**Trabajo Práctico final :**

**Materia: Complejidad Temporal y Estructura de datos**

Proyecto implementación Servidor DNS

Autor: Baez Jonathan

Legajo:9172

Fecha: 02/06/2016

Introduccion:

El presente informe busca describir la implementación de un software diseñado objetivo de simular el comportamiento de un Servidor DNS.

Servidor DNS(Servidor de Nombres de Dominio) es un servidor que almacena referencias a dominios (dominio = host de lado cliente conectado a una red) .

Es un sistema para asignar nombres a equipos y servicios de red que se organiza en una jerarquía de dominios.

https://msdn.microsoft.com/es-es/library/cc787920(v=ws.10).aspx

Explicacion de funcionamiento:

El desarrollo del mismo consiste en la implementación del ArbolGeneral padre (Servidor DNS). Dicho árbol posee en cada una de sus ramas los nodos (Subárboles generales) que representan a cada una de las partes de un dominio completo como por ejemplo: [www.google.com](http://www.google.com).

Para poder llevarse a cabo dicha estructura se tuvo que dividir la url por puntos. Es decir, por cada punto la parte a la izquierda es un nodo. Para este ejemplo contamos con tres nodos: www, google y com.

La parte más a la derecha de dicho dominio es el Subdominio de Nivel Superior, del cual desprende el resto de los subdominios representados por cada una de las partes más a la izquierda.

Cada nodo puede contener más de dos hijos (no es un Árbol Binario) por el hecho de que un mismo subdominio puede contener diversos subdominios hijos.

Datos de entrada:

Para la inserción de dominios en el Servidor de traducciones, se ingresa como String (cadena de texto o texto normal) mediante la consola (utilizando el Scanner de java) y luego en cada método respectivo se les hace una división por puntos a la url que representa al dominio. Dicha división se hace utilizando el split de Java. Todos los datos se ingresan como String, sin embargo, al momento de utilizarlos se los parsea (se los transforma a su representación necesaria). Si este parseo no se puede realizar se lanza una excepción.

Por ejemplo, en la búsqueda de nodos que se encuentran a una determinada profundidad es necesario ingresar un número (se ingresa como String y se parsea a entero).

Las estructuras utilizadas fueron:

Arbol General

Nodo General

ListaEnlazada

Nodo

ListaConArreglo

Recorredor

Pila

Cola

Las clases utilizadas fueron:

ArbolGeneralDNS :núcleo de las funcionalidades del DNS, adaptación de Árbol General

NodoGeneral: utilizada para la raíz de cada Árbol. No tuvo modificación con respecto a la definición de Nodo General

ListaEnlazada:utilizada para representar la lista de hijos del Árbol, Se sobrescribió el método toString para mostrar los elementos.

Nodo:representa cada elemento de una lista enlazada (cada hijo). Se sobrescribió también el método toString para mostrar el dato contenido.

ListaConArreglo: reemplaza los arreglos en java, ej: String[], en este desarrollo. Se sobrescribió el método toString para mostrar los elementos contenidos.

Recorredor: utilizada para recorrer cada elemento de una lista.

ModAdmin: utilizada para gestionar las operaciones de tipo Administración delegadas por el Árbol General.

ModCons: utilizada para gestionar las operaciones de tipo Consulta delegadas por el Árbol General.

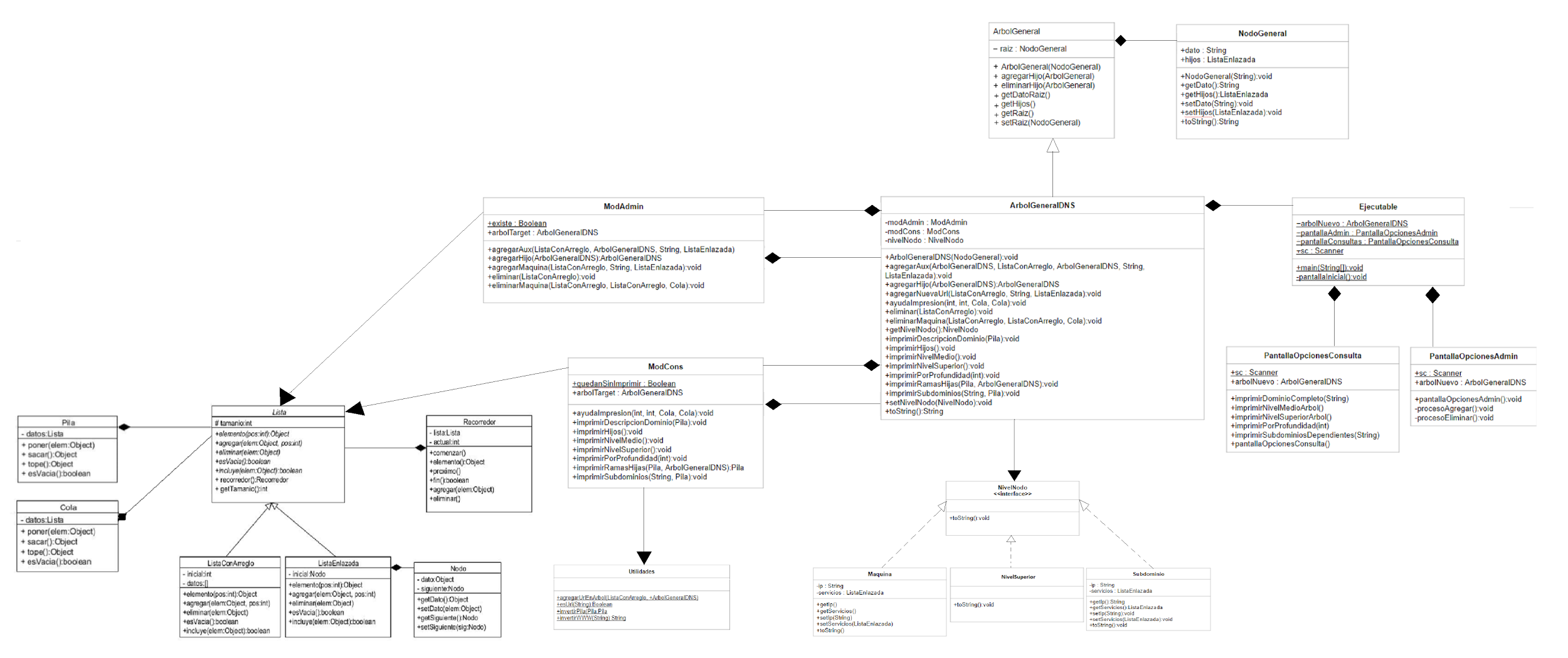
PantallaOpcionesAdmin: clase creada para la interacción con el usuario en la pantalla de administración.

PantallaOpcionesConsulta: clase creada para la interacción con el usuario en la pantalla de consultas.

Utilidades: utilizada para operaciones genéricas como : invertir una estructura de datos, validar que sea un url el dato ingresado.

Ejecutable: clase principal para la ejecución de la aplicación. (De la cual derivan el resto de las clases mencionadas anteriormente).

Diagrama UML (Diagrama de jerarquía de clases):



Se utilizaron los recorridos :

Módulo de consultas:

PostOrden (Hijos después raíz):

imprimirRamasHijas

imprimirSubdominoDependiente

imprimirDescripcionDominio

Por niveles:

imprimirNivelSuperior

imprimirPorProfundidad

Módulo administración:

Preorden:

agregarHijo

eliminarMaquina

Algunos de los problemas encontrados:

Muchos problemas de compartición de variables y métodos entre subclases y clases padres. (Polimorfismo)

Necesidad de utilizar métodos auxiliares para realizar las recurrencias que necesitaban mantener un dato estático (para no modificar la clase contenedora con muchos atributos)

No se encontró manera de “limpiar ” la consola de java, por este motivo se realizó un for con 100 iteraciones imprimiendo espacios en blanco

En la consola de java original, existen problemas con el encoding (diferente al eclipse)

Muchos problema con los índices y las iteraciones (hubo casos en que buscaba elementos null, y otros casos en los que se buscaba fuera del rango en el Recorredor por ejemplo). Se fueron solventando a medida que iba avanzando el desarrollo.

Fue muy dificil la creación del diagrama UML, debido a que no se cuenta con una herramienta especifica gratuita y que sea facil integrar con el documento sin necesidad de utilizar herramientas externas para la interpretacion del mismo (por ejemplo:Altova MissionKit que genera el uml en formato .XML ), o herramientas que solicitan Suscripcion y no soy de uso libre y gratuito,como por ejemplo: https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693.

Sin embargo logré crear el UML a base de Paint y la aplicación de dibujado de Google, copiando desde el Eclipse los textos.

Requisitos para el correcto funcionamiento del software mencionado:

Este software funcionará correctamente en plataformas Windows, Linux (distribuciones) o IOS tales que contengan las últimas actualizaciones de la plataforma Java (JVM). Debería poder funcionar en cualquier Sistema Operativo que provea la plataforma Java instalada.

Faltaría implementar para un futuro desarrollo:

* Solucionado de problemas de encoding para los caracteres especiales en la consola de Java: dependiendo del estándar de traducción de texto que esté definido en las configuraciones de la consola, podrá traducir determinados caracteres (Algunos posibles estándares: utf-8,ISO-8859-1).
* Validación de formato correcto para una IP ingresada: cuatro segmentos (hasta el momento) donde cada uno de ellos es un entero menor a 255 y con un separador definido por un punto (.) por cada segmento.
* Validación de servicios válidos definidos hasta el momento en los estándares, como por ejemplo: FTP, P2P.
* Utilización de una Base de Datos para almacenar los datos a la cual cada nodo del Árbol hará referencia al momento de las consultas.
* Interfaz gráfica para una mejor interacción entre el lado cliente y la parte lógica (Backend o parte sin interacción usuaria), apuntando a una mejor experiencia de usuario.
* Copia de seguridad del Árbol General la cual se ve relacionada a un nuevo Árbol apuntando a la misma BD.

Aclaraciones:

La búsqueda de nodos dependientes se realizó buscando por nodo subdominio e imprimiendo todo los dominios dependientes completos. El enunciado fue entendido de dicha manera, sin embargo, en la primera exposición del trabajo en desarrollo se llegó a la conclusión de que a pesar de que el enunciado se comprendió de forma errónea, fue más compleja la solución que se halló y por este motivo el profesor permitió que en la aplicación final se tenga un resultado diferente al esperado por el enunciado.

El enunciado refería a la búsqueda de un subdominio y a partir de este imprimir solo los hijos directos.