Universidad de Cienfuegos “Carlos Rafael Rodríguez”

Facultad de Ingeniería

Carrera de Ingeniería Informática

Título: “Servicio web para la interoperabilidad del sistema InterFirewall”.

Trabajo de diploma para optar por el título de Ingeniería en Informática

Autor(es):

Yairo Sosa Díaz de Villegas

Tutor(es):

Msc. Denis Morejón López

Msc. Anay Carrillo Ramos

Consultante(es):

Ing. Jesús Alberto Leandro León

Ing. Ernesto Fuentes Gómez

Cienfuegos, Cuba

Curso 2016 - 2017

Agradecimientos

A todas la personas involucradas que de una manera u otra me ayudaron en la realización de esta investigación y en especial:

A mis tutores Denis Morejón López y Anay Carrillo Ramos por haberme brindado su apoyo, su tiempo, esfuerzo y confianza.

A mis compañeros de aula en especial Dayron, Richard y David, por haber compartido conmigo todo este tiempo.

A mi familia por haberme brindo todo su amor y haberme convertido en la persona que soy hoy.

A todos muchas gracias.

Dedicatoria

Dedico este trabajo a todos las personas que al igual que yo están luchando por obtener un título universitario y con el mismo doy constancia de que si se puede, lograr dicho sueño.

Resumen.

En la Empresa de Telecomunicaciones de Cuba S.A (ETECSA), en Cienfuegos, es necesario programar un módulo para lograr la interoperabilidad con aplicaciones externas del sistema InterFirewall, que se desarrollando en esta empresa. Con este módulo otras aplicaciones podrían comunicarse con el cortafuegos de forma tal, que si se necesitara bloquear el IP de una PC, por determinados motivos, pudieran emitir la solicitud al cortafuegos y éste llevará a cabo dicha tarea sin la intervención directa de un operador. El módulo se realizará con el lenguaje de programación de alto nivel Python, el framework de desarrollo de aplicación web Django, para la comunicación entre cliente y servidor se utilizará JSON y como gestor de base de datos se empleará PostgreSQL.

Abstract

In order to external applications interoperate with InterFirewall system that is developing, at this time, in the Cuban Telecommunication Enterprise S.A (ETECSA), in Cienfuegos Province, a module has been needed. Using this module, other applications could intercommunicate to a firewall, such in case, an IP of a PC needs to be blocked for any reason, these applications could send a request directly to the firewall and it fulfills the task without an operator. This module will be developed using Python high level programming language, Django web application developing framework, JSON to communicate client to server and PostgreSQL as a data base manager.

Índice

Índice de tablas

Índice de figuras

Introducción:

La programación para internet cambia continuamente debido a que los usuarios necesitan del navegador Web para acceder a una serie de servicios que se encuentran disponibles en la red de redes. Estos cambios han provocado un giro hacia la construcción de aplicaciones orientadas a servicios, en lugar de continuar construyendo aplicaciones de escritorio. Esto trae consigo el aumento del intercambio de información creando así vulnerabilidad en las redes. Para controlar este tráfico se crearon los cortafuegos perimetrales los mismos pueden ser implementados en hardware o software, o una combinación de ambos.

Se implementan también otras medidas para asegurar las redes como son:

Sistema Antivirus.

Sistemas Detectores de Intrusos (IDS, por sus siglas en inglés).

Sistemas para la supervisión de tráfico.

Todos estos sistemas por si solos tiene limitantes por lo que se intenta que trabajen en conjuntos pero no siempre son capaces de comunicarse entre sí por lo que aparecen los llamados servicio web.

Un servicio web es un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

Situación problémica:

En la división territorial de ETECSA en la provincia de Cienfuegos, se lleva acabo el desarrollo del sistema InterFirewall. Este se empleará para controlar tres tipos de routers de fabricantes distintos al mismo tiempo permitiendo administrar la red, pero este no se puede comunicar con otros programas.

Por ejemplo, si el antivirus detecta una anomalía en una PC este no puede comunicarle al routers que bloque el IP de dicha máquina.

En el ejemplo anterior el programa maligno puede atacar los servicios críticos de la red dejando la empresa sin la posibilidad de realizar las operaciones que tiene automatizadas.

Problema a resolver:

En la división territorial de ETECSA en la provincia de Cienfuegos, se lleva a cabo la creación del sistema nombrado InterFirewall. Éste no cuenta con la capacidad de comunicarse con otras aplicaciones.

Objetivo de estudio:

Los protocolos de la tecnología Servicio Web.

Campo de acción:

El sistema InterFirewall que se encuentra en fase de desarrollado en la división territorial ETECSA en Cienfuegos.

Idea a defender:

El desarrollo de un módulo para la interoperabilidad del sistema InterFirewall con otras aplicaciones externas posibilitaría un mayor control, brindando un elevado nivel de seguridad en la red de la empresa ETECSA en Cienfuegos.

Objetivos generales:

Proponer un módulo para la interoperabilidad del sistema con otras aplicaciones externas.

Objetivos Específicos:

- Estudiar los servicios web.

- Diseñar un módulo que permita la interoperabilidad de las aplicaciones.
- Implementar un módulo que permita la interoperabilidad de las aplicaciones.
- Validar el módulo mediante las pruebas funcionales.

Tareas desarrolladas para el cumplimiento de los objetivos:

- Entrevistas con el personal encargado de la seguridad en ETECSA.
- Análisis de las tecnologías utilizadas para la interoperabilidad de sistema.
- Selección de las metodologías, lenguajes, herramientas y tecnologías para la implementación de la aplicación.
- Diseño de la base de datos.
- Implementación de una Api bajo la arquitectura REST.
- Realización del estudio de factibilidad de la solución propuesta.
- Diseño de las pruebas funcionales.
- Elaboración de la documentación del módulo.

Aporte práctico:

El módulo de interoperabilidad del sistema InterFirewall permitirá que este reciba solicitudes de aplicaciones o sistemas externos para la realización de acciones de control.

Estructuración del contenido del trabajo de diploma:

El presente trabajo, está estructurado en 3 capítulos, a los que se le hace referencia a continuación:

Capítulo 1: Fundamentación teórica:

En este capítulo se explica la fundamentación teórica del tema y los conceptos asociados al dominio del problema. Se plantea el problema a resolver y el campo de acción donde se desarrolla, se expone las tecnologías, lenguajes y metodologías utilizadas para su desarrollo.

Capítulo 2: Construcción de la solución propuesta:

Este capítulo describe el proceso llevado a cabo para la construcción de la solución problémica planteada. Describe de forma general el funcionamiento de la aplicación. Define requerimientos funcionales y no funcionales, se describe el modelo de casos de uso, el diseño e implementación del módulo.

Capítulo 3: Validación y Estudio de factibilidad:

En este capítulo se describen los resultados obtenidos al someter a prueba al sistema así como sus validaciones. Además se realiza un estudio de factibilidad de la aplicación que se propone, teniendo en cuenta la estimación del esfuerzo, los beneficios tangibles e intangibles y el análisis beneficio–costo.

1. Fundamentación teórica

1.1. Introducción:

En este capítulo se lleva a cabo la fundamentación teórica, en él se muestra los principales conceptos del dominio del problema. Se da a conocer lenguaje, tecnologías, metodologías y herramientas actuales seleccionadas para dar solución a la problemática propuesta.

1.2. Servicios Web:

Un servicio web es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como internet.[1]

1.3. Tipos de lenguajes de los Servicios web:

Los servicios web son un componente de software que se comunica con otras aplicaciones codificando los mensajes en XML (“Extensible Markup Language“) o en JSON (“JavaScript Object Notation”) y enviando estos mensajes a través de protocolos estándares de Internet tales como HTTP.

1.3.1. XML:

XML es un formato universal para documentos y datos estructurados en Internet; este estándar permite el intercambio de información estructurada entre diferentes plataformas. Por lo que se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.[2]

1.3.2. JSON:

JSON (*JavaScript Object Notation*) es un formato para el intercambios de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. JSON nació como una alternativa a XML, el fácil uso en javascript ha generado un gran número de seguidores de esta alternativa. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.[3]

1.3.3. Características de XML y JSON:

En la siguiente tabla se muestran las características de XML y JSON, así como también las ventajas y desventajas de estos.

	XML	JSON
Características	• Extensibilidad • Estructura • Validación • Basado en texto. • Orientado a los contenidos no presentación. • Las etiquetas se definen para crear los documentos, no tienen un significado preestablecido. • No es sustituto de HTML. • No existe un visor genérico de XML.	• Soporta dos tipos de estructura, una de ellas son objetos que contienen una colección de pares llave-valor y el otro trata arrays de valores. • Cada objeto es un conjunto de claves independientes de cualquier otro objeto. • No necesita ser extensible porque es flexible por sí solo. • Presenta una sintaxis sencilla.
Ventajas	• Tiene un formato muy estructurado y fácil de comprender. • Puede ser validado fácilmente mediante Schemas(XSD)	• Formato sumamente simple • Velocidad de procesamiento alta • Archivos de menor tamaño

	• Se pueden definir estructuras complejas y re utilizables.	
Desventajas	• Es más complicado de entender • El formato es sumamente estricto. • Lleva más tiempo procesarlo	• Tiene una estructura enredosa y difícil de interpretar a simple vista

Tabla 1.Características, ventajas y desventajas de XML y JSON.

1.4. Selección del lenguaje a utilizar por el Servicio Web:

Se decidió utilizar el formato JSON para la comunicación entre la aplicación y el servicio web, debido a que este es mucho más sencillo desde el punto de vista sintáctico; presenta una velocidad de procesamiento alta y sus archivos son de menor tamaño. Por tales motivos se realizó un estudio más profundo del mismo.

1.5. Tipos de Servicios web:

El concepto ha sido perfilado en varios trabajos del comité Web Service Activity perteneciente al W3C, particularmente con la propuesta del protocolo SOAP. Ha sido utilizado desde su concepción para automatizar el intercambio empresarial. No obstante el concepto se ha enriquecido con la profundización de las nociones de recurso y de estado, dentro del comité de modelación REST y en la profundización de la noción de servicio dentro con el advenimiento de SOAP.[4]

En este sentido, podemos distinguir dos tipos de servicios Web: los servicios Web SOAP, y servicios Web REST.

1.5.1.SOAP:

SOAP es un protocolo para el intercambio de mensajes sobre redes de computadoras, generalmente usando HTTP. Está basado en XML, esto facilita la lectura por parte de los humanos, pero también los mensajes resultan más largos y, por lo tanto, considerablemente más lentos de transferir.[6]

1.5.2.REST:

REST (Representational State Transfer) es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP.

REST permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP, por lo que es increíblemente más simple y convencional que otras alternativas que se han usado en los últimos diez años como SOAP y XML-RPC.

Existen tres niveles de calidad a la hora de aplicar REST en el desarrollo de una aplicación web y más concretamente una API que se recogen en un modelo llamado Richardson Maturity Model en honor a la persona que lo estableció, Leonard Richardson padre de la arquitectura orientada a recursos. Estos niveles son:

- Uso correcto de URIs
- Uso correcto de HTTP.
- Implementar Hipermedia.[7]

1.6. Características de REST y SOAP:

En la siguiente tabla se muestran las características entre REST y SOAP, así como también las ventajas y desventajas de estos.

	REST	SOAP
Características	• Las operaciones se definen en los mensajes. • Una dirección única para cada instancia del proceso. • Cada objeto soporta las operaciones estándares definidas. • Componentes débilmente acoplados.	• Las operaciones son definidas como puertos WSDL. • Dirección única para todas las operaciones. • Múltiples instancias del proceso comparten la misma operación. • Componentes fuertemente acoplados.
Ventajas	• Bajo consumo de recursos. • Las instancias del proceso son creadas explícitamente. • El cliente no necesita información de enrutamiento a partir de la URI inicial. • Los clientes pueden tener una interfaz ¨listener¨ (escuchadoras) genérica para las notificaciones. • Generalmente fácil de construir y adoptar.	• Fácil (generalmente) de utilizar. • La depuración es posible. • Las operaciones complejas pueden ser escondidas detrás de una fachada. • Envolver APIs existentes es sencillo. • Incrementa la privacidad. • Herramientas de desarrollo.
Desventajas	• Gran número de	• Los clientes

	objetos. • Manejar el espacio de nombres (URIs) puede ser engorroso. • La descripción sintáctica/semántica muy informal (orientada al usuario). • Pocas herramientas de desarrollo.	necesitan saber las operaciones y su semántica antes del uso. • Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones. • Las instancias del proceso son creadas implícitamente.

Tabla 2.Características, ventajas y desventajas de REST y SOAP.

1.7. Diferencias entre REST y SOAP:

En la siguiente tabla se muestran las diferencias entre REST y SOAP desde varios puntos de vista.

	REST	SOAP
WEB	"La Web es el universo de la información accesible globalmente" (Tim Berners Lee).	"La Web es el transporte universal de mensajes"
Tecnología	• Interacción dirigida por el usuario por medio de formularios. • Pocas operaciones con muchos recursos. • Mecanismo consistente de nombrado de recursos (URI). • Se centra en la	• Flujo de eventos orquestados. • Muchas operaciones con pocos recursos. • Falta de un mecanismo de nombrado. • Se centra en el diseño de aplicaciones distribuidas.

	escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia.	
Protocolo	• XML auto descriptivo. • HTTP. • HTTP es un protocolo de aplicación. • Síncrono.	• Tipado fuerte, XML Schema. • Independiente del trasporte. • HTTP es un protocolo de trasporte. • Síncrono y Asíncrono
Descripción del servicio	• Confía en documentos orientados al usuario que define las direcciones de petición y las respuestas. • Interactuar con el servicio supone horas de testeado y depuración de URIs. • No es necesario el tipado fuerte, si ambos lados están de acuerdo con el contenido. • WADL propuesto en noviembre del 2006.	• WSDL. • Se pueden construirá automáticamente stubs (clientes) por medio del WSDL. • Tipado fuerte. • WSDL 2.0.
Gestión de estado	• El servidor no tiene estado (stateless).	• El servidor puede mantener el estado

	• Los recursos contienen datos y enlaces representando estados válidos. • Los clientes mantienen el estado siguiendo los enlaces. • Técnicas para añadir sesiones: Cookies.	de la conversación. • Los mensajes solo contienen datos. • Los clientes mantienen el estado suponiendo el estado del servicio. • Técnicas para añadir sesiones: Cabecera de sesión (no estándar)
Seguridad	• HTTPS. • Implementado desde hace muchos años. • Comunicación punto a punto segura. • Identificar recursos a ser expuestos como servicios. • Definir URLs para direccionarlos. • Distinguir los recursos de solo lectura (GET) de los modificables (POST, PUT, DELETE). • Implementar e implantar el servidor Web.	• WS-Security. • Las implementaciones están comenzando a aparecer ahora. • Comunicación origen a destino segura. • Listar las operaciones del servicio en el documento WSDL. • Definir un modelo de datos para el contenido de los mensajes. • Elegir un protocolo de transporte apropiado y definir las correspondientes políticas QoS, de seguridad y transaccional.

		• Implementar e implantar el contenedor del servidor Web.

Tabla 3.Diferencia entre REST y SOAP.

1.8. Selección del tipo de Servicio Web:

En este trabajo se seleccionó el Servicio Web de tipo REST por los siguientes aspectos:

- REST es un Servicio Web fácil de construir y adoptar.
- El cliente no necesita información de enrutamiento a partir de la URL inicial.
- El framework que se utilizará para la implementación del subsistema posee una herramienta nombrada Django Rest Framework que nos permite construir API bajo la arquitectura REST.

1.9. Tendencias, metodologías y/o tecnologías actuales:

1.9.1.Metodología RUP:

Cuando se habla de metodologías de desarrollo de software, se está definiendo el "conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software."

La metodología RUP, abreviatura de Rational Unified Process *(o Proceso Unificado Racional*), es un proceso propietario de la ingeniería de software creado por Rational Software , adquirida por IBM , ganando un nuevo nombre Irup que ahora es una abreviatura Rational Unified Process y lo que es una marca en el área de software, proporcionando técnicas que deben seguir los miembros del equipo de desarrollo de software con el fin de aumentar su productividad en el proceso de desarrollo.[8]

1.9.2.UML:

Es un lenguaje para hacer modelos y es independiente de los métodos de análisis y diseño. Existen diferencias importantes entre un método y un lenguaje de modelado. Un *método* es una manera explícita de estructurar el pensamiento y las acciones de cada individuo. Además, el método le dice al usuario qué hacer, cómo hacerlo, cuándo hacerlo y por qué hacerlo; mientras que el lenguaje de modelado carece de estas instrucciones. Los métodos contienen modelos y esos modelos son utilizados para describir algo y comunicar los resultados del uso del método.[9]

Se utilizó la metodología RUP junto con UML porque es una de las metodologías de software más robustas y completas a nivel mundial. Esta es iterativa e incremental y genera gran cantidad de documentación dando paso a que la actualización y mejoras del subsistema se puedan hacer por otras personas de manera rápida. También se utilizó por las ventajas que este presenta.

1.10. Uso de lenguajes y Tecnologías Web:

1.10.1. Python:

Python es un lenguaje de programación creado por Guido van Rossum a principios delos años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “MontyPython”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible.

Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.[10]

Se seleccionó este lenguaje a petición del cliente.

1.10.2. Framework Django:

Django es un framework de desarrollo Web que ahorra tiempo y hace que el desarrollo Web sea divertido. Utilizando Django puedes crear y mantener aplicaciones Web de alta calidad con un mínimo esfuerzo.

Proporciona una serie de herramientas para facilitar la creación de páginas, siguiendo los principios DRY (Don't Repeat Yourself; No Te Repitas) para evitar duplicidad en las líneas de código e invertir el menor esfuerzo posible. Por ejemplo, levantar un panel de administración básico sólo requiere un par de líneas de Python.

También se adscribe al diseño MVC (Modelo-Vista-Controlador), por lo que las diferentes partes del sitio están claramente separadas. Por ejemplo, el código de acceso a los datos es completamente independiente al que gobierna el aspecto externo de la página.[11]

El MVC de Django presenta la siguiente estructura:

Modelo:

- Abstracción del acceso a datos.
- Define los objetos de nuestra base de datos.
- Permite establecer relaciones entre los objetos.
- Una vez definidos nos permite crearlos en nuestra base de datos.
- Una vez creados nuestros objetos nos permite acceder a ellos cómodamente.

Vista:

- Generación y presentación del documento presentado al usuario.
- Ejecuta las operaciones necesarias para devolver el HTML al cliente.
- Es el encargado de la lógica de programa.
- Hace uso del modelo para extraer los datos de la base de datos.
- Hace uso de las plantillas para generar el código HTML.

Controlador:

- Redirige las consultas a la vista apropiada.
- Asocia consultas (urls) a vistas.
- Usa expresiones regulares para determinar las equivalencias.
- Es capaz de capturar parámetros desde la url.
- El uso de expresiones regulares nos permite un primer nivel de validación.

Plantilla:

- Generación del HTML a partir de unos datos.
- Nos permiten crear la presentación de los datos.
- Genera HTML a partir de un código HTML y una serie de variables y estructuras simples.
- Las plantillas que incluye Django soportan herencia, bucles, bifurcaciones, variables y filtros.

Django también incluye una Interfaz de administración que nos permite administrar nuestros objetos de la base de datos. Facilita la inserción de datos durante el proceso de desarrollo. Pudiéndose personalizar, tanto el aspecto como el funcionamiento de este.

En el estudio del framework Django conocimos la existencia del Django Rest framework. Este es una aplicación Django que permite construir proyectos de software bajo la arquitectura REST, incluye gran cantidad de código para reutilizar y una interfaz administrativa desde la cual es posible realizar pruebas sobre las operaciones HTTP como lo son: POST y GET. Esta nos facilitará el desarrollo de la aplicación.

1.10.3. Gestor de Base de datos PostgreSQL:

PostgreSQL es un potente sistema de base de datos objeto-relacional de código abierto. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad e integridad de datos. Se ejecuta en los principales sistemas operativos que existen en la actualidad como:

- Linux
- UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64)
- Windows[12]

1.11. Herramienta a Utilizar:

1.11.1. Visual Paradigm:

Es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

1.11.2. PyCharm:

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web con varios precios.[13]

1.12. Esquema general de la solución propuesta:

1.12.1. InterFirewall:

Esta aplicación manejará de manera dinámica los cortafuegos en la División Territorial de ETECSA en Cienfuegos (DTCF). El sistema manejaría en un principio los cortafuegos que ya se utilizan en la DTCF para en un futuro incluir

soporte para más tipos de cortafuegos. Éste desde la red administrará las reglas necesarias y comunes en todos los cortafuegos, de esta manera creará una capa de abstracción sobre las interioridades sintácticas de cada cortafuego para realizar las tareas de forma centralizada.

1.12.2. Servicio Web para el InterFirewall:

Esta aplicación permitirá la comunicación entre el InterFirewall, utilizando formato JSON, con otras aplicaciones como es el Kaspersky. Permitirá controlar el InterFirewall a través de una interfaz web permitiendo poder acceder a ella desde cualquier parte siempre que se tenga permiso.

1.13. Conclusiones:

En este capítulo se introdujo a los Servicios Web. Después de haber hecho un análisis de las tecnologías y tendencias web, lenguajes de programación y base de datos, se seleccionó la metodología RUP como guía para la documentación del software propuesto. Se escogió la arquitectura REST para la implementación del módulo. Para el intercambio de información se utilizará JSON. El sistema de gestor de bases de datos utilizado es PostgreSQL. Como lenguaje de programación Python con el framework Django para ejecutar los scripts del lado del servidor y del lado del cliente.

2.Construcción de la solución propuesta

2.1. Introducción:

Este capítulo describe el proceso llevado a cabo para la construcción de la solución problémica planteada a través de los flujos de trabajo de RUP: Modelamiento del Dominio, Requisitos y Diseño. Se describe de forma general el funcionamiento de la aplicación e identifican los requerimientos funcionales y no funcionales, describen el modelo de casos de uso, el diseño lógico y físico de base de datos para la implementación del módulo.

2.2. Modelo de dominio:

El modelo de dominio es utilizado por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir.

Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. El modelo de dominio es utilizado por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir.[14]

2.2.1.Definición de las entidades y los conceptos principales:

En el modelo de dominio se define las siguientes clases: Firewall, Zona, Objeto, Acl y Regla.

Firewall: Dispositivo de hardware que permite la interconexión de ordenadores en red.

Zona: Subredes conectadas a un Swicth o router.

Objeto: Es el conjunto de estaciones de red que van hacer objeto de control.

Acl: Las listas de control de acceso o ACL son el conjunto de reglas ordenadas que permiten el control de flujo de tráfico en equipos de redes.

Regla: Norma creada para el control de tráfico de la red, tiene clasificador y acción.

2.2.2.Reglas del negocio a considerar:

Una Zona que se encuentra asignada a un Firewall solo presenta dos Acl una en entrada y una en salida.

Un Objeto pertenece a una sola Zona.

Cuando se cree una Regla esta se le tiene que asignar la Acl a la que pertenece.

Un Puerto puede pertenecer a muchas Zona si se define que es de tipo tronco.

2.2.3.Representación del modelo del dominio:

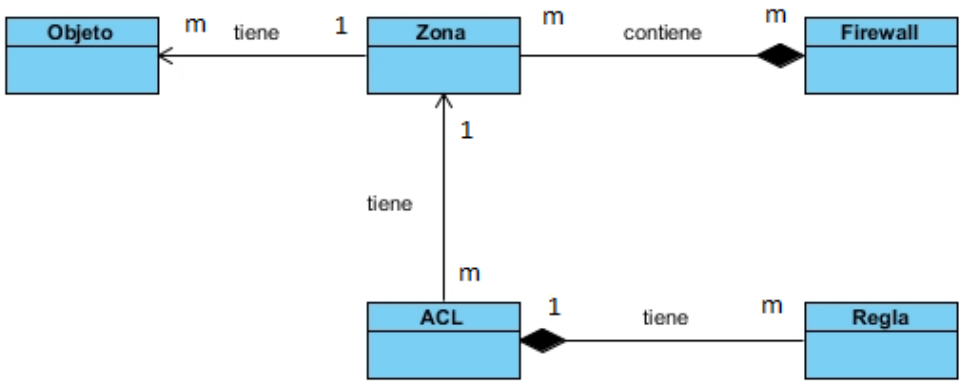


Figura 1. Diagrama de clases del modelo de objetos del dominio.

2.3. Requisitos:

2.3.1.Concepción general del sistema:

El presente proyecto tiene por objeto crear un módulo informático para la interoperabilidad del sistema InterFirewall.

2.3.2.Requerimientos funcionales:

R1.Auntenticarce.

R2.Insertar Zona en la base de datos

R3.Modificar Zona en la base de datos

R4.Eliminar Zona en la base de datos

R5.Insertar Firewall en la base de datos

R6.Modificar Firewall en la base de datos

R7.Eliminar Firewall en la base de datos

R8.Insertar Zona_Firewall en la base de datos

R9.Modificar Zona_Firewall en la base de datos

R10.Eliminar Zona_Firewall en la base de datos

R11.Insertar Puerto en la base de datos

R12.Modificar Puerto en la base de datos

R13.Eliminar Puerto en la base de datos

R14.Insertar ACL en la base de datos

R15.Modificar ACL en la base de datos

R16.Eliminar ACL en la base de datos

R17.Insertar Regla en la base de datos

R18.Modificar Regla en la base de datos

R19.Eliminar Regla en la base de datos

R20. Insertar Servicio en la base de datos

R21. Modificar Servicio en la base de datos

R22. Eliminar Servicio en la base de datos

R23. Insertar Servicio_Configuración en la base de datos

R24. Modificar Servicio_Configuración en la base de datos

R25. Eliminar Servicio_Configuración en la base de datos

R26. Insertar Objeto en la base de datos

R27. Modificar Objeto en la base de datos

R28. Eliminar Objeto en la base de datos

R29. Insertar Miembro en la base de datos

R30. Modificar Miembro en la base de datos

R31. Eliminar Miembro en la base de datos

2.3.3.Requerimientos no funcionales:

Requisitos No Funcionales, son requisitos que imponen restricciones en el diseño o la implementación como restricciones en el diseño o estándares de calidad. Son propiedades o cualidades que el producto debe tener.[15]

Existen múltiples categorías para clasificar a los requerimientos no funcionales, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta.

Usabilidad:

El módulo será utilizado por la interface web y cualquier otra aplicación que tenga la necesidad de comunicarse con el cortafuegos.

Rendimiento:

El módulo deberá responder en el menor tiempo posible ante las solicitudes de información por parte de otros sistemas y ejecutar de manera exitosa las operaciones indicadas.

Portabilidad:

El subsistema debe ser multiplataforma por lo que debe ser posible instalar en los sistemas operativos Windows y Linux.

Legales:

El sistema propuesto debe cumplir con las regulaciones y normas indicadas oficialmente en la División Territorial de ETECSA en la provincia de Cienfuegos.

Fiabilidad:

El sistema deberá estar disponible 24 horas durante los 365 días del año y solo ciertos usuarios tendrán acceso a modificar la información sobre la que basa su funcionamiento.

Software:

El subsistema necesita Python v2.7 y el framework Django v1.8 o superior, un servidor web que puede ser Apache con el módulo mod_python y servidor de base de datos PostgreSQL.

Hardware:

Del lado del servidor se necesita máquinas con capacidad de memoria ram no menor de 1GB, capacidad de almacenamiento con no menos de 20GB, microprocesador Pentium IV a 2,8 MHz como mínimo. Del lado del cliente las terminales deben cumplir con los requisitos mínimos de hardware del sistema operativo que se emplee y se requerirá estar conectados a la red para poder ejecutar los navegadores Web.

Seguridad:

Se garantizará control estricto sobre la seguridad de la información teniendo en cuenta el establecimiento de niveles de acceso. Serán denegados accesos no autorizados.

2.3.4.Modelo de casos de uso del sistema:

El modelo de casos de uso es la técnica más efectiva y a la vez la más simple que emplean los desarrolladores de software para modelar los requisitos del sistema desde la perspectiva del usuario.

“Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios.”[16]

2.3.4.1. Actores del sistema:

Actor	Descripción
Webservice	Es quien utiliza el módulo y se beneficia de las funcionalidades del servicio web (el mismo).

Tabla 4.Actores del sistema.

2.3.4.2. Diagramas de casos de uso del sistema:

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).[17]

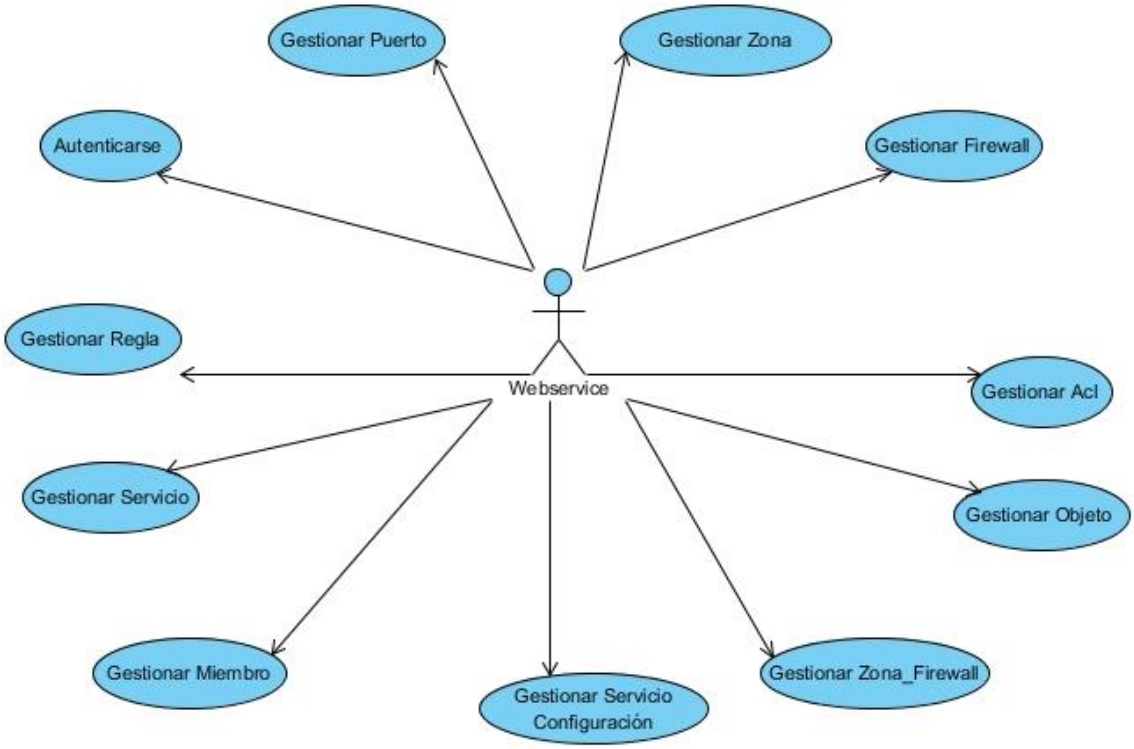


Figura 2.Diagrama de casos de uso del sistema

2.3.4.3. Descripción de casos de uso del sistema:

A continuación, se muestra las descripciones de los casos de uso sistema en forma de tablas.

Caso de uso	Autenticarse
Actores	Webservice

Propósito	Autenticarse para poder acceder al subsistema.
Resumen El caso de uso inicia cuando se desea acceder a las funcionalidades del subsistema. Para esto deben enviar el usuario y la contraseña como variables en la petición, si estos son correctos se le da acceso, en caso contrario se le deniega el acceso; culminando así el caso de uso.	
Referencias	R1
Precondiciones	Debe existir un usuario y contraseña insertados en la base de datos.
Post-condiciones	El usuario puede acceder a las funcionalidades del subsistema.

Tabla 5.Descripción de casos de uso del sistema: Autenticarse.

Caso de uso	Gestionar Zona
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar una Zona. En caso que se desee Insertar una Zona siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de las Zonas. En caso que se desee Eliminar una Zona se eliminará la zona .El caso de uso culmina cuando el subsistema inserta, modifica o elimina una zona.	
Referencias	R5,R6,R7
Precondiciones	El Webservice debe estar autentificado.
Post-condiciones	

Tabla 6.Descripción de casos de uso del sistema: Gestionar Zona.

Caso de uso	Gestionar Firewall
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar un Firewall. En caso que se desee Insertar un Firewall siempre que se le hallan	

agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de los Firewalls. En caso que se desee Eliminar un Firewall se eliminará el firewall .El caso de uso culmina cuando el subsistema inserta, modifica o elimina un firewall.	
Referencias	R8, R9, R10
Precondiciones	El Webservice debe estar autentificado.
Post-condiciones	

Tabla 7.Descripción de casos de uso del sistema: Gestionar Firewall.

Caso de uso	Gestionar Zona_Firewall
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar una Zona_Firewall. En caso que se desee Insertar una Zona_Firewall siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de las Zona_Firewall. En caso que se desee Eliminar una Zona_Firewall se eliminará la zona_firewall. El caso de uso culmina cuando el subsistema inserta, modifica o elimina una zona_firewall.	
Referencias	R11, R12, R13
Precondiciones	El Webservice debe estar autentificado
Post-condiciones	

Tabla 8.Descripción de casos de uso del sistema: Gestionar Zona_Firewall.

Caso de uso	Gestionar Puerto
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar un Puerto. En caso que se desee Insertar un Puerto siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de los Puertos. En caso que se desee Eliminar una Puerto se eliminará el puerto. El caso de uso culmina	

cuando el subsistema inserta, modifica o elimina un puerto.	
Referencias	R14, R15, R16
Precondiciones	El Webservice debe estar autentificado.
Post-condiciones	

Tabla 9.Descripción de casos de uso del sistema: Gestionar Puerto.

Caso de uso	Gestionar ACL
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar una ACL. En caso que se desee Insertar una ACL siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de las ACL. En caso que se desee Eliminar una ACL se eliminará la ACL. El caso de uso culmina cuando el subsistema inserta, modifica o elimina una ACL.	
Referencias	R17, R18, R19
Precondiciones	El Webservice debe estar autentificado.
Post-condiciones	

Tabla 10.Descripción de casos de uso del sistema: Gestionar ACL.

Caso de uso	Gestionar Regla
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar una Regla. En caso que se desee Insertar una Regla siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de las Reglas. En caso que se desee Eliminar una Regla se eliminará la regla. El caso de uso culmina cuando el subsistema inserta, modifica o elimina una regla.	
Referencias	R20, R21, R22
Precondiciones	El Webservice debe estar autentificado.

Post-condiciones	

Tabla 11.Descripción de casos de uso del sistema: Gestionar Regla.

Caso de uso	Gestionar Servicio
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar un Servicio. En caso que se desee Insertar un Servicio siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de los Servicios. En caso que se desee Eliminar un Servicio se eliminará el servicio. El caso de uso culmina cuando el subsistema inserta, modifica o elimina un servicio.	
Referencias	R23, R24, R25
Precondiciones	El Webservice debe estar autentificado.
Post-condiciones	

Tabla 12.Descripción de casos de uso del sistema: Gestionar Servicio.

Caso de uso	Gestionar Servicio_Configuración
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar un Servicio_Configuración. En caso que se desee Insertar un Servicio_Configuración siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de los Servicio_Configuración. En caso que se desee Eliminar un Servicio_Configuración se eliminará el servicio_configuración. El caso de uso culmina cuando el subsistema inserta, modifica o elimina un servicio_configuración.	
Referencias	R26, R27, R28
Precondiciones	El Webservice debe estar autentificado.
Post-condiciones	

Tabla 13.Descripción de casos de uso del sistema: Gestionar Servico_Configuración.

Caso de uso	Gestionar Objeto
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar un Objeto. En caso que se desee Insertar un Objeto siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de los Objeto. En caso que se desee Eliminar un Objeto se eliminará el objeto. El caso de uso culmina cuando el subsistema inserta, modifica o elimina un objeto.	
Referencias	R29, R30, R31
Precondiciones	El Webservice debe estar autentificado.
Post-condiciones	

Tabla 14.Descripción de casos de uso del sistema: Gestionar Objeto.

Caso de uso	Gestionar Miembro
Actores	Webservice(Inicia)
Resumen El caso de uso inicia cuando se desea Insertar, Modificar o Eliminar un Miembro. En caso que se desee Insertar un Miembro siempre que se le hallan agregado los datos correctamente estos se almacenaran. En caso que se quiera Modificar el subsistema permite editar los datos de los Miembro. En caso que se desee Eliminar un Miembro se eliminará el miembro. El caso de uso culmina cuando el subsistema inserta, modifica o elimina un miembro.	
Referencias	R32, R33, R34
Precondiciones	El Webservice debe estar autentificado.
Post-condiciones	

Tabla 15.Descripción de casos de uso del sistema: Gestionar Miembro.

2.4. Diseño:

2.4.1.Diagramas de clases de diseño:

Diagrama de clases de caso de uso: Autenticarse

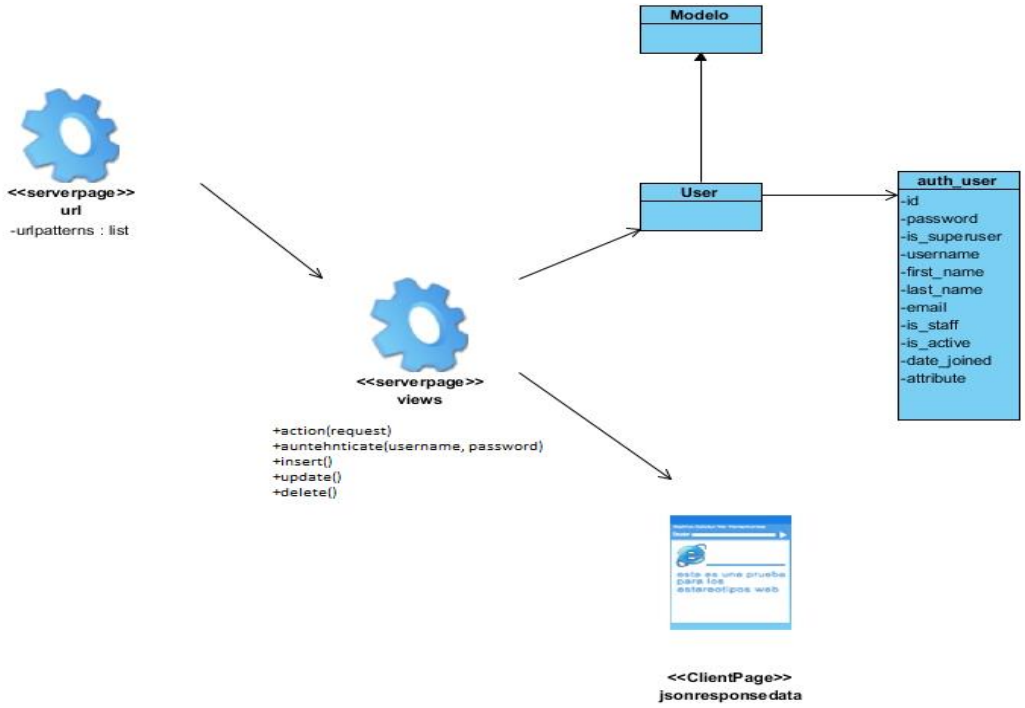


Figura 3.Diagrama de clases 1.

Diagrama de clases de caso de uso: Gestionar Zona

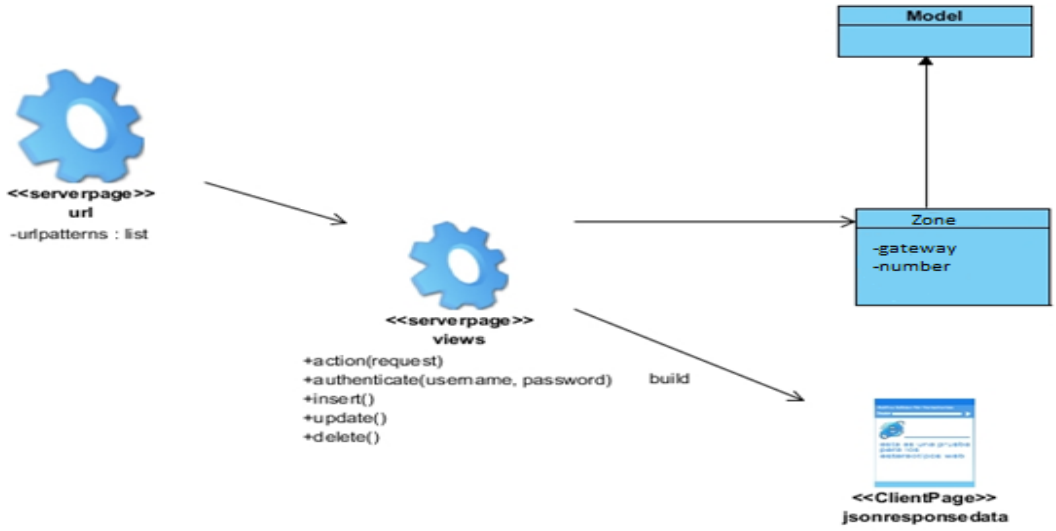


Figura 4.Diagrama de clases 2.

Diagrama de clases de caso de uso: Firewall

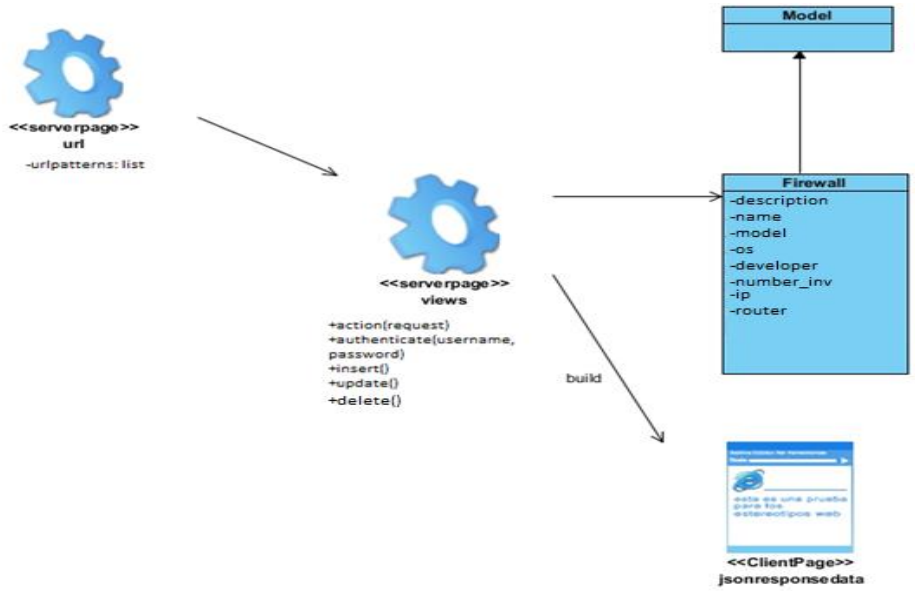


Figura 5.Diagrama de clases 3.

Diagrama de clases de caso de uso: Zona_Firewall

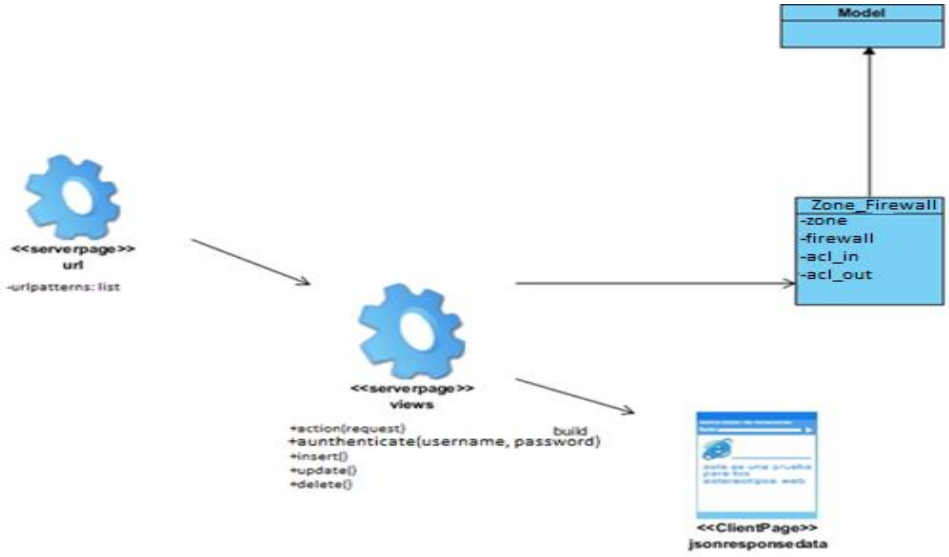


Figura 6.Diagrama de clases 4.

Diagrama de clases de caso de uso: Acl

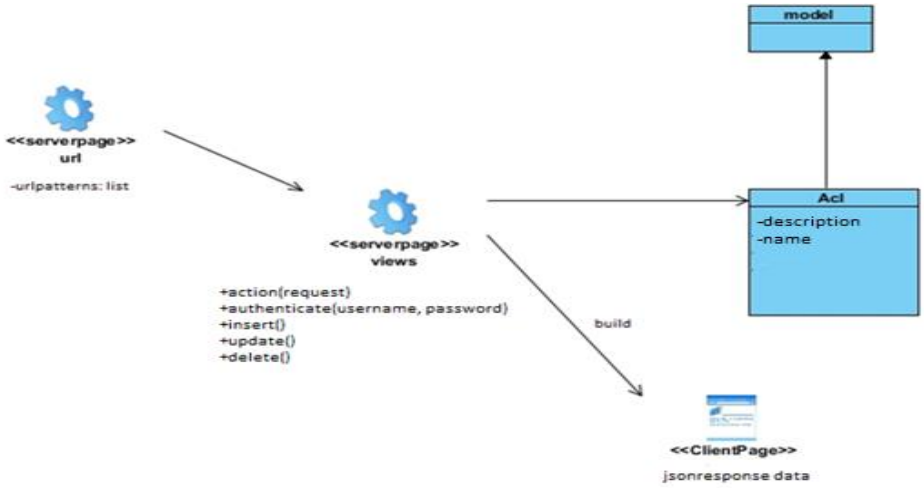


Figura 7.Diagrama de clases 5.

Diagrama de clases de caso de uso: Regla

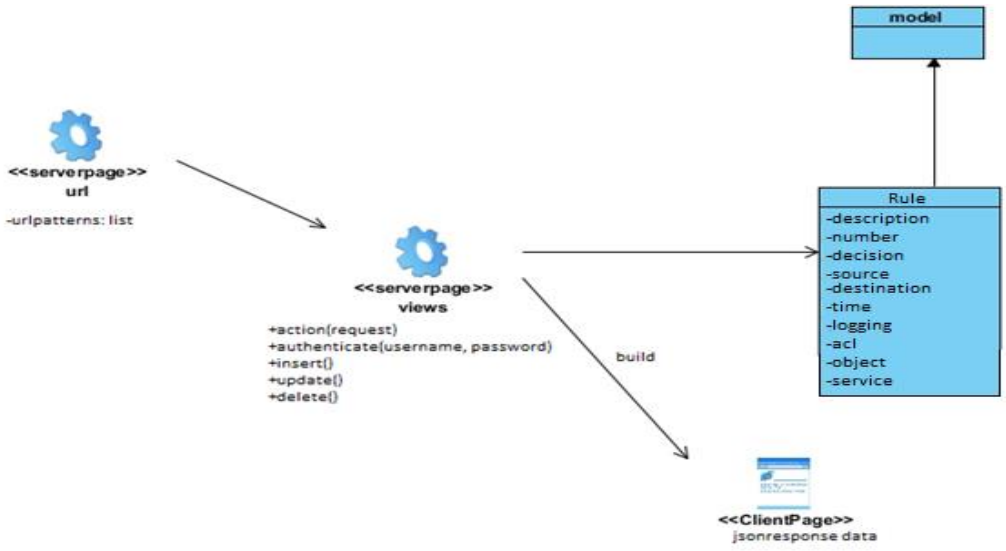


Figura 8.Diagrama de clases 6.

Diagrama de clases de caso de uso: Puerto

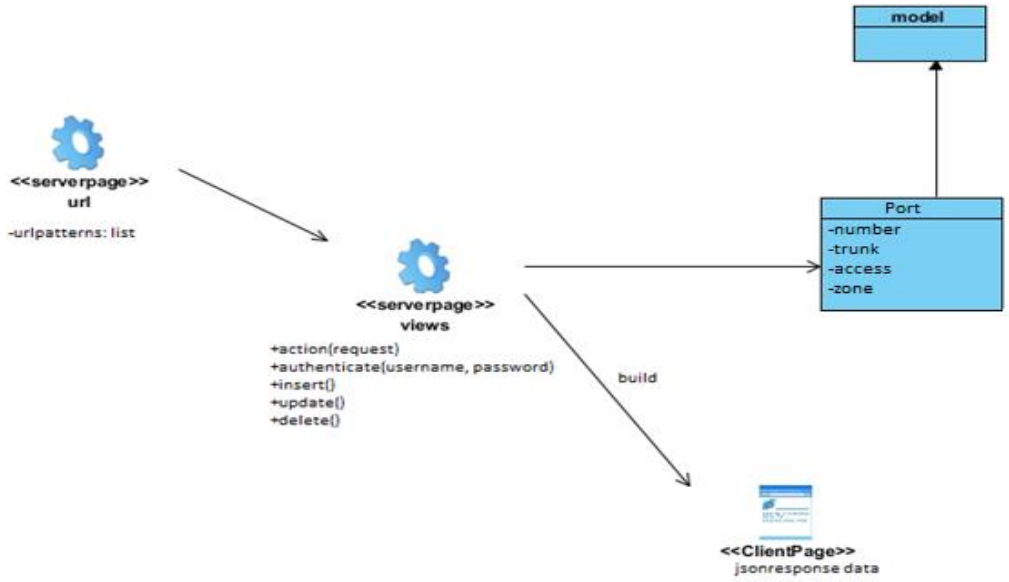


Figura 9.Diagrama de clases 7.

Diagrama de clases de caso de uso: Servicio

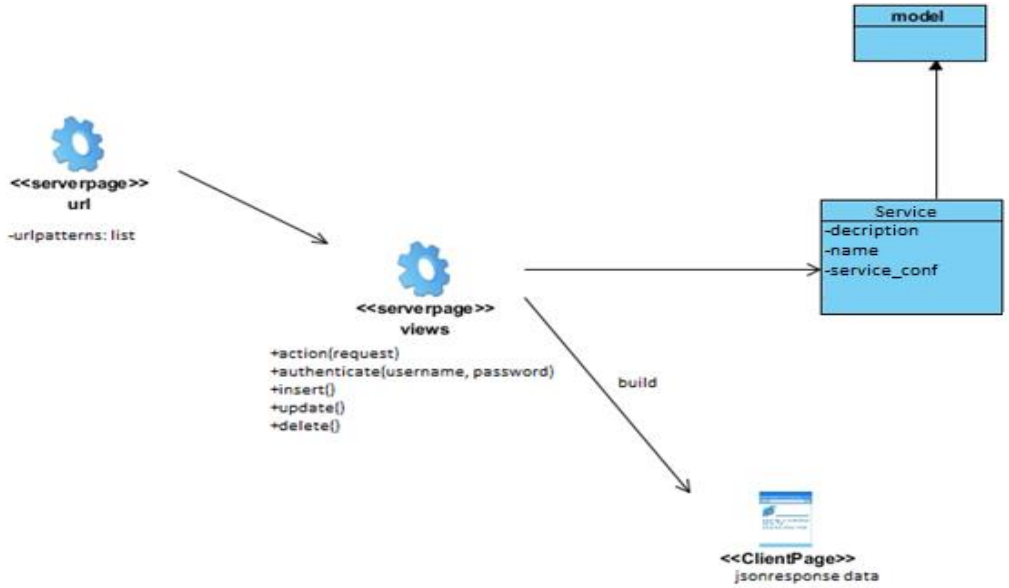


Figura 10.Diagrama de clases 8.

Diagrama de clases de caso de uso: Servicio_Configuración

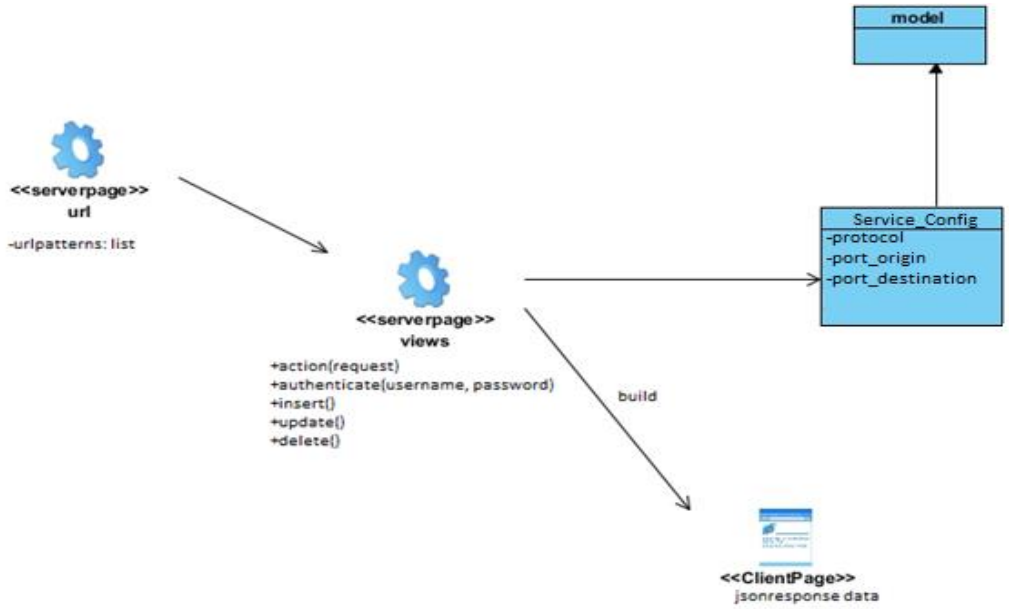


Figura 11.Diagrama de clases 9.

Diagrama de clases de caso de uso: Objeto

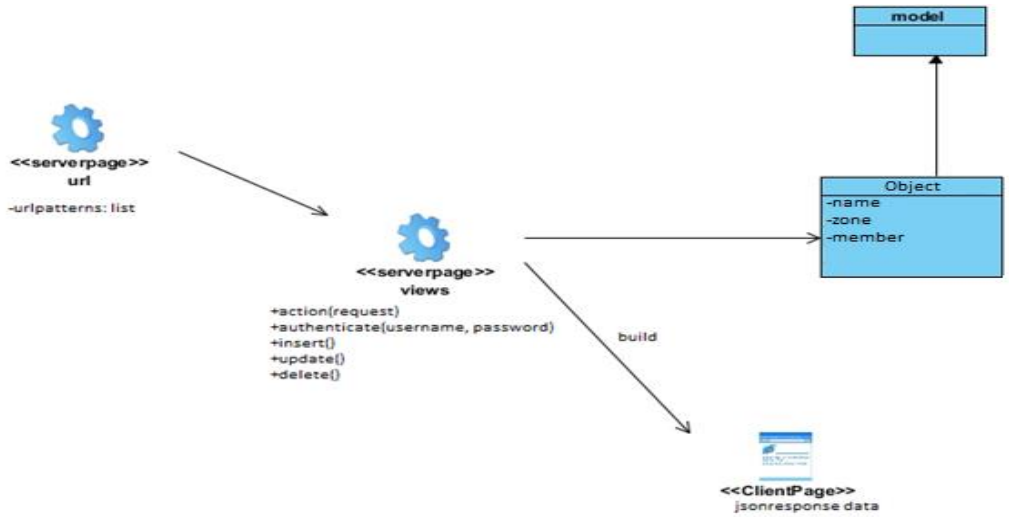


Figura 12.Diagrama de clases 10.

Diagrama de clases de caso de uso: Miembro

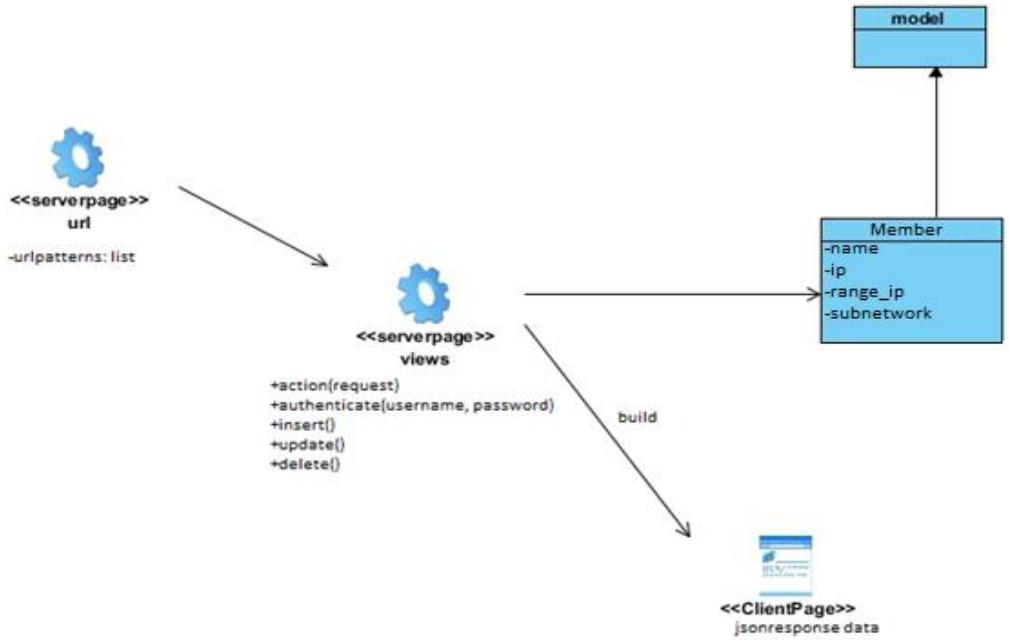


Figura 13.Diagrama de clases 11.

2.4.2.Diseño de base de datos:

Se hace el diseño de la base de datos teniendo en cuenta el modelo entidad relación y las particularidades del framework Django que influye en el hecho de que todas las tablas tengan un id auto numérico como llave primaria.

Modelo lógico de datos:

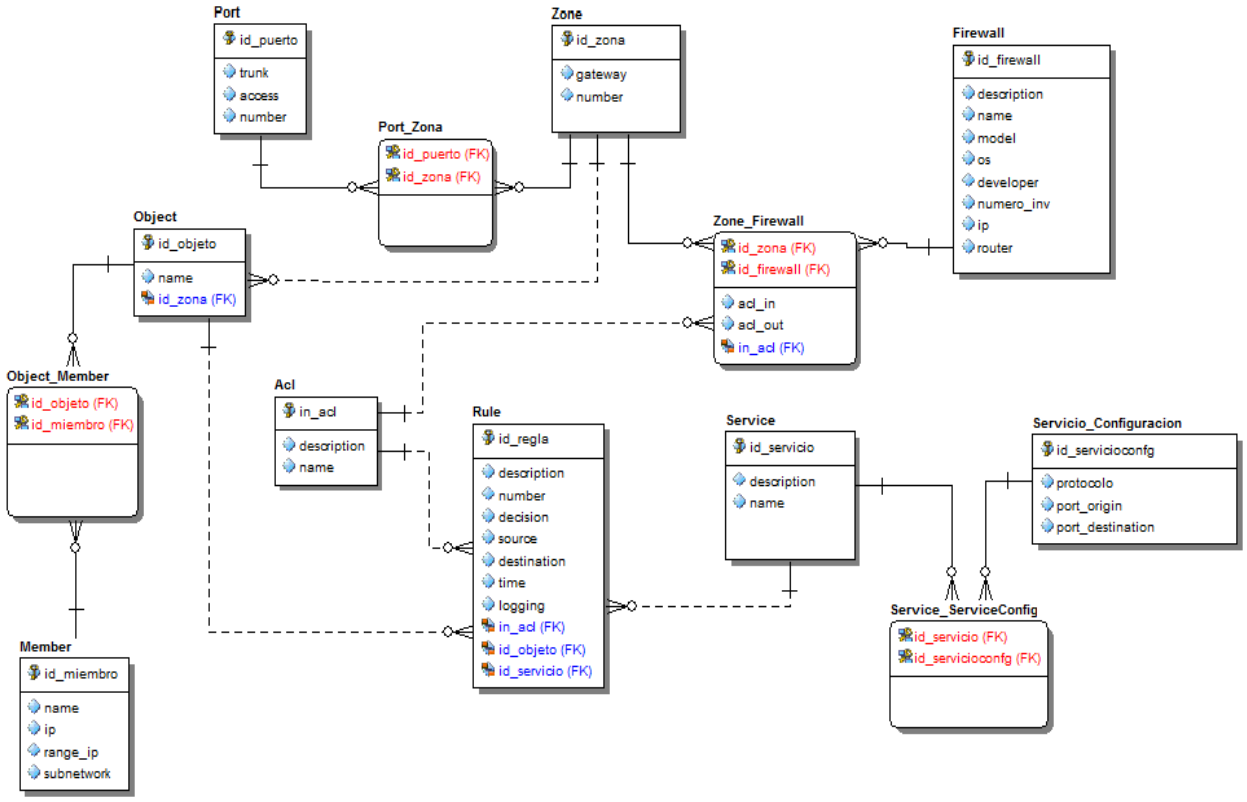


Figura 14.Diagrama del modelo lógico de datos.

Modelo físico de datos

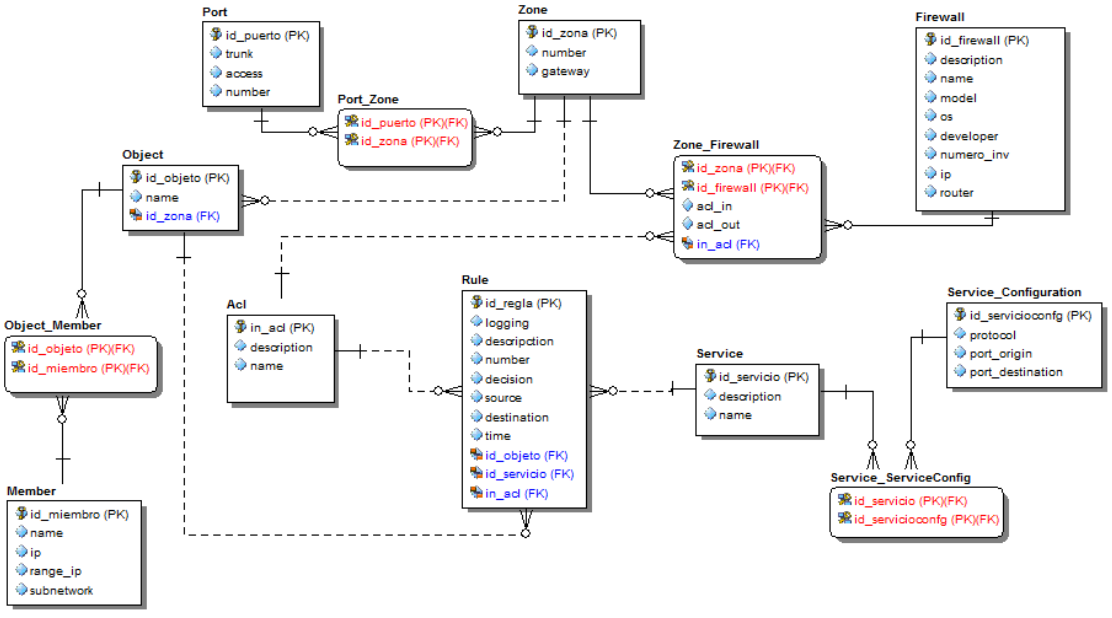


Figura 15.Diagrama del modelo físico de datos.

2.4.3.Diagrama de implementación:

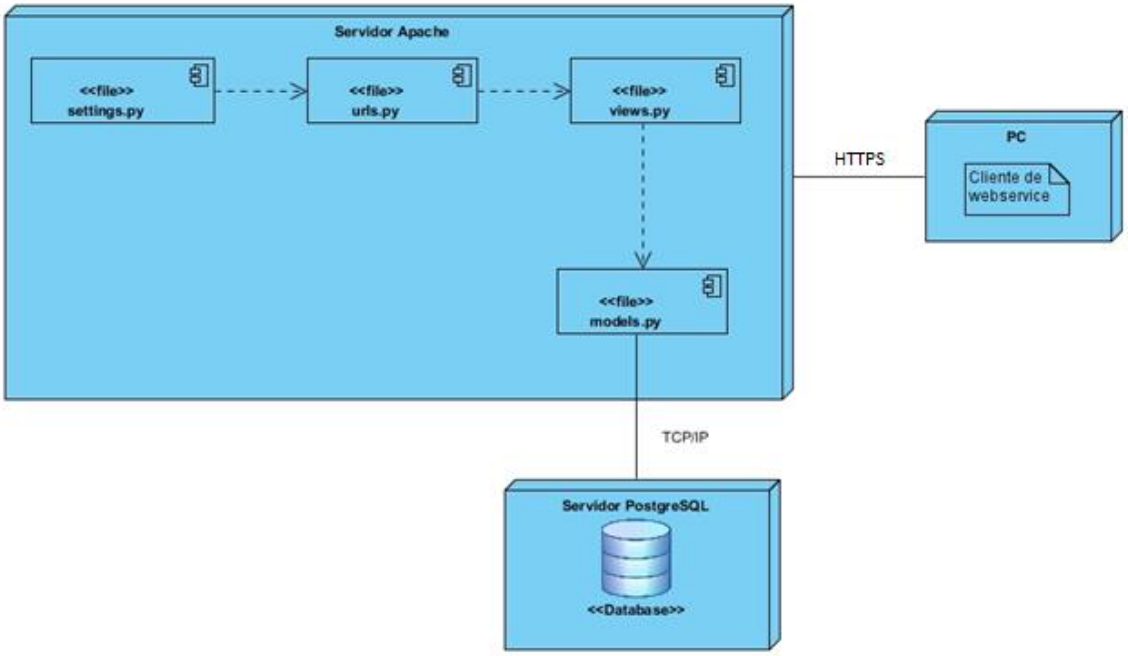


Figura 16.Diagrama de implementación.

2.5. Tratamiento de errores:

Para el tratamiento de excepciones en el sistema propuesto se hizo uso de todas las facilidades que brinda la plataforma utilizada. Igualmente el framework Django brinda un conjunto de excepciones predefinidas que permiten capturar estas situaciones desde la clase controladora y mostrarlas.

2.6. Concepción General de la ayuda:

La plataforma tendrá un manual de ayuda de fácil compresión para usuarios que quieran utilizar el módulo y se inician en la tecnología de los servicios web. También presenta una descripción detallada de las funcionalidades del mismo para futuras modificaciones.

2.7. Conclusiones:

Se definieron los artefactos necesarios, correspondientes a los flujos de trabajo de RUP. Una vez definidos los requerimientos funcionales y no funcionales, se identificaron y describieron los actores del sistema así como los casos de uso a implementar. Se describió la solución propuesta a través de la representación gráfica de los diagramas de clases del modelo de dominio, el

modelo lógico y físico de datos, los diagramas web y el diagrama de implementación.

3. Validación de la solución y Estudio de Factibilidad

3.1. Introducción:

En el presente capítulo se aborda el tema estudio de factibilidad del producto, se ofrece una descripción de la planificación de proyecto, así como los costos asociados al mismo, los beneficios tangibles e intangibles que reportaría su elaboración. Se sometió el modulo a través de escenarios de prueba para comprobar su correcto funcionamiento. Además se diseña los casos de pruebas funcionales en función de la calidad del desarrollo del software.

3.2. Validación de la solución propuesta:

Para la validación del módulo se crearon dos escenarios reales poniendo a prueba las funcionalidades de mismo. Para realizar las validaciones se trabajó con el departamento de seguridad informática ETECSA en Cienfuegos y se tomaron todas las medidas para no poner en riesgo la seguridad de la red.

3.2.1. Escenario de prueba #1:

En el departamento de informática de la empresa ETECSA se introdujo una PC (Computadora Personal) en la red, no perteneciente a ningún trabajador de la empresa. Mediante un Sistema de Detección de Intrusos (IDS) se identificó la dirección IP del intruso y a la subred donde estaba conectado, en este caso la zona Usuarios.

Teniendo ubicado el intruso en la red, se creó una regla para bloquear el acceso de dicha PC a la zona Usuarios. Luego de haberse aplicado la regla se comprobó que la PC ya no tenía acceso a la zona concluyendo que esta validación fue satisfactoria. (Ver anexo 1).

Escenario de prueba #2:

La empresa ETECSA cuenta una infraestructura de red dividida en varias zonas, en las cuales se limita el tráfico entre ellas, para una mejor seguridad de la red. Entre esas zonas se encuentra la zona Servidores, la zona Usuarios y la

zona Administradores, se quiere establecer una conjunto de reglas para que solo tengan acceso a los servidores los administradores, los cuales serán lo únicos autorizados a configurar dichos servidores mediante pruebas de monitorización (ver anexo 2).

Después de realizar los 2 escenarios de prueba el sistema respondió de manera satisfactoria, por lo que podemos decir que el sistema funciona correctamente.

3.3. Pruebas Funcionales del Sistema:

Las pruebas funcionales son un proceso de control de calidad que consiste en asegurar el cumplimiento de un sistema o componente con requerimientos funcionales.

Estas pruebas pueden realizarse durante la fase de desarrollo, individualmente para secciones específicas desarrolladas por su equipo, al final del desarrollo de su proyecto, cuando las diferentes secciones de su proyecto están unidas.

El objetivo principal de las pruebas funcionales es analizar el producto terminado y determinar si hace todo lo que debería hacer y si lo hace correctamente.[18]

Las pruebas funcionales son las que se aplican al producto final, permitiendo detectar los puntos del producto que no cumplen sus especificaciones, es decir, que no funcionan correctamente.

A continuación se muestra algunas de las validaciones realizadas ya que este módulo se encuentra en la parte de atrás consumido por la interfaz web, que es la que se encarga de mostrar los mensajes de error al usuario.

3.3.1.Pruebas funcionales del caso de uso Autentificarse.

Caso de uso	**Autenticarse**
Resumen	Reúnen todo lo referente a los que

	pueden acceder a las funcionalidades del servicio web.
Validaciones: La validación ocurre cuando se hace clic sobre el botón ENTER o al presionar la tecla ENTER cuando algún campo tiene el foco: ➢ Los campos Usuario y Contraseña no pueden estar vacíos. **(ver la validación anterior en el anexo 3)** ➢ El Usuario debe existir en el sistema. **(ver la validación anterior en el anexo 4)** ➢ El campo Contraseña debe coincidir con el del Usuario especificado. **(ver la validación anterior en el anexo 4)**	

Tabla 16. Pruebas funcionales del caso de uso Autenticarse.

3.3.2.Pruebas funcionales del caso de uso Gestionar Zona.

Caso de uso	**Gestionar Zona**
Resumen	Reúnen todo lo referente a la gestión de una zona
Validaciones: La validación ocurre cuando se hace clic sobre el botón POST o al presionar la tecla ENTER cuando algún campo tiene el foco: ➢ El campo number es obligatorio. **(ver la validación anterior en el anexo 5)** ➢ El campo Gateway tiene que ser una subred. **(ver la validación anterior en el anexo 6)**	

Tabla 17. Pruebas funcionales del caso de uso Gestionar Zona.

3.3.3.Pruebas funcionales del caso de uso Gestionar Puerto.

Caso de uso	**Gestionar Puerto**
Resumen	Reúnen todo lo referente a la gestión de un puerto.
Validaciones: La validación ocurre cuando se hace clic sobre el botón POST o al presionar la tecla ENTER cuando algún campo tiene el foco: ➢ El campo number es obligatorio. **(ver la validación anterior en el anexo 7)** ➢ Se debe seleccionar al menos un tipo de puerto. **(ver la validación anterior en el anexo 8)** ➢ Se debe seleccionar al menos un tipo de puerto access o trunk. **(ver la validación anterior en el anexo 9)**	

Tabla 18. Pruebas funcionales del caso de uso Gestionar Puerto.

3.3.4.Pruebas funcionales del caso de uso Gestionar Objeto.

Caso de uso	**Gestionar Objeto**
Resumen	Reúnen todo lo referente a la gestión de un objeto.
Validaciones: La validación ocurre cuando se hace clic sobre el botón POST o al presionar la tecla ENTER cuando algún campo tiene el foco: ➢ El campo service debe ser un ip o un rango o una subred. **(ver la validación anterior en el anexo 10)**	

Tabla 19. Pruebas funcionales del caso de uso Gestionar Objeto.

3.3.5.Pruebas funcionales del caso de uso Gestionar Regla.

Caso de uso	**Gestionar Regla**
Resumen	Reúnen todo lo referente a la gestión de una regla.
Validaciones: La validación ocurre cuando se hace clic sobre el botón POST o al presionar la tecla ENTER cuando algún campo tiene el foco: ➢ El campo number es obligatorio. **(ver la validación anterior en el anexo 11)** ➢ El valor de campo source no puede ser igual al del campo destination. **(ver la validación anterior en el anexo 12)**	

Tabla 20. Pruebas funcionales del caso de uso Gestionar Regla.

3.4. Estimación:

3.4.1.Método de estimación Puntos por Casos de Uso.

El método de Punto de Caso de Uso (UCP – Use Case Point), está basado en los tradicionales Puntos Función. Es un método originado de la tesis de master de Gustav Karner (Karner, 1993), desarrollada mientras trabajaba en Objectory AB, bajo supervisión de Ivar Jacobson (creador de los casos de uso). La técnica ha sido usada por la empresa Rational (posteriormente adquirida por IBM) durante varios años y con buenos resultados. Además la técnica se ha documentado en varias publicaciones (Carroll, 2005; Clemmons, 2006; Karner, 1993; Nageswaran, 2007).[19]

Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

3.4.1.1. Cálculo de Puntos de Casos de Uso sin Ajustar

UUCP = UAW + UUCW

UUCP: Puntos de Casos de Uso sin ajustar.

UAW: Factor de Peso de los Actores sin ajustar.

UUCW: Factor de Peso de los Casos de Uso sin ajustar

Factor de Peso de los Actores sin ajustar (UAW)

Este valor se calcula mediante un análisis de la complejidad de los actores del sistema. Para obtener este valor se le asigna un valor a cada tipo de actor como se muestra en la tabla siguiente**.**

Tipo de Actor	**Descripción**	**Factor de Peso**
Simple	Sistema que interactúan con el sistema a través de una interfaz de programación	1
Medio	Sistema que interactúan con el sistema a través de un protocolo o interfaz basada en texto	2
Complejo	Persona que interactúa con el sistema a través de una interfaz grafica	3

Tabla 21.Factor de Peso de los Actores sin ajustar.

El actor es de tipo medio ya que se utiliza el sistema a través de un protocolo o interfaz basada en texto, por lo que se le asigna un factor de peso 2.

Multiplicando la cantidad de actores de cada tipo por el peso correspondiente se obtiene que:

UAW= (Cantidad de actores) * Peso

UAW = 1*2

UAW = 2

Factor de Peso de los Casos de Uso sin Ajustar

Este valor se calcula mediante un análisis de la complejidad de los casos de uso sin ajustar existentes en el sistema, esta complejidad está dada por la cantidad de transacciones que se realizan, donde una transacción es una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa o no se efectúa ninguna de las actividades de la secuencia.

En la tabla siguiente se dividen los casos de uso del sistema de acuerdo a su complejidad.

Tipo	**Descripción**	**Factor de Peso**
Simple	El caso de uso contiene de 1 a 3 transacciones.	5
Medio	El caso de uso contiene de 4 a 7 transacciones.	10
Complejo	El caso de uso contiene más de 8 transacciones.	15

Tabla 22.División de los casos de usos según su complejidad.

Por tanto los casos de uso del sistema se clasifican como se muestra en la tabla siguiente:

Casos de Uso del Sistema	Clasificación	Peso

Autenticarse	Simple	5
Gestionar Zona	Simple	5
Gestionar Firewall	Simple	5
Gestionar Zona_Firewall	Simple	5
Gestionar Acl	Simple	5
Gestionar Regla	Medio	10
Gestionar Objeto	Simple	5
Gestionar Miembro	Simple	5
Gestionar Servicio	Simple	5
Gestionar Servico_Configurqación	Simple	5
Gestionar Puerto	Simple	5

Tabla 23.Factor de Peso de los Casos de Uso sin Ajustar.

En la tabla de clasificación anterior se observa que el sistema está compuesto por 11 casos de uso, de ellos 10 simple y 1 medios

Calculando el factor de peso de los Casos de Uso como:

UUCW = 10*5 + 1*10

UUCW = 60

Sustituyendo el valor de los puntos de caso de uso sin ajustar es:

UUCP = UAW + UUCW

UUCP = 2 + 60

UUCP = 62

3.4.1.2. Cálculo de Puntos de Casos de Uso ajustados

Una vez que se obtienen los Puntos de Casos de Uso sin ajustar, se debe ajustar este valor mediante la siguiente ecuación:

UCP = UUCP x TCF x EF

Donde:

UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

Cálculo de TCF:

Factor	**Descripción**	**Peso**	**Valor**	**Peso*Valor**
T1	Sistema distribuido	2	3	6
T2	Objetivos de performance o tiempo de respuesta.	1	4	4
T3	Eficiencia del usuario final.	1	4	4
T4	Procesamiento interno complejo.	1	3	3
T5	El código debe ser reutilizable.	1	5	5
T6	Facilidad de instalación.	0,5	4	2
T7	Facilidad de uso.	0,5	4	2
T8	Portabilidad.	1	5	5
T9	Facilidad de cambio.	1	4	4
T10	Concurrencia.	1	4	4

T11	Incluye objetivos especiales de seguridad.	1	3	3
T12	Provee acceso directo a terceras partes.	1	3	3
T13	Se requieren facilidades especiales de entrenamiento a usuarios.	1	1	1

Tabla 24.Cálculo de TCF.

El Factor de Complejidad Técnica resulta:

TCF =0.6 + 0.01* Σ (Peso *Valor asignado)

TCF =0.6 + 0.01*(6+4+4+3+5+2+2+5+4+4+3+3+1)

TCF =0.6 + 0.01*(46)

TCF =1.06

Cálculo de EF:

Factor	**Descripción**	**Aporte**	**Valor**	**Aporte*Valor**
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	4	6
E2	Experiencia en la aplicación.	0.5	4	2
E3	Experiencia en orientación a objetos.	1	4	4

E4	Capacidad del analista líder.	0.5	5	2.5
E5	Motivación.	1	5	5
E6	Estabilidad de los requerimientos.	2	5	10
E7	Personal part-time.	-1	0	0
E8	Dificultad del lenguaje de programación.	-1	1	-1

Tabla 25.Cálculo de EF.

El factor de ambiente se calcula mediante la siguiente ecuación:

EF = 1.4 – 0.03 * Σ (Peso * Valor asignado)

EF = 1.4 – 0.03 * (6 + 2 + 4 + 2.5 + 5 + 10 + 0 - 1)

EF = 1.4 – 0.03 * 28.5

EF = 0.545

Cálculo de UCP:

UCP = UUCP * TCF * EF

UCP = 62 * 1.06 * 0.545

UCP = 35.8174 ≈35.82

Conociendo que:

E: esfuerzo (horas/hombre)

ET: esfuerzo total (horas/hombre)

CF: factor de conversión: CF = 20 horas / hombre

3.4.1.3. Cálculo del esfuerzo:

El esfuerzo en horas-hombre viene dado por:

E = UCP * CF

Donde:

E: esfuerzo estimado en horas/hombre.

UCP: puntos de Casos de Uso ajustados.

CF: factor de conversión.

Cálculo de CF:

CF = 20 horas-hombre (si EF ≤ 2)

CF = 28 horas-hombre (si EF = 3 o EF = 4)

CF = abandonar o cambiar proyecto (si EF ≥ 5)

Por lo tanto, CF = 20 horas-hombre

Cálculo de E:

E = UCP * CF

E = 35.82* 20

E = 716.4 horas-hombre

El esfuerzo calculado corresponde a la implementación y representa el 40% del esfuerzo total, por tanto, se tiene que:

ET = 1676.8

3.4.1.4. Determinación del costo del proyecto

Salario básico: $1000.00

Cantidad de hombres: 1

Se obtiene que:

Duración del proyecto: 1676.8 horas

Si se trabaja:

Al día: 8 horas.

En la semana: 40 horas.

En un mes: 160 Horas

El proyecto tendrá una duración de aproximadamente 10 meses y un costo de $10000

3.4.2.Beneficios tangibles e intangibles.

Los beneficios obtenidos con el desarrollo del módulo permiten mejorar el control y la seguridad en una red, permitiendo que no solo el InterFirewall sea quien administre la red sino cualquier software con la ayuda del mismo.

3.4.3.Análisis de costos y beneficios.

Este módulo, como resultado del presente trabajo, no implica costo alguno para la División Territorial de ETECSA, sin embargo, al desarrollo de todo producto informático conlleva un costo, el de esta aplicación es de 10 000 pesos por concepto de salario y su justificación económica viene dado por los beneficios tangibles e intangibles que este produce.

Para la realización de este módulo no fue necesaria una inversión en los medios técnicos. Estos beneficios implican un ahorro del tiempo que se invierte en la gestión y control de seguridad de una red de computadoras.

3.5. Conclusiones

El desarrollo de la validación del sistema mostró resultados favorables a partir de las situaciones de pruebas que le fueron creadas. Se realizó el estudio de factibilidad del producto informático, estimando un tiempo de desarrollo de aproximadamente 10 meses y un costo por concepto de salario de 10000 pesos. Se determinaron los beneficios tangibles e intangibles obteniendo el esfuerzo, tiempo y costo que implica el desarrollo del sistema.

Conclusiones Generales

- Se estudió los conceptos de los servicios web permitiendo escoger como arquitectura del mismo, REST y como lenguaje para el intercambio de información JSON.
- Se diseñó una arquitectura para el módulo que permite la interacción con aplicaciones tanto locales como externas al servidor.
- Implemento un módulo que permite la interoperabilidad de cortafuegos InterFirewall.
- Se validó el módulo mediante las pruebas funcionales, minimizando la posibilidad de errores y elevando la calidad de software.

Recomendaciones

Encontrar nuevos escenarios de aplicaciones externas que puedan y necesiten interactuar con el InterFirewall para que utilicen el módulo.

Referencias bibliográficas

[1] “¿Qué es y para qué sirve un web service? - Culturación,” 25-May-2017. [Online]. Available: http://culturacion.com/que-es-y-para-que-sirve-un-web-service/. [Accessed: 25-May-2017].
[2] “definicion xml | Educación, Sistemas, Redes y TIC,” 25-May-2017. [Online]. Available: https://conocimientoysistemas.wordpress.com/tag/definicion-xml/. [Accessed: 25-May-2017].
[3] “JSON I - ¿Qué es y para qué sirve JSON?,” 25-May-2017. [Online]. Available: https://geekytheory.com/json-i-que-es-y-para-que-sirve-json. [Accessed: 25-May-2017].
[4] “Servicio Web - EcuRed,” 25-May-2017. [Online]. Available: https://www.ecured.cu/Servicio_Web#Tipos_de_Servicios. [Accessed: 25-May-2017].
[5] “EcuRed (es).” [Online]. Available: http://www.ecured.cu/opensearch_desc.php.
[6] “Definicion de SOAP,” 25-May-2017. [Online]. Available: http://www.alegsa.com.ar/Dic/soap.php. [Accessed: 25-May-2017].
[7] A. Asier, “Conceptos sobre APIs REST,” *Asier Marqués*, 25-May-2017. .
[8] “Metodología RUP,” *Metodoss*, 19-May-2016. [Online]. Available: http://metodoss.com/metodologia-rup/. [Accessed: 25-May-2017].
[9] “¿Qué es UML,” 25-May-2017. [Online]. Available: http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html. [Accessed: 25-May-2017].
[10] “¿Qué es Python?,” 25-May-2017. [Online]. Available: http://mundogeek.net/archivos/2008/01/10/%C2%BFque-es-python/. [Accessed: 25-May-2017].
[11] “Descubre qué es Django, el framework web de moda,” *ComputerHoy*, 25-May-2017. [Online]. Available: http://computerhoy.com/noticias/internet/descubre-que-es-django-framework-web-moda-8641. [Accessed: 25-May-2017].
[12] “¿Qué es PostgreSQL? | Microbuffer.” [Online]. Available: https://microbuffer.wordpress.com/2011/05/04/que-es-postgresql/. [Accessed: 01-Jun-2017].
[13] “PyCharm el IDE de Python lanza su versión 3.4.” [Online]. Available: https://hipertextual.com/archivo/2014/06/pycharm-ide-python/.
[14] “Modelo de dominio - EcuRed.” [Online]. Available: https://www.ecured.cu/Modelo_de_dominio. [Accessed: 20-Mar-2017].
[15] “Requisitos no funcionales - EcuRed.” [Online]. Available: https://www.ecured.cu/Requisitos_no_funcionales. [Accessed: 26-Mar-2017].
[16] “Casos de Uso - CasosDeUso.pdf.” [Online]. Available: http://www-2.dc.uba.ar/materias/isoft1/2001_2/apuntes/CasosDeUso.pdf. [Accessed: 26-Mar-2017].
[17] “Tutorial de UML - Casos de Uso.” [Online]. Available: https://users.dcc.uchile.cl/~psalinas/uml/casosuso.html. [Accessed: 26-Mar-2017].

[18] “Pruebas Funcionales - Software Testing and QA.” [Online]. Available: http://www.calidadysoftware.com/testing/pruebas_funcionales.php. [Accessed: 27-May-2017].
[19] “Método de Estimación de Puntos de Caso de Uso – 233 Grados de TI.” .

Bibliografía

[1] «Web Services - REST vs SOAP | Blog de Tecnología Qode Apps», 25-may-2017. [En línea]. Disponible en: http://qode.pro/blog/web-services-rest-vs-soap/. [Accedido: 25-may-2017].

[2] «SOAP vs REST web services - javatpoint», 25-may-2017. [En línea]. Disponible en: https://www.javatpoint.com/soap-vs-rest-web-services. [Accedido: 25-may-2017].

[3] «SOAP VS REST», 25-may-2017. .

[4] «Servicio Web - EcuRed», 25-may-2017. [En línea]. Disponible en: https://www.ecured.cu/Servicio_Web#Tipos_de_Servicios. [Accedido: 25-may-2017].

[5] «REST vs. SOAP: Choosing the best web service», 25-may-2017. [En línea]. Disponible en: http://searchmicroservices.techtarget.com/tip/REST-vs-SOAP-Choosing-the-best-web-service. [Accedido: 25-may-2017].

[6] «REST vs SOAP | Thanks Network», 25-may-2017. [En línea]. Disponible en: https://thanksnetwork.wordpress.com/2012/10/16/rest-vs-soap/. [Accedido: 25-may-2017].

[7] J. · in Html *et al.*, «REST vs SOAP», *Blog del equipo de desarrollo de software en Qbit Mexhico*, 14-feb-2012. .

[8] «¿Qué es y para qué sirve un web service? - Culturación», 25-may-2017. [En línea]. Disponible en: http://culturacion.com/que-es-y-para-que-sirve-un-web-service/. [Accedido: 25-may-2017].

[9] «¿Qué es XML? - Definición de XML», 25-may-2017. [En línea]. Disponible en: http://www.masadelante.com/faqs/xml. [Accedido: 25-may-2017].

[10] «¿Qué es UML», 25-may-2017. [En línea]. Disponible en: http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html. [Accedido: 25-may-2017].

[11] «¿Qué es SOAP?», 28-feb-2015. [En línea]. Disponible en: https://www.ibm.com/support/knowledgecenter/es/SSKM8N_8.0.0/com.ibm.etools.mft.doc/ac55770_.htm?view=embed. [Accedido: 25-may-2017].

[12] «¿Qué es Python? - Aprende a Programar - Codejobs», 25-may-2017. [En línea]. Disponible en: https://www.codejobs.biz/es/blog/2013/03/02/que-es-python. [Accedido: 25-may-2017].

[13] «¿Qué es Python?», 25-may-2017. [En línea]. Disponible en: http://mundogeek.net/archivos/2008/01/10/%C2%BFque-es-python/. [Accedido: 25-may-2017].

[14] L. A. says, «¿Qué es PostgreSQL?», *Microbuffer*, 04-may-2011. .

[15] W. Chakray, «¿Qué diferencias hay entre REST y SOAP?», *Chakray*, 21-dic-2016. .

[16] «PyCharm el IDE de Python lanza su versión 3.4», 25-may-2017. [En línea]. Disponible en: https://hipertextual.com/archivo/2014/06/pycharm-ide-python/. [Accedido: 25-may-2017].

[17] «Pruebas Funcionales - Software Testing and QA». [En línea]. Disponible en: http://www.calidadysoftware.com/testing/pruebas_funcionales.php. [Accedido: 27-may-2017].

[18] «Pruebas funcionales», *Crowdsourced Testing*. [En línea]. Disponible en: https://crowdsourcedtesting.com/es/pruebas-funcionales. [Accedido: 27-may-2017].

[19] «Modelo de dominio - EcuRed», 25-may-2017. [En línea]. Disponible en: https://www.ecured.cu/Modelo_de_dominio. [Accedido: 25-may-2017].

[20] «Metodología RUP», *Metodoss*, 19-may-2016. [En línea]. Disponible en: http://metodoss.com/metodologia-rup/. [Accedido: 25-may-2017].

[21] «Método de Estimación de Puntos de Caso de Uso – 233 Grados de TI». .

[22] «JSON vs XML - Oscar Blancarte Blog», 25-may-2017. [En línea]. Disponible en: https://www.oscarblancarteblog.com/2014/07/18/json-vs-xml/. [Accedido: 25-may-2017].

[23] «JSON I - ¿Qué es y para qué sirve JSON?», 25-may-2017. [En línea]. Disponible en: https://geekytheory.com/json-i-que-es-y-para-que-sirve-json. [Accedido: 25-may-2017].

[24] «Introducción a JavaScript Object Notation (JSON) en JavaScript y .NET», 25-may-2017. [En línea]. Disponible en: https://msdn.microsoft.com/es-es/library/bb299886.aspx. [Accedido: 25-may-2017].

[25] Center for History and New Media, «Guía rápida». [En línea]. Disponible en: http://zotero.org/support/quick_start_guide.

[26] «Guía Breve de Servicios Web», 25-may-2017. [En línea]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb. [Accedido: 25-may-2017].

[27] «El libro de Django 2.0 - 34ba425e-5985-11e5-964d-04015fb6ba01.pdf». .

[28] «Descubre qué es Django, el framework web de moda», *ComputerHoy*, 25-may-2017. [En línea]. Disponible en: http://computerhoy.com/noticias/internet/descubre-que-es-django-framework-web-moda-8641. [Accedido: 25-may-2017].

[29] «definicion xml | Educación, Sistemas, Redes y TIC», 25-may-2017. [En línea]. Disponible en: https://conocimientoysistemas.wordpress.com/tag/definicion-xml/. [Accedido: 25-may-2017].

[30] «Definicion de SOAP», 25-may-2017. [En línea]. Disponible en: http://www.alegsa.com.ar/Dic/soap.php. [Accedido: 25-may-2017].

[31] Exes, «Cursos oracle,cursos java,master oracle,master java,formación en informatica», 25-may-2017. [En línea]. Disponible en: http://www.mundolinux.info/que-es-xml.htm. [Accedido: 25-may-2017].

[32] «Cuando usar SOAP, Cuando usar REST | $> SoloCodigoWeb», *Solo Codigo Web*, 15-sep-2016. .

[33] A. Asier, «Conceptos sobre APIs REST», *Asier Marqués*, 25-may-2017. .

[34] «caracteristicas xml | Educación, Sistemas, Redes y TIC», 25-may-2017. [En línea]. Disponible en: https://conocimientoysistemas.wordpress.com/tag/caracteristicas-xml/. [Accedido: 25-may-2017].

Glosario de términos

Administrador de red: Una persona responsable de la configuración y administración de la red. El administrador generalmente configura la red, asigna contraseña y permisos, y ayuda a los usuarios.

Antivirus: Son todos aquellos programas que permiten analizar la memoria, las unidades de disco y otros elementos de un ordenador, en busca de virus.

API: Una interfaz de programación de aplicaciones o API (del inglés application programming interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

BD: Una base de datos o banco de datos en ingles Data Base, se refiere a una serie de datos almacenados para ser luego usados.

Cortafuego: Mecanismo que permite que las comunicaciones entre una red local e Internet se realicen conforme a las políticas de seguridad de quien los instala. Estos sistemas suelen incorporar elementos que garantizan la privacidad, autenticación, etc., con lo que se impide el acceso no autorizado desde Internet.

Framework: Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular.

Hardware: Término que hace referencia a cada uno de los elementos físicos de un sistema informático (pantalla, teclado, ratón, memoria, discos duros, microprocesador, etc.).

HTTP: Es el protocolo estándar utilizado por la WWW (Word Wide Web) que define la sintaxis básica que utilizan los elementos de software para poder comunicarse en internet. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

IP: Internet Protocol (Protocolo de Internet). Uno de los protocolos más representativos del estándar TCP/IP, es el responsable del esquema de direccionamiento utilizado en Internet y define la estructura y el intercambio de los datagramas IP entre redes distantes, definido en el RFC 791.

JSON: JavaScript Object Notation (JSON) fue desarrollado en 1999 como alternativa a XML. JSON es un formato para intercambiar datos basados en texto, Se usa para representar objetos de Java Script como cadenas, para luego trasmitirlos por la red. Es comúnmente usado en aplicaciones que utilizan AJAX. El texto JSON es fácil de generar y leer siendo este más rápido de obtener que el equivalente en formato XML.

LAN: Local Area Network (Red de área local). Red de datos para dar servicio a un área geográfica máxima de unos pocos kilómetros cuadrados, por lo cual pueden optimizarse los protocolos de señal de la red para llegar a velocidades de transmisión de Gbps (gigabits por segundo).

PC: Personal Computer (PC). Computadora personal, es una computadora que puede ser una laptop o una computadora de escritorio

Protocolo: Es el sistema, reglas y conjunto de normas que establece, gobierna y permite la comunicación entre dispositivos informáticos (la transferencia de datos).

Python: Es un lenguaje de programación interpretado creado por Guido Van Rossum en el año 1991.Script: Es un guion o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de órdenes y usualmente son archivos de texto.

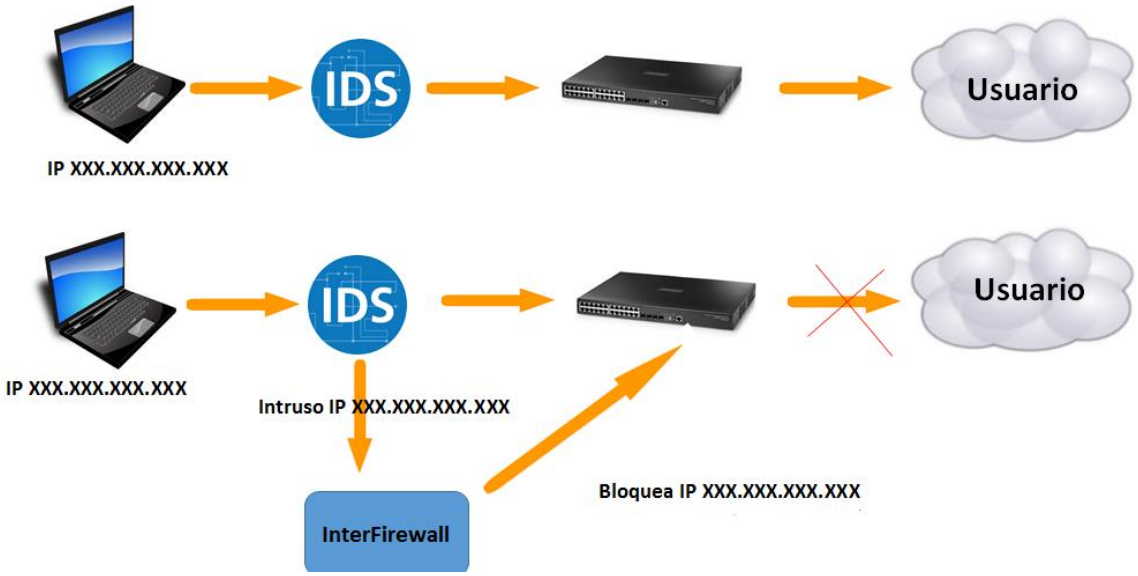
Software: Son los ficheros, programas, aplicaciones y sistemas operativos que nos permiten trabajar con el ordenador o sistema informático. Se trata de los elementos que hacen funcionar al hardware.

UML: (Unified Modeling Language) El lenguaje para modelado unificado (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema. Fue originalmente concebido por la Corporación Rational Software. El lenguaje ha ganado un significante soporte en la industria de varias organizaciones. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software antes de escribir cualquier línea de código.

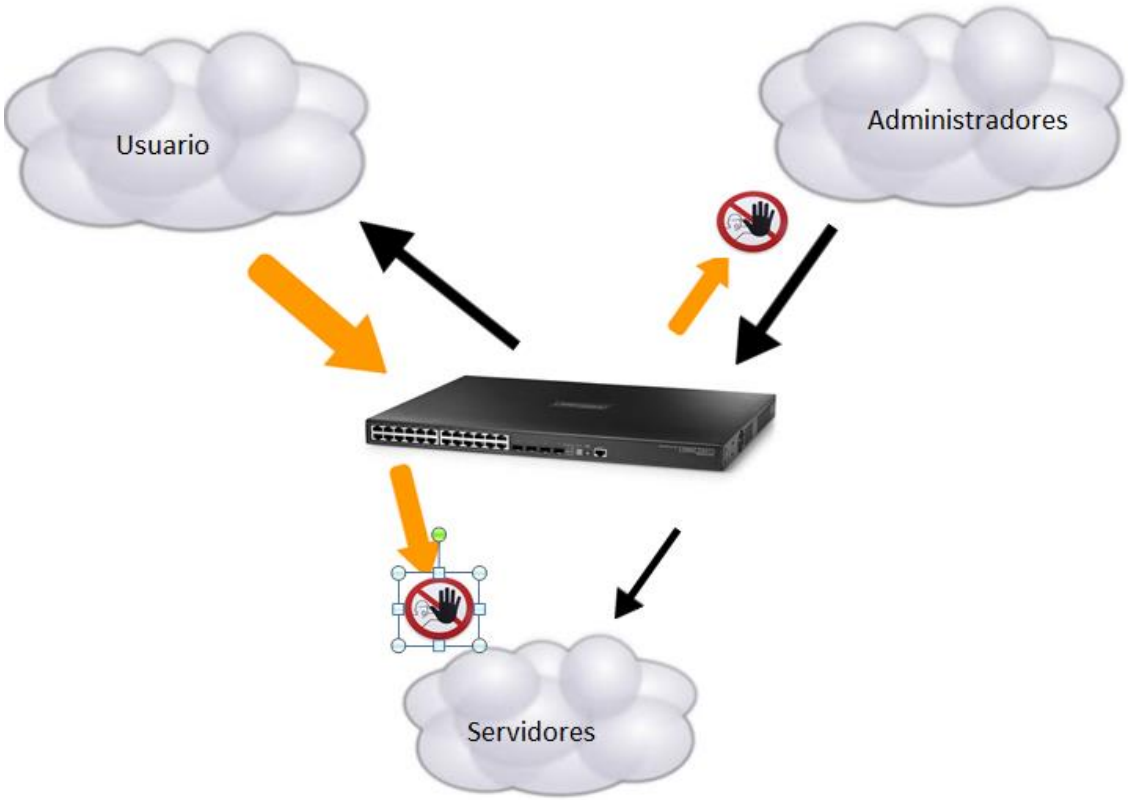
URL: Localizador uniforme de recurso, es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización. El URL de un recurso de información es su dirección en Internet, la cual permite que el navegador la encuentre y la muestre de forma adecuada.

Anexos

Anexo 1.Escenario#1



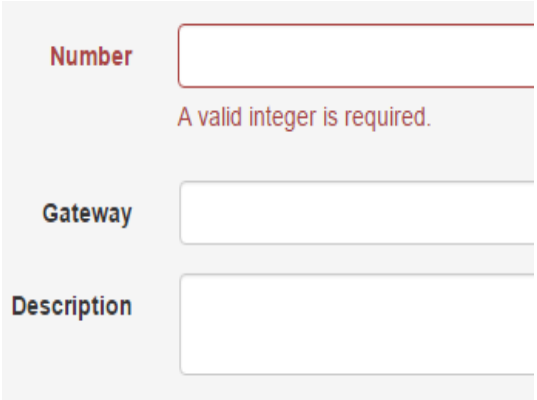
Anexo 2.Escenario#2



Anexo 3.Validación del Caso de uso Autenticarse.

Anexo 4.Validación del Caso de uso Autenticarse.

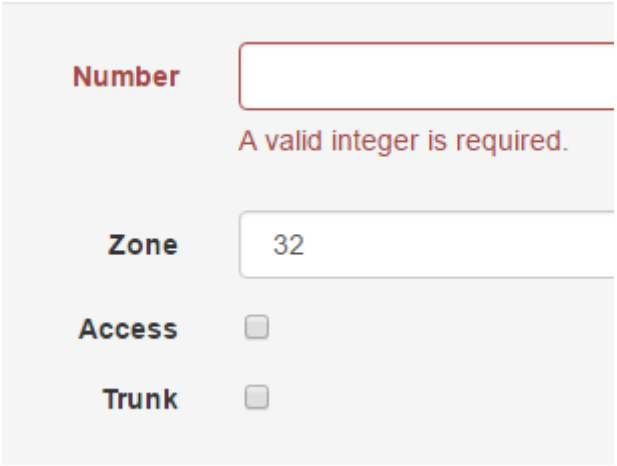
Anexo 5.Validación del Caso de uso Gestionar Zona.



Anexo 6.Validación del Caso de uso Gestionar Zona.


```
{
    "non_field_errors": [
        "Only a subnetwork"
    ]
}
```


Anexo 7.Validación del Caso de uso Gestionar Puerto.

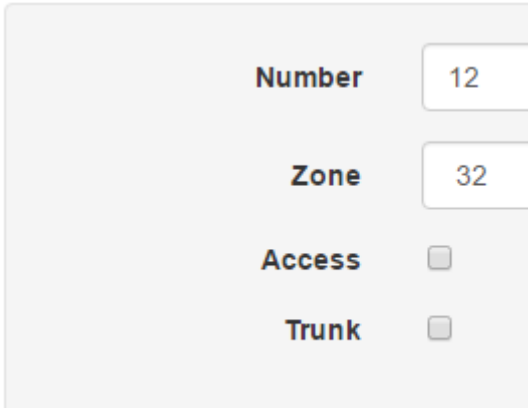


Anexo 8.Validación del Caso de uso Gestionar Puerto.


```
{
    "non_field_errors": [
        "Chose either mode access or mode trunk"
    ]
}
```


Anexo 9.Validación del Caso de uso Gestionar Puerto.

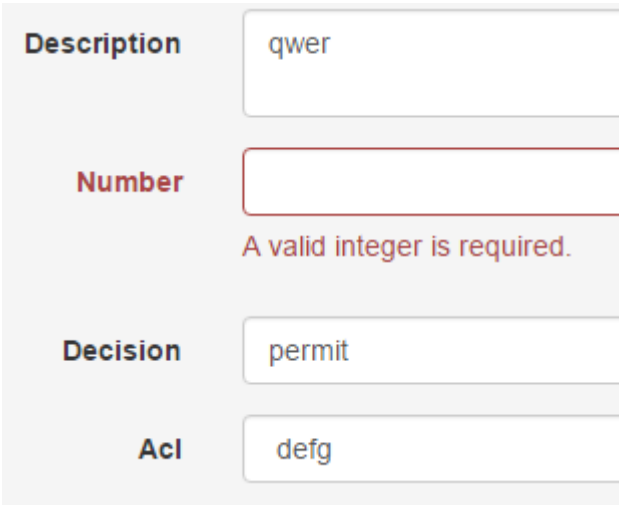

```
{
    "non_field_errors": [
        "Chose a mode"
    ]
}
```

Anexo 10.Validación del Caso de uso Gestionar Objeto.


```
{
    "non_field_errors": [
        "Only a subnetwork or a range or ip can be inserted at a time"
    ]
}
```


Anexo 11.Validación del Caso de uso Gestionar Regla.



Anexo 12.Validación del Caso de uso Gestionar Regla.