Tytus X

Manual Técnico

Proyecto 1

José Andres Rodas Arrecis

201504220

INTRODUCCIÓN:

TytusX es un administrador de bases de datos documental de código abierto desarrollado bajo licencia MIT que utilizó el lenguaje JavaScript y TypeScript para su construcción. Soporta archivos XML y maneja los lenguajes de consultas XPath y XQuery.

El proyecto consta de dos fases: en la primera, se debe construyó un parser de XML y un parser e intérprete de XPath; en la segunda, se debe agregar el parser e intérprete de XQuery, en ambas fases se debe utilizar GitHub Pages para mostrar el funcionamiento de la aplicación Web.

Adicionalmente, en la segunda fase se debe traducir todo el proyecto en código de tres direcciones, este debe cumplir con las reglas definidas en el libro de texto y basarse en la sintaxis del lenguaje C, ya que se utilizará el compilador de C para ejecutar dicha traducción.

ESTRUCTURA DEL PROYECTO:

La estructura de archivos y carpetas que se utilizó en la realización del proyecto TytusX es la siguiente:

```
.
`-- team#/
    |-- analyzers/
    |   |-- xml/
    |   |   |-- expresiones
    |   |   |-- interfaces
    |   |   `-- etc
    |   `-- xpath/
    |       |-- expresiones
    |       |-- interfaces
    |       `-- etc
    |-- css/
    |   `-- style.css
    |-- js/
    |   |-- vis.js
    |   `-- script.js
    |-- index.html
    |-- graph.html
    `-- README.MD
```

Consta de una carpeta principal con el número de grupo, en donde se encuentra la página principal index.html junto con las carpetas que contienen el código realizado.

INDEX.HTML:

El archivo principal del proyecto es el index.html y se encuentra en la ruta inicial del grupo del proyecto. Para poder iniciar el programa de TytusX se debe de abrir la página index. En ella están cargados todos los demás archivos de código que se utilizaran (javascript), así como los archivos de estilos y toda la interfaz del proyecto en sí.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- ============================== MATERIALIZE ============================== -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">

    <!-- ============================== MATERIAL ICONS ============================== -->
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

    <!-- ============================== DATA TABLE ============================== -->
    <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.25/css/jquery.dataTables.css">

    <!-- ============================== CODE MIRROR ============================== -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.58.1/codemirror.min.css" integri

    <!-- ============================== CODE MIRROR THEME ============================== -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/codemirror@5.61.0/theme/material-ocean.css">


    <!-- ============================== MY STYLE ============================== -->
    <link rel="stylesheet" href="css/style.css">

    <title>Tytus X</title>
</head>
```

Para la interfaz gráfica del proyecto y algunas otras funcionalidades se utilizó CodeMirror.

MAIN.JS:

Este es el archivo principal donde se guarda toda la lógica de TytusX y donde se conectan las funciones y demas archivos de JavaScript, por ejemplo aquí se encuentra la función que ejecuta el parser de el lenguaje XML asi como la función que genera el parser de XPath.

```
119  /**
120   * ANALIZADOR DESCENDENTE XML
121   */
122  const analysDescXml = () => {
123
124    console.log('Analizador Descendente')
125    var texto = xmlEditor.getValue();
126
127    //creacion del arbol
128    const raiz_arbol = gramdesc.parse(texto);
129    GlobalTree = raiz_arbol;
130
131    //creacion de la tabla de simbolos
132    const tabla_simbolos = new Tabla_Simbolos(CrearTabla(raiz_arbol.objeto));
133    GlobalSymbolTable = tabla_simbolos;
134
135    //generacion del arbol
136    dotStringCst = CST_XML(raiz_arbol.objeto);
137
138    //generacion reporte tabla de simbolos
139    const TablaSimbolos = ReporteTabla(tabla_simbolos.simbolos, 'Global', []);
140
141    //cargando datos a tabla de simbolos
142    symbolTableXml.destroy();
143    symbolTableXml = newDataTable('#symbolTableXml',
144      [{ data: "Idientifier" }, { data: "Type" }, { data: "Value" }, { data: "Row" },
145      TablaSimbolos);
146
```

En el main.js se llaman las funciones y clases necesarias asi como se inicializan las variables globales que seran necesarias.

```
/**
 * ANALIZADOR ASCENDENTE XPATH
 */
const analysAscXpath = () => {

  console.log('Analizador XPath Ascendente')
  var consol_out = ''
  var query_out = ''
  var path = xpathEditor.getValue();

  //Analizador XPath
  const xpath = gram_xpath_asc.parse(path);
  if (xpath) consol_out = 'LA CONSULTA XPATH ES VALIDA'
  else consol_out = 'LA CONSULTA XPATH CONTIENE ERRORES'


  //test consulta
  let entrada = path.split('/')
  entrada.splice(0, 1);
  const consulta = new Consulta(entrada, 1, 1);
  query_out = consulta.ejecutar(GlobalSymbolTable.simbolos, {});

  //encoding
  query_out = Encoding(query_out);

  //seteando consola
  consoleResult.setValue(consol_out + '\n' + query_out);
```

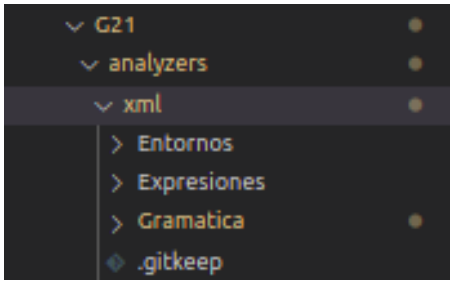
```
/**
 * Necessary Functions
 */
function CrearTabla(objeto) {
  //validando si existe el nodo
  if (objeto != undefined) {
    //definiendo valores
    let id = objeto.identificador;
    let tipo = 'etiqueta'
    let valor = objeto.texto;
    let linea = objeto.linea;
    let columna = objeto.columna;
    //creando el simbolo (entorno)
    let entorno = new Simbolo(id, tipo, valor, linea, columna, [])

    //creando entornos hijos
    if (objeto.listaAtributos != undefined) {
      objeto.listaAtributos.forEach(atr => {
        //definiendo valores
        let id = atr.identificador;
        let tipo = 'atributo'
        let valor = atr.valor;
```

ANALYZERS:

Aquí se encuentra el funcionamiento interno de los analizadores que se implementaron en el proyecto de TytusX. Esta carpeta se divide en dos mas las cuales son XML y XPath.

XML:

En esta carpeta estan todas las clases y funciones ideales para que funcione el analizador de archivos xml, principalmente contiene tres carpetas, las cuales son: Entorno, Expresiones y Gramatica.



ENTORNOS:

En la carpeta de entornos tenemos tres archivos, los cuales son Raiz.js, Simbolo.js y Tabla_Simbolos.js los cuales son utilizados para la creacion de la tabla de simbolos y el arbol generado al momento de analizar el archivo XML de entrada.

```
"use strict";
// Object.defineProperty(exports, "__esModule", { v
// exports.Raiz = void 0;
class Raiz {
    constructor(objeto, config, errores, cst) {
        this.objeto = objeto;
        this.config = config;
        this.errores = errores;
        this.cst = cst;
    }
}
// exports.Raiz = Raiz;

```

```
class Simbolo {
    constructor(id, tipo, valor, linea, columna, entorno) {
        this.id = id;
        this.tipo = tipo;
        this.valor = valor;
        this.linea = linea;
        this.columna = columna;
        this.entorno = entorno;
    }
    existe(id) {
        for (let ent of this.entorno) {
            if (id == ent.id)
                return true;
        }
        return false;
    }
    getSimbolo(id) {
        let simboloTemp = [];
        for (let ent of this.entorno) {
            if (id == ent.id) {
                simboloTemp.push(ent);
            }
        }
        return simboloTemp;
    }
```

```
class Tabla_Simbolos {
    constructor(simbolos) {
        this.simbolos = simbolos;
    }
}
```

GRAMATICA:

En la carpeta de gramatica encontramos los archivos .jison necesarios para que funcione el analizador de XML, para la implementación de dicha gramatica se utilizó la herramienta Jison, la cual emplea un analizador ascendente en base a reglas gramaticales pre definidas en base a sintaxis de la misma herramienta.

```
START : RAIZ EOF          { $$ = new Raiz($1,encoding, errores );
                            errores = [];
                            encoding = [];
                            return $$;}
    ;

RAIZ : ENC OBJETO         { $$ = $2 }
    | OBJETO              { $$ = $1 }
    | error               {
                            errores.push({'Error Type': 'Sintactico', 'Row': @1.first_line, 'Column': @1.first_column, 'D
                            $$ = new Raiz({},encoding, errores );
                            errores = [];
                            encoding = [];
                            return $$;
                           }
;

ENC : menor quest xml LATRIBUTOS quest mayor { encoding = $4; }
;

OBJETO : menor identificador LATRIBUTOS mayor OBJETOS menor div identificador mayor       { $$ = new Objeto($2,'',@1.first
                                                                                              if($2 != $8) {
                                                                                                  //errores.push({error
                                                                                                  errores.push({'Error
                                                                                              };
                                                                                          }
    | menor identificador LATRIBUTOS mayor LISTA_ID_OBJETO   menor div identificador mayor  { $$ = new Objeto($2,$5,@1.fi
                                                                                              if($2 != $8) {
                                                                                                  //errores.push({error
                                                                                                  errores.push({'Error
                                                                                              };
```