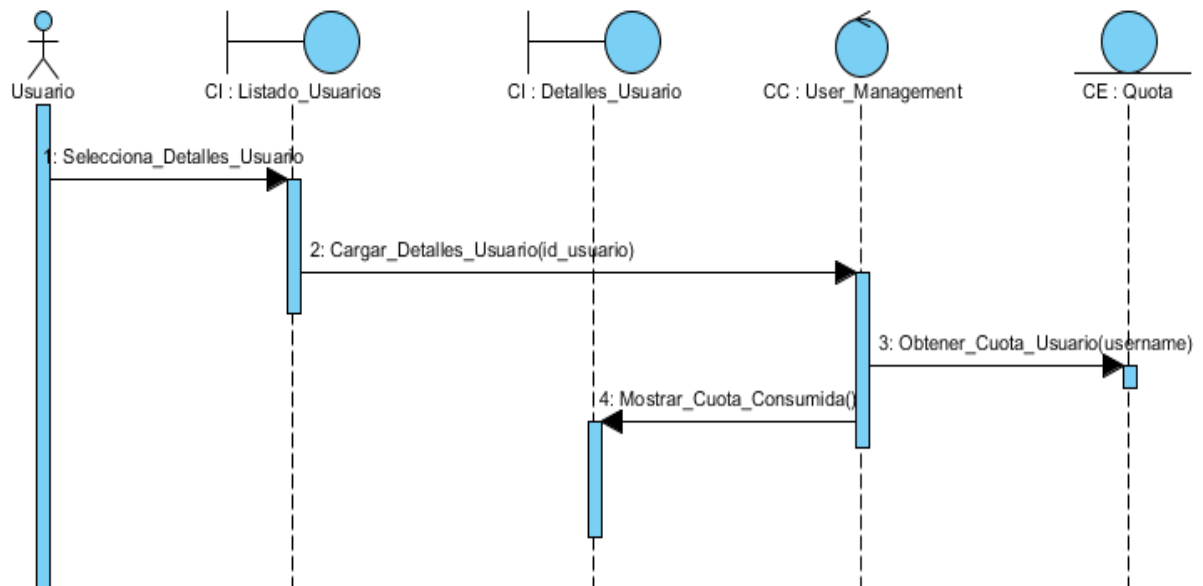


Figura 34: Diagrama de Secuencia: CU Consultar la cuota consumida.

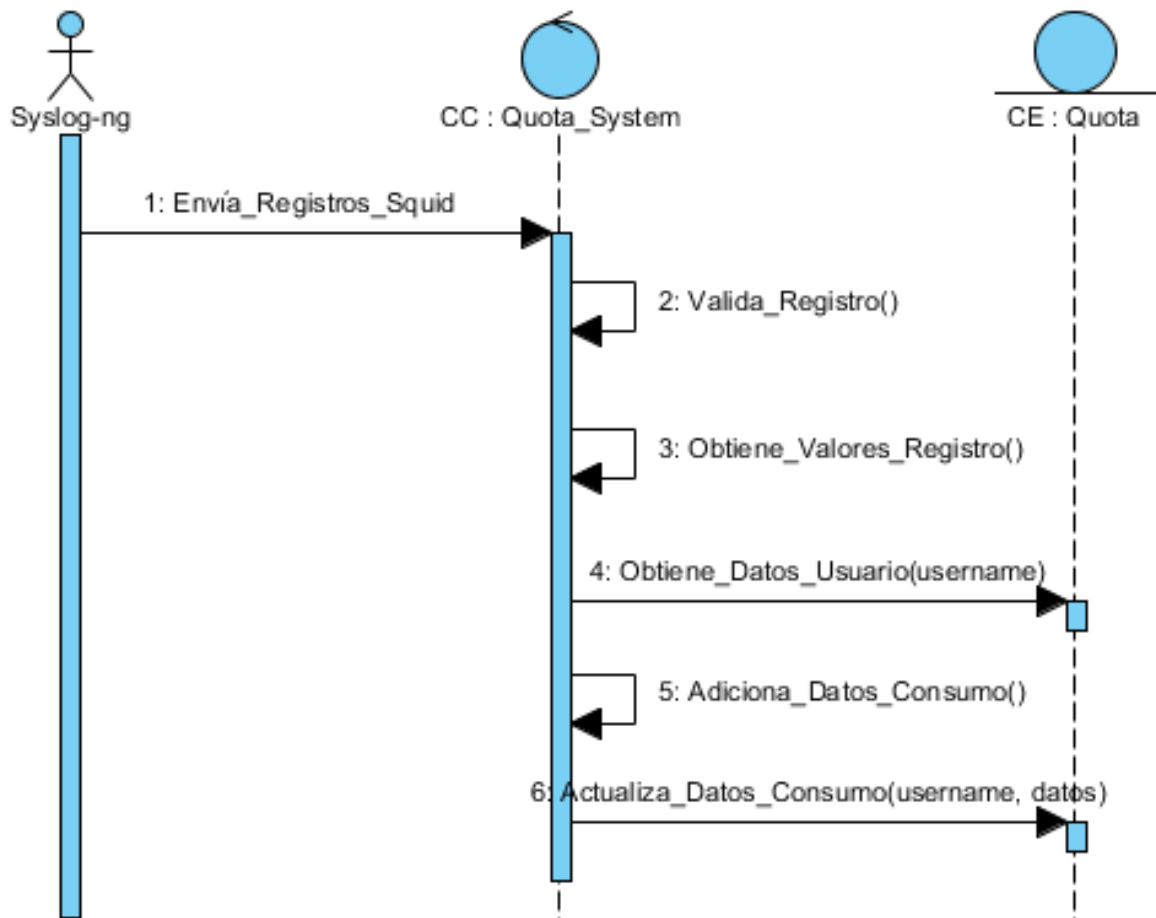


Figura 35: Diagrama de Secuencia: CU Adicionar valores de consumo.

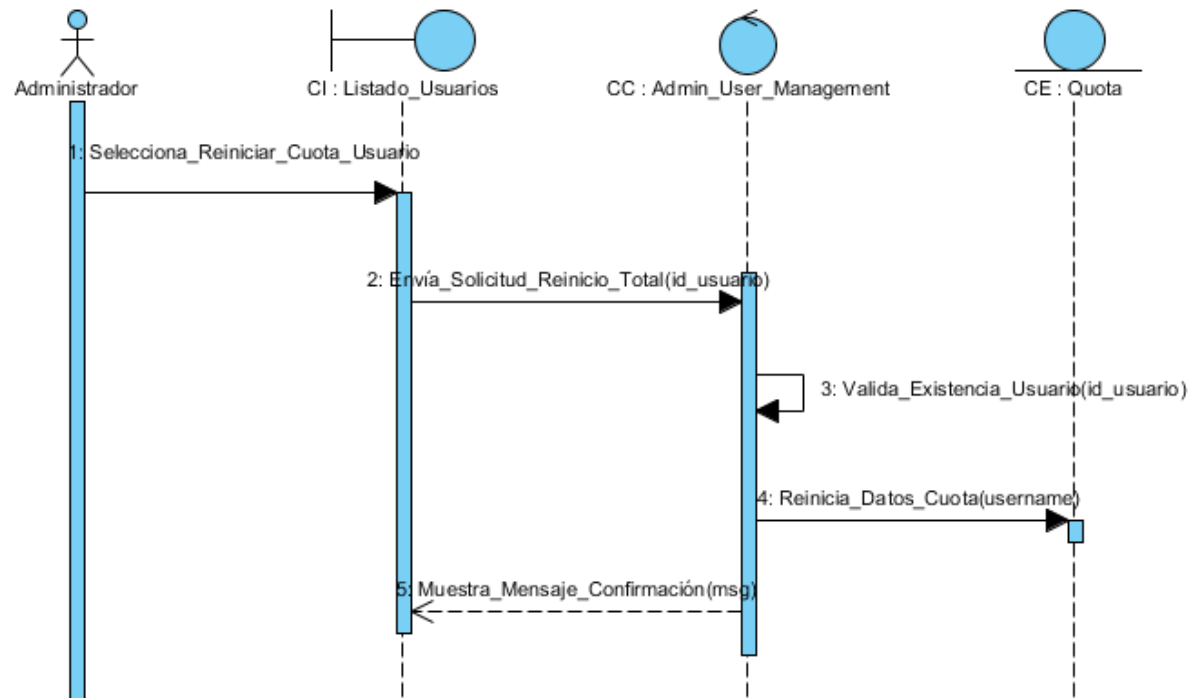


Figura 36: Diagrama de Secuencia: CU Reiniciar cuota total a usuario.

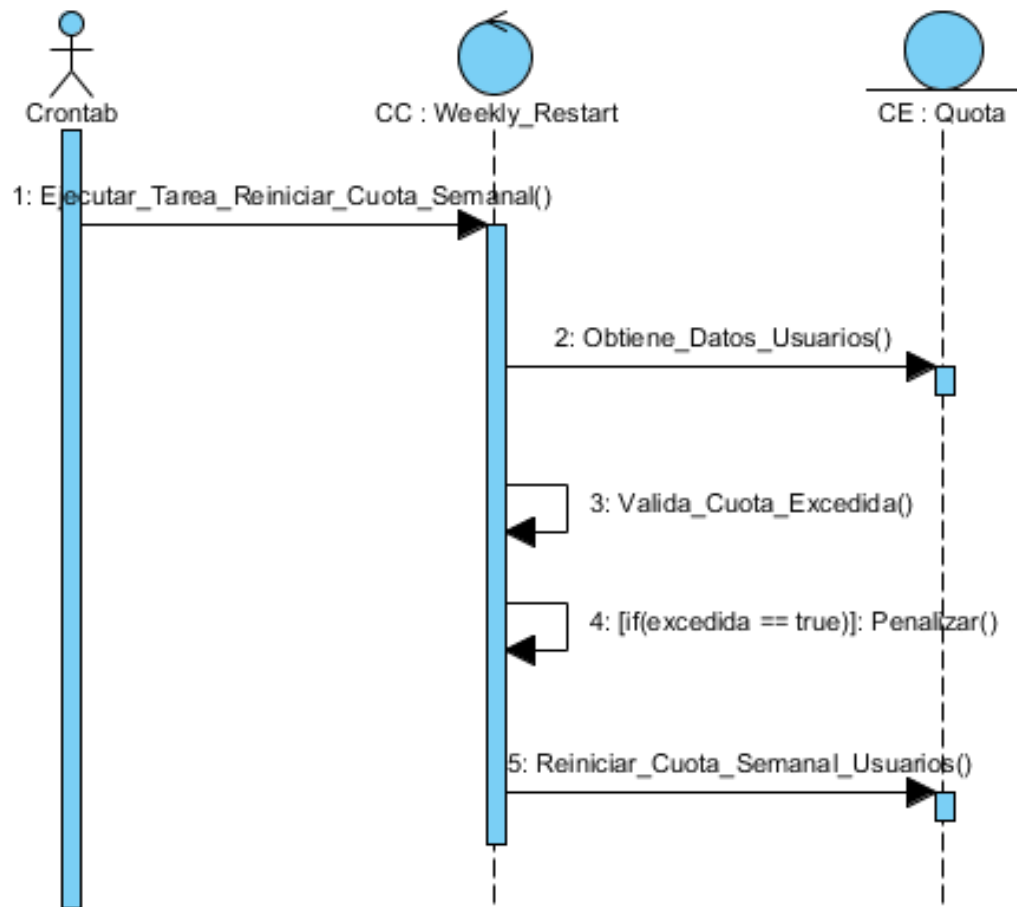


Figura 37: Diagrama de Secuencia: CU Reiniciar cuota semanal.

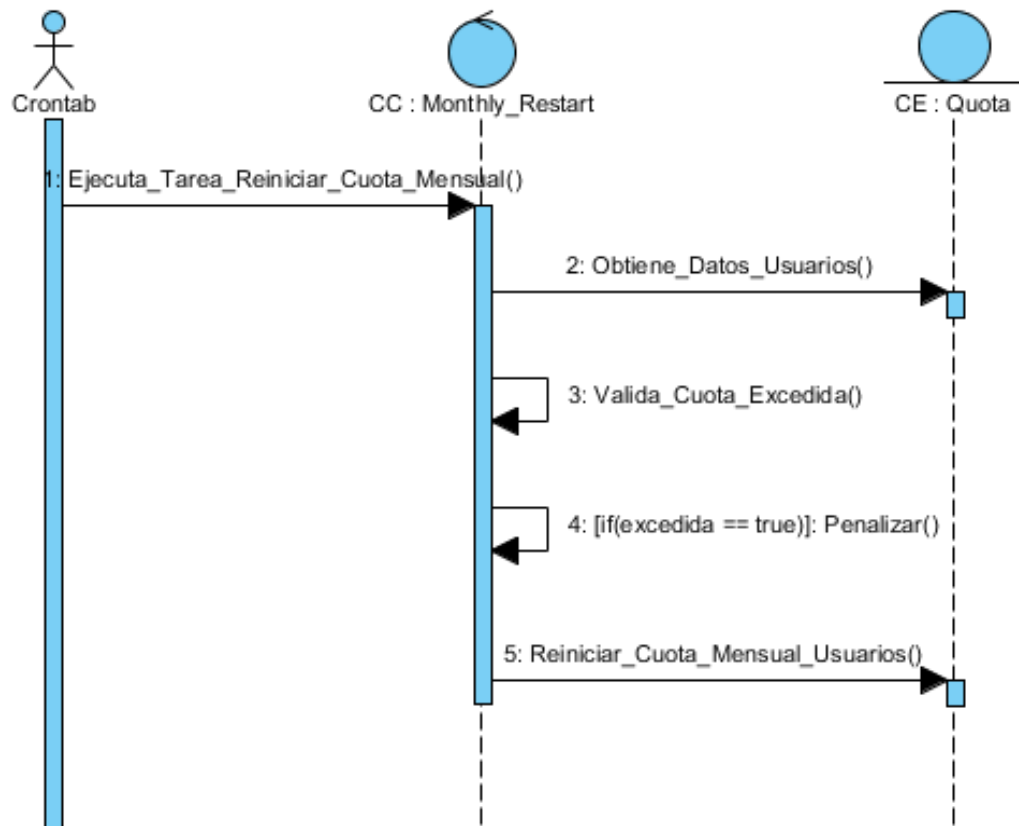


Figura 38: Diagrama de Secuencia: CU Reiniciar cuota mensual.

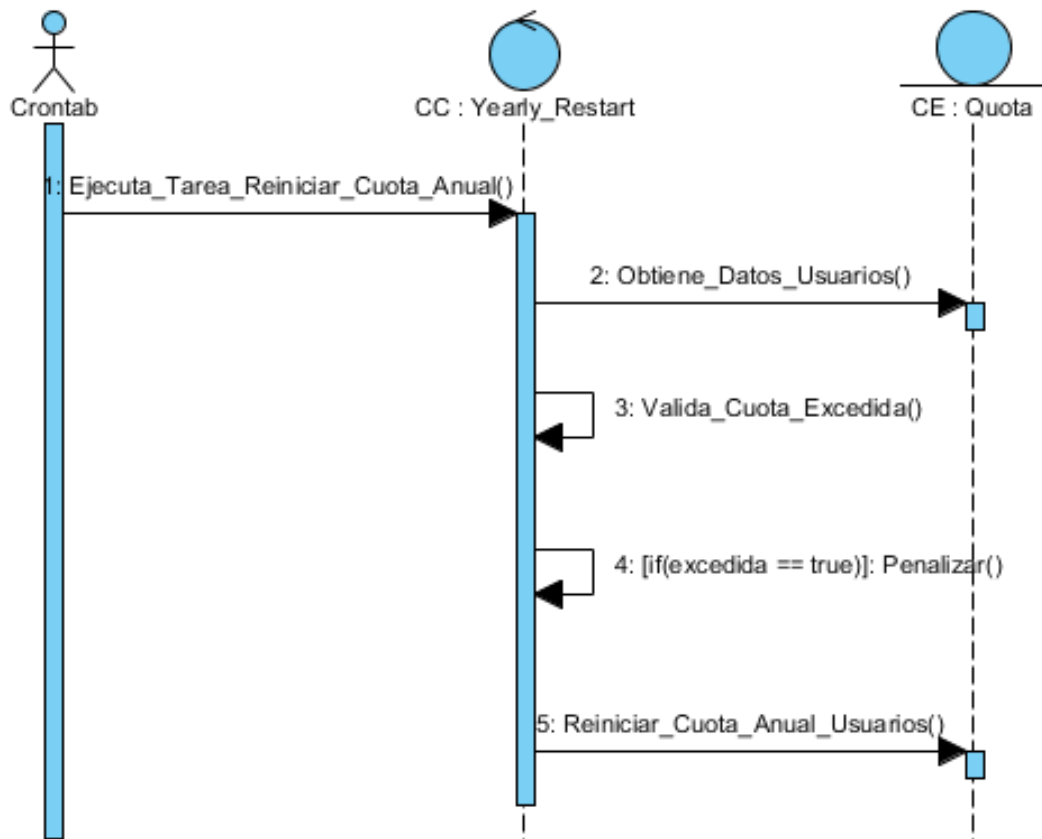


Figura 39: Diagrama de Secuencia: CU Reiniciar cuota anual.

ANEXO E: DISEÑOS DE CASOS DE PRUEBA BASADOS EN CASOS DE USO

Caso de Uso Consultar la cuota consumida

Escenario	Descripción	Respuesta del sistema	Flujo central
Mostrar cuota consumida usuario Administrador.	Se muestran los detalles de la definición de cuota asignada al usuario y el consumo por día, semana, mes y año.	Una interfaz muestra la tabla con los datos de la definición de cuota asignada al usuario además del consumo por día, semana, mes y año.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado.
Mostrar cuota consumida usuario sin privilegios.	Se muestran los detalles de la definición de cuota asignada al usuario y el consumo por día, semana, mes y año.	Una interfaz muestra la tabla con los datos de la definición de cuota asignada al usuario además del consumo por día, semana, mes y año.	<ul style="list-style-type: none"> • Al usuario una vez autenticado se le muestra la vista "Detalles". • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año.
Problemas de disponibilidad de MongoDB al consultar la cuota consumida.	Se intenta consultar los detalles de la cuota consumida y la base de datos MongoDB no se encuentra disponible.	Se muestra un mensaje imposibilidad de mostrar los detalles de la cuota por fallo de MongoDB.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" un mensaje de error.

Tabla 21: Caso de Prueba: CU Consultar la cuota consumida.

Caso de Uso Adicionar valores de consumo

Escenario	Descripción	Respuesta del sistema	Flujo central
Adicionar los valores de la cuota correctamente	Se analiza un registro enviado al sistema por Syslog-ng por la entrada estándar y se adicionan los valores de la cuota consumida al usuario correspondiente.	Se acumulada el consumo de ancho de banda del usuario.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <code>cd /usr/local/fcweb/bin/</code> <code>./cuota_constraint /usr/local/fcweb/etc/fcweb.xml</code> Una vez que el programa se ejecute adicionar entradas del archivo de registro ubicado en <code>/var/log/squid3/access.log</code> .
Problemas de disponibilidad de MongoDB	Se intenta adicionar los valores de la cuota consumida y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro <code>quota.log</code> .	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <code>cd /usr/local/fcweb/bin/</code> <code>./cuota_constraint /usr/local/fcweb/etc/fcweb.xml</code> Una vez que el programa se ejecute adicionar entradas del archivo de registro ubicado en <code>/var/log/squid3/access.log</code> .

Tabla 22: Casos de Prueba: CU Adicionar valores de consumo.

Caso de Uso Cambiar estado de la cuota

Escenario	Descripción	Respuesta del sistema	Flujo central
El usuario excede la cuota asignada.	Se analiza un registro enviado al sistema por Syslog-ng por la entrada estándar y se analiza si el usuario ha excedido el consumo de su cuota asignada.	Si el usuario ha excedido el consumo de su cuota se actualiza el campo <i>enable</i> a false y se almacena el periodo en el que fue excedida la cuota.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <code>cd /usr/local/fcweb/bin/ ./cuota_constraint /usr/local/fcweb/etc/fcweb.xml</code> Una vez que el programa se ejecute adicionar entradas del archivo de registro ubicado en /var/log/squid3/access.log.
El usuario no excede la cuota asignada.	Se analiza un registro enviado al sistema por Syslog-ng por la entrada estándar y se analiza si el usuario ha excedido el consumo de su cuota asignada.	El campo enable permanece inalterado en true.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <code>cd /usr/local/fcweb/bin/ ./cuota_constraint /usr/local/fcweb/etc/fcweb.xml</code> Una vez que el programa se ejecute adicionar entradas del archivo de registro ubicado en /var/log/squid3/access.log.
Problemas de disponibilidad de MongoDB.	Se intenta cambiar el estado de la cuota al usuario correspondiente y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <code>cd /usr/local/fcweb/bin/ ./cuota_constraint /usr/local/fcweb/etc/fcweb.xml</code> Una vez que el programa se ejecute adicionar entradas del archivo de registro ubicado en /var/log/squid3/access.log.

Tabla 23: Casos de Prueba: CU Cambiar estado de la cuota.

Caso de Uso Reiniciar cuota total a usuario

Escenario	Descripción	Respuesta del sistema	Flujo central
Se reinicia totalmente de forma correcta.	El actor selecciona la opción Reiniciar Cuota de un usuario y el sistema reinicia todos los valores de la cuota consumida por el usuario.	El sistema reinicia todos los valores de la cuota consumida por el usuario y muestra un mensaje de confirmación.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Reiniciar cuota" de un usuario listado. • El sistema muestra un mensaje de confirmación.
Problemas de disponibilidad de MongoDB al cambiar el estado de la cuota del usuario.	El actor selecciona la opción Reiniciar Cuota de un usuario y la base de datos MongoDB no se encuentra disponible.	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de MongoDB.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Reiniciar cuota" de un usuario listado. • El sistema muestra un mensaje de error.

Tabla 24: Casos de Prueba: CU Reiniciar cuota total a usuario.

Caso de Uso Reiniciar cuota diaria

Escenario	Descripción	Respuesta del sistema	Flujo central
Usuario excede la cuota asignada.	Se ejecuta una tarea programada para reiniciar la cuota consumida en el día por cada usuario. Se detecta que existen usuarios que exceden la cuota diaria.	A cada usuario que excedió su cuota diaria en más de 5 megabytes o 5 minutos, para los datos y el tiempo respectivamente, se le penaliza el reinicio de la cuota diaria con el valor del exceso. Ej. Cuota asignada 5 megabytes, cuota consumida 12 megabytes, reinicio 7 megabytes.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <pre>cd /usr/local/fcweb/bin/ ./daily /usr/local/fcweb/etc/fcweb.xml</pre>
Usuario no excede a cuota asignada.	Se ejecuta una tarea programada para reiniciar la cuota consumida en el día por cada usuario. No se detectan usuario que excedan la cuota diaria.	A cada usuario que no excedió su cuota diaria en más de 5 megabytes o 5 minutos, para los datos y el tiempo respectivamente, se le reinicia la diaria cuota con 0.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <pre>cd /usr/local/fcweb/bin/ ./daily /usr/local/fcweb/etc/fcweb.xml</pre>
Problemas de disponibilidad de MongoDB al consultar el consumo de cuota al usuario correspondiente.	Se intenta consultar el consumo de cuota al usuario correspondiente y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <pre>cd /usr/local/fcweb/bin/ ./daily /usr/local/fcweb/etc/fcweb.xml</pre>

Problemas de disponibilidad de MongoDB al reiniciar la cuota consumida del día al usuario correspondiente.	Se intenta reiniciar la cuota consumida del día al usuario correspondiente y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./daily /usr/local/fcweb/etc/fcweb.xml
--	---	---	--

Tabla 25: Casos de Prueba: Reiniciar cuota diaria.

Caso de Uso Reiniciar cuota semanal

Escenario	Descripción	Respuesta del sistema	Flujo central
Usuario excede la cuota asignada.	Se ejecuta una tarea programada para reiniciar la cuota consumida en la semana por cada usuario. Se detecta que existen usuarios que exceden la cuota semanal.	A cada usuario que excedió su cuota semanal en más de 5 megabytes o 5 minutos, para los datos y el tiempo respectivamente, se le penaliza el reinicio de la cuota semanal con el valor del exceso. Ej. Cuota asignada 5 megabytes, cuota consumida 12 megabytes, reinicio 7 megabytes.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./weekly /usr/local/fcweb/etc/fcweb.xml
Usuario no excede la cuota asignada.	Se ejecuta una tarea programada para reiniciar la cuota consumida en la	A cada usuario que no excedió su cuota semanal en más de 5 megabytes o 5 minutos, para los datos y el	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./weekly /usr/local/fcweb/etc/fcweb.xml

	semana por cada usuario. No se detectan usuario que excedan la cuota semanal.	tiempo respectivamente, se le reinicia la cuota semanal con 0.	
Problemas de disponibilidad de MongoDB al consultar el consumo de cuota al usuario correspondiente.	Se intenta consultar el consumo de cuota al usuario correspondiente y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./weekly /usr/local/fcweb/etc/fcweb.xml
Problemas de disponibilidad de MongoDB al reiniciar la cuota consumida de la semana al usuario correspondiente.	Se intenta reiniciar la cuota consumida de la semana al usuario correspondiente y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./weekly /usr/local/fcweb/etc/fcweb.xml

Tabla 26: Casos de Prueba: CU Reiniciar cuota semanal.

Caso de Uso Reiniciar cuota mensual

Escenario	Descripción	Respuesta del sistema	Flujo central
Usuario excede la cuota asignada.	Se ejecuta una tarea programada para reiniciar la cuota consumida en el mes	A cada usuario que excedió su cuota mensual en más de 5 megabytes o 5 minutos, para los datos y el tiempo	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./monthly /usr/local/fcweb/etc/fcweb.xml

	por cada usuario. Se detecta que existen usuarios que exceden la cuota mensual.	respectivamente, se le penaliza el reinicio de la cuota mensual con el valor del exceso. Ej. Cuota asignada 5 megabytes, cuota consumida 12 megabytes, reinicio 7 megabytes.	
Usuario no excede la cuota asignada.	Se ejecuta una tarea programada para reiniciar la cuota consumida en el mes por cada usuario. No se detectan usuario que excedan la cuota mensual.	A cada usuario que no excedió su cuota mensual en más de 5 megabytes o 5 minutos, para los datos y el tiempo respectivamente, se le reinicia la cuota mensual con 0.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./monthly /usr/local/fcweb/etc/fcweb.xml
Problemas de disponibilidad de MongoDB al consultar el consumo de cuota al usuario correspondiente.	Se intenta consultar el consumo de cuota al usuario correspondiente y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./monthly /usr/local/fcweb/etc/fcweb.xml
Problemas de disponibilidad de MongoDB al reiniciar la cuota consumida del mes al usuario correspondiente.	Se intenta reiniciar la cuota consumida del mes al usuario correspondiente y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./monthly /usr/local/fcweb/etc/fcweb.xml

Tabla 27: Casos de Prueba: CU Reiniciar cuota mensual.

Caso de Uso Reiniciar cuota anual

Escenario	Descripción	Respuesta del sistema	Flujo central
Usuario excede la cuota asignada.	Se ejecuta una tarea programada para reiniciar la cuota consumida en el año por cada usuario. Se detecta que existen usuario que exceden la cuota anual.	A cada usuario que excedió su cuota anual en más de 5 megabytes o 5 minutos, para los datos y el tiempo respectivamente, se le penaliza el reinicio de la cuota anual con el valor del exceso. Ej. Cuota asignada 5 megabytes, cuota consumida 12 megabytes, reinicio 7 megabytes.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <pre>cd /usr/local/fcweb/bin/ ./yearly /usr/local/fcweb/etc/fcweb.xml</pre>
Usuario no excede la cuota asignada.	Se ejecuta una tarea programada para reiniciar la cuota consumida en el año por cada usuario. No se detectan usuario que excedan la cuota anual.	A cada usuario que no excedió su cuota anual en más de 5 megabytes o 5 minutos, para los datos y el tiempo respectivamente, se le reinicia la cuota anual con 0.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <pre>cd /usr/local/fcweb/bin/ ./yearly /usr/local/fcweb/etc/fcweb.xml</pre>
Problemas de disponibilidad de MongoDB al consultar el	Se intenta consultar el consumo de cuota al usuario correspondiente y la	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: <pre>cd /usr/local/fcweb/bin/ ./yearly /usr/local/fcweb/etc/fcweb.xml</pre>

consumo de cuota al usuario correspondiente.	base de datos MongoDB no se encuentra disponible.		
Problemas de disponibilidad de MongoDB al reiniciar la cuota consumida del año al usuario correspondiente.	Se intenta reiniciar la cuota consumida del año al usuario correspondiente y la base de datos MongoDB no se encuentra disponible.	El sistema guarda un registro de error en el archivo de registro quota.log.	Ejecutar manualmente, como root del sistema, los comandos en el siguiente orden: cd /usr/local/fcweb/bin/ ./yearly /usr/local/fcweb/etc/fcweb.xml

Tabla 28: Casos de Prueba: CU Reiniciar cuota anual.

Caso de Uso Reiniciar cuota a un usuario según parámetro

Escenario	Descripción	Respuesta del sistema	Flujo central
Reiniciar datos consumidos del día	El actor selecciona la opción Reiniciar ubicada debajo del consumo de datos del día de un usuario y el sistema reinicia el valor del consumo de datos del día del usuario.	El sistema reinicia el valor del consumo de datos del día del usuario en 0.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de datos del día. • El sistema muestra un mensaje de confirmación.
Reiniciar tiempo	El actor selecciona la	El sistema reinicia el valor del	• El usuario una vez autenticado en el sistema selecciona el

consumido del día	opción Reiniciar ubicada debajo del consumo de tiempo del día de un usuario y el sistema reinicia el valor del consumo de tiempo del día del usuario.	consumo de tiempo del día del usuario en 0.	módulo "Filtrado básico". <ul style="list-style-type: none"> • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de tiempo del día. • El sistema muestra un mensaje de confirmación.
Reiniciar datos consumidos de la semana.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de datos de la semana de un usuario y el sistema reinicia el valor del consumo de datos de la semana del usuario.	El sistema reinicia el valor del consumo de datos de la semana del usuario en 0.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de datos de la semana. • El sistema muestra un mensaje de confirmación.
Reiniciar tiempo consumido de la semana.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de tiempo de la semana de un usuario y el sistema reinicia el valor del consumo de tiempo de la semana del usuario.	El sistema reinicia el valor del consumo de tiempo de la semana del usuario en 0.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de tiempo de la semana. • El sistema muestra un mensaje de confirmación.

Reiniciar datos consumidos del mes.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de datos del mes de un usuario y el sistema reinicia el valor del consumo de datos del mes del usuario.	El sistema reinicia el valor del consumo de datos del mes del usuario en 0.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de datos del mes. • El sistema muestra un mensaje de confirmación.
Reiniciar tiempo consumido del mes.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de tiempo del mes de un usuario y el sistema reinicia el valor del consumo de tiempo del mes del usuario.	El sistema reinicia el valor del consumo de tiempo del mes del usuario en 0.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de tiempo del mes. • El sistema muestra un mensaje de confirmación.
Reiniciar datos consumidos del año.	El actor selecciona la opción Reiniciar ubicada debajo del	El sistema reinicia el valor del consumo de datos del año del usuario en 0.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema.

	consumo de datos del año de un usuario y el sistema reinicia el valor del consumo de datos del año del usuario.		<ul style="list-style-type: none"> • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de datos del año. • El sistema muestra un mensaje de confirmación.
Reiniciar tiempo consumido del año	El actor selecciona la opción Reiniciar ubicada debajo del consumo de tiempo del año de un usuario y el sistema reinicia el valor del consumo de tiempo del año del usuario.	El sistema reinicia el valor del consumo de tiempo del año del usuario en 0.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de tiempo del año. • El sistema muestra un mensaje de confirmación.
Problemas de disponibilidad de MongoDB al reiniciar los datos consumidos del día.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de datos del día de un usuario y la base de datos MongoDB no se encuentra disponible.	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de MongoDB.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de datos del día. • El sistema muestra un mensaje de error.
Problemas de disponibilidad de MongoDB al reiniciar	El actor selecciona la opción Reiniciar ubicada debajo del	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema.

los datos consumidos del día al reiniciar el tiempo consumido del día	consumo de datos del día de un usuario y la base de datos MongoDB no se encuentra disponible.	MongoDB.	<ul style="list-style-type: none"> • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de tiempo del día. • El sistema muestra un mensaje de error.
Problemas de disponibilidad de MongoDB al reiniciar los datos consumidos de la semana.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de datos de la semana de un usuario y la base de datos MongoDB no se encuentra disponible.	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de MongoDB.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de datos de la semana. • El sistema muestra un mensaje de error.
Problemas de disponibilidad de MongoDB al reiniciar el tiempo consumido de la semana.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de datos de la semana de un usuario y la base de datos MongoDB no se encuentra disponible.	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de MongoDB.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de tiempo de la semana. • El sistema muestra un mensaje de error.
Problemas de disponibilidad de MongoDB al reiniciar	El actor selecciona la opción Reiniciar ubicada debajo del	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema.

los datos consumidos del mes.	consumo de datos del mes de un usuario y la base de datos MongoDB no se encuentra disponible.	MongoDB.	<ul style="list-style-type: none"> • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de datos del mes. • El sistema muestra un mensaje de error.
Problemas de disponibilidad de MongoDB al reiniciar el tiempo consumido del mes.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de datos del mes de un usuario y la base de datos MongoDB no se encuentra disponible.	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de MongoDB.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de tiempo del mes. • El sistema muestra un mensaje de error.
Problemas de disponibilidad de MongoDB al reiniciar los datos consumidos del año.	El actor selecciona la opción Reiniciar ubicada debajo del consumo de datos del año de un usuario y la base de datos MongoDB no se encuentra disponible.	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de MongoDB.	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema. • El usuario selecciona la acción "Detalles" de un usuario listado. • El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado. • El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de datos del año. • El sistema muestra un mensaje de error.
Problemas de disponibilidad de MongoDB al reiniciar	El actor selecciona la opción Reiniciar ubicada debajo del	Se muestra un mensaje de imposibilidad de reiniciar la cuota del usuario por fallo de	<ul style="list-style-type: none"> • El usuario una vez autenticado en el sistema selecciona el módulo "Filtrado básico". • Se muestra un listado con los usuarios del sistema.

el tiempo consumido del año	consumo de datos del año de un usuario y la base de datos MongoDB no se encuentra disponible.	MongoDB.	<ul style="list-style-type: none">• El usuario selecciona la acción "Detalles" de un usuario listado.• El sistema muestra en la tabla "Cuota usada" los detalles del consumo de la cuota por día, semana, mes y año del usuario seleccionado.• El usuario selecciona la opción "Reiniciar" ubicada debajo del consumo de tiempo del año.• El sistema muestra un mensaje de error.
-----------------------------	---	----------	--

Tabla 29: Casos de Prueba: Reiniciar cuota a un usuario según parámetro.