Javier Isaías Coyoy Marín **202030556**

Manual Técnico

Versión de Android Studios

2023.2.1

Versión de Koltlin

1.9.0

Gramática

Esta gramática es utilizada para analizar cada una de las estructuras con las cuales nosotros deseamos construir nuestras páginas, a continuación, detallamos más el funcionamiento de cada una de las partes:

La primera parte es el analizador léxico el cual se encarga de obtener los tokens de la cadena de texto que reciba, y para su implementación necesitamos definir primero el paquete al que va pertenecer, la importación del java_cup, y luego vienen algunas características como la declaración de la línea, la columna, el cup para conectar con el analizador sintáctico, luego definimos nuestras expresiones regulares y por ultimo nuestros tokens que vamos a retornar.

```
package com.example.graficasapp.analizadores;

import java_cup.runtime.*;

%% // Separador de Area

%class LexicoKoltlin
%public
%cup
%line
%column
%eofval{
    return symbol(sym.EOF);
%eofval}

%{
    private Symbol symbol(int type) {
        return new Symbol(type, yyline+1, yycolumn+1);
```

```
    }

    private Symbol symbol(int type, Object value) {
        return new Symbol(type, yyline+1, yycolumn+1, value);
    }
%}

Saltos = [\r|\n|\r\n]
Espacios = {Saltos} | [ \t\f]
Numero = [0-9]+
Pixel = \"[0-9]+[pP][xX]\"
Color = \"[#][[a-z]|[0-9]]+\"
Tipo = \"[A-Z]\"
Palabra = \"[^\"]*\"
Identificador = [iI][dD][a-zA-Z]*

%% // Separador de Area

<YYINITIAL> {

    "\"title\""             { return symbol(sym.TITULO, yytext()); }

    "\"description\""       { return symbol(sym.DESCRIPCION, yytext()); }

    "\"keywords\""          { return symbol(sym.PALABRA_CLAVE, yytext()); }

    "\"header\""            { return symbol(sym.ENCABEZADO, yytext()); }

    "\"footer\""            { return symbol(sym.PIE, yytext()); }

    "\"copyright\""         { return symbol(sym.COPYRIGHT, yytext()); }

    "\"backgroundColor\""   { return symbol(sym.FONDO, yytext()); }

    "\"fontFamily\""        { return symbol(sym.FUENTE, yytext()); }

    "\"fontSize\""          { return symbol(sym.TAMANIO_FUENTE, yytext()); }

    "\"data\""              { return symbol(sym.DATA, yytext()); }

    "\"category\""          { return symbol(sym.CATEGORIA, yytext()); }

    "\"value\""             { return symbol(sym.VALOR, yytext()); }

    "\"color\""             { return symbol(sym.COLOR, yytext()); }
```

```
"\"chart\""            { return symbol(sym.CHART, yytext()); }
"\"xAxisLabel\""       { return symbol(sym.X_BARRA, yytext()); }
"\"yAxisLabel\""       { return symbol(sym.Y_BARRA, yytext()); }
"\"label\""            { return symbol(sym.LABEL, yytext()); }
"\"legendPosition\""   { return symbol(sym.POSICION, yytext()); }
"\"x\""                { return symbol(sym.X, yytext()); }
"\"y\""                { return symbol(sym.Y, yytext()); }
"\"size\""             { return symbol(sym.TAMANIO, yytext()); }
"\"name\""             { return symbol(sym.NOMBRE, yytext()); }
"\"points\""           { return symbol(sym.PUNTOS, yytext()); }
"\"lineStyle\""        { return symbol(sym.LINEA_ESTILO, yytext()); }
"{"                    { return symbol(sym.LLAVE_I, yytext()); }
"}"                    { return symbol(sym.LLAVE_D, yytext()); }
"["                    { return symbol(sym.CORCHETE_I, yytext()); }
"]"                    { return symbol(sym.CORCHETE_D, yytext()); }
":"                    { return symbol(sym.DOS_PUNTOS, yytext()); }
";"                    { return symbol(sym.PUNTO_Y_COMA, yytext()); }
","                    { return symbol(sym.COMA, yytext()); }
"+"                    { return symbol(sym.MAS, yytext()); }
"if"                   { return symbol(sym.IF, yytext()); }
"else"                 { return symbol(sym.ELSE, yytext()); }
"for"                  { return symbol(sym.FOR, yytext()); }
```

```
    "while"                 { return symbol(sym.WHILE, yytext()); }

    "="                     { return symbol(sym.IGUAL, yytext()); }

    "=="                    { return symbol(sym.DOBLE_IGUAL, yytext()); }

    "!="                    { return symbol(sym.NO_IGUAL, yytext()); }

    ">"                     { return symbol(sym.MAYOR, yytext()); }

    ">="                    { return symbol(sym.MAYOR_IGUAL, yytext()); }

    "<"                     { return symbol(sym.MENOR, yytext()); }

    "<="                    { return symbol(sym.MENOR_IGUAL, yytext()); }

    "("                     { return symbol(sym.PARENTESIS_I, yytext()); }

    ")"                     { return symbol(sym.PARENTESIS_D, yytext()); }

    {Numero}                { return symbol(sym.NUMERO, yytext()); }

    {Pixel}                 { return symbol(sym.PIXEL, yytext()); }

    {Color}                 { return symbol(sym.NUMERO_COLOR, yytext()); }

    {Tipo}                  { return symbol(sym.TIPO, yytext()); }

    {Identificador}         { return symbol(sym.IDENTIFICADOR, yytext()); }

    {Palabra}               { return symbol(sym.PALABRA, yytext()); }

    {Espacios}              {/* ignoramos */}

}

[^]                         { throw new Error("Error Léxico caracter Invalido en
la linea " + (yyline+1) + ", columna " + (yycolumn+1) + ": " + yytext()); }
```

La segunda parte de nuestra gramatica es el analizador sintáctico que consta de las declaración de producciones con las cuales vamos a ir componiendo nuestros tokens para organizarlos de manera que se acomoden en el orden que nosotros estamos esperando. Además de que buscamos analizamos los posibles errores que podamos encontrar y tratamos de controlarlos.

```
package com.example.graficasapp.analizadores;

import java_cup.runtime.*;

parser code {:

    private Symbol symbol;
    private String contenido;

    public SintacticoKoltlin(LexicoKoltlin lexico) {
        super(lexico);
    }

    public void syntax_error(Symbol cur_token) {
        symbol = cur_token;
        contenido = (String)(cur_token.value);
    }

    public Symbol getSymbol() {
        return this.symbol;
    }

    public String getContenido() {
        return this.contenido;
    }
:}

/* Terminales */
terminal            LLAVE_I, LLAVE_D, CORCHETE_I, CORCHETE_D, DOS_PUNTOS, COMA,
DATA, CHART,
                    TITULO, DESCRIPCION, PALABRA_CLAVE, ENCABEZADO, PIE,
COPYRIGHT, FONDO, FUENTE, TAMANIO_FUENTE, X_BARRA, Y_BARRA,
                    CATEGORIA, LABEL, TAMANIO, X, Y, VALOR, COLOR, NOMBRE,
PUNTOS, POSICION, LINEA_ESTILO,
                    IF, ELSE, FOR, WHILE, PARENTESIS_I, PARENTESIS_D,
DOBLE_IGUAL, NO_IGUAL, IGUAL, MAYOR, MAYOR_IGUAL, MENOR, MENOR_IGUAL, MAS,
PUNTO_Y_COMA;

terminal String     PIXEL, NUMERO_COLOR, TIPO, PALABRA, IDENTIFICADOR;
terminal int        NUMERO;

/* No Terminales */
non terminal        inicio, claves, iniciodata, interior1, interior2,
                    simplebarra, expandidobarra,
                    simplepastel, expandidopastel, extra2,
```

```
                    simplepunto, expandidopunto, extra1,
                    simplelinea, expandidolinea, extra3,
                    complemento, identificador;

/* Gramatica */
start with inicio;

/* ---------------------------------- Comienzo ------------------------------
---- */
inicio ::= LLAVE_I TITULO DOS_PUNTOS PALABRA COMA
        DESCRIPCION DOS_PUNTOS PALABRA COMA
        PALABRA_CLAVE DOS_PUNTOS CORCHETE_I claves CORCHETE_D COMA
        ENCABEZADO DOS_PUNTOS LLAVE_I TITULO DOS_PUNTOS PALABRA LLAVE_D COMA
        PIE DOS_PUNTOS LLAVE_I COPYRIGHT DOS_PUNTOS PALABRA LLAVE_D COMA
        FONDO DOS_PUNTOS NUMERO_COLOR COMA
        FUENTE DOS_PUNTOS PALABRA COMA
        TAMANIO_FUENTE DOS_PUNTOS PIXEL iniciodata LLAVE_D
        ;

claves ::= PALABRA COMA claves
        | PALABRA
        ;

iniciodata ::= LLAVE_I DATA DOS_PUNTOS CORCHETE_I interior1 LLAVE_D iniciodata
        | IF PARENTESIS_I complemento PARENTESIS_D LLAVE_I iniciodata LLAVE_D
iniciodata
        | IF PARENTESIS_I complemento PARENTESIS_D LLAVE_I iniciodata LLAVE_D
        | IF PARENTESIS_I complemento PARENTESIS_D LLAVE_I iniciodata LLAVE_D
ELSE LLAVE_I iniciodata LLAVE_D iniciodata
        | IF PARENTESIS_I complemento PARENTESIS_D LLAVE_I iniciodata LLAVE_D
ELSE LLAVE_I iniciodata LLAVE_D
        | FOR PARENTESIS_I complemento PUNTO_Y_COMA complemento PUNTO_Y_COMA
IDENTIFICADOR MAS MAS PARENTESIS_D LLAVE_I iniciodata LLAVE_D iniciodata
        | FOR PARENTESIS_I complemento PUNTO_Y_COMA complemento PUNTO_Y_COMA
IDENTIFICADOR MAS MAS PARENTESIS_D LLAVE_I iniciodata LLAVE_D
        | WHILE PARENTESIS_I complemento PARENTESIS_D LLAVE_I iniciodata
IDENTIFICADOR MAS MAS LLAVE_D iniciodata
        | WHILE PARENTESIS_I complemento PARENTESIS_D LLAVE_I iniciodata
IDENTIFICADOR MAS MAS LLAVE_D
        | LLAVE_I DATA DOS_PUNTOS CORCHETE_I interior1 LLAVE_D
        ;

complemento ::= identificador IGUAL identificador
        | identificador DOBLE_IGUAL identificador
        | identificador NO_IGUAL identificador
```

```
        | identificador MAYOR identificador
        | identificador MENOR identificador
        | identificador MAYOR_IGUAL identificador
        | identificador MENOR_IGUAL identificador
        ;

identificador ::= IDENTIFICADOR
        | NUMERO
        ;

interior1 ::= LLAVE_I CATEGORIA DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO
LLAVE_D COMA simplebarra
        | LLAVE_I CATEGORIA DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO LLAVE_D
CORCHETE_D
        | LLAVE_I CATEGORIA DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO COMA
COLOR DOS_PUNTOS NUMERO_COLOR LLAVE_D COMA expandidobarra
        | LLAVE_I CATEGORIA DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO COMA
COLOR DOS_PUNTOS NUMERO_COLOR LLAVE_D CORCHETE_D COMA extra1
        | LLAVE_I LABEL DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO LLAVE_D COMA
simplepastel
        | LLAVE_I LABEL DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO LLAVE_D
CORCHETE_D
        | LLAVE_I LABEL DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO COMA COLOR
DOS_PUNTOS NUMERO_COLOR LLAVE_D COMA expandidopastel
        | LLAVE_I LABEL DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO COMA COLOR
DOS_PUNTOS NUMERO_COLOR LLAVE_D CORCHETE_D COMA extra2
        | LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO LLAVE_D COMA
simplepunto
        | LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO LLAVE_D CORCHETE_D
        | LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO COMA TAMANIO
DOS_PUNTOS NUMERO COMA COLOR DOS_PUNTOS NUMERO_COLOR LLAVE_D COMA expandidopunto
        | LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO COMA TAMANIO
DOS_PUNTOS NUMERO COMA COLOR DOS_PUNTOS NUMERO_COLOR LLAVE_D COMA extra1
        | LLAVE_I NOMBRE interior2
        ;

interior2 ::= DOS_PUNTOS PALABRA COMA PUNTOS DOS_PUNTOS CORCHETE_I simplelinea
CORCHETE_D LLAVE_D COMA LLAVE_I NOMBRE interior2
    | DOS_PUNTOS PALABRA COMA PUNTOS DOS_PUNTOS CORCHETE_I simplelinea CORCHETE_D
LLAVE_D CORCHETE_D COMA extra1
        | DOS_PUNTOS PALABRA COMA PUNTOS DOS_PUNTOS CORCHETE_I expandidolinea
CORCHETE_D COMA extra3 LLAVE_D COMA LLAVE_I NOMBRE interior2
        | DOS_PUNTOS PALABRA COMA PUNTOS DOS_PUNTOS CORCHETE_I expandidolinea
CORCHETE_D COMA extra3 LLAVE_D CORCHETE_D COMA extra1
        ;
```

```
/* ---------------------------------- Barras ----------------------------------
-- */
simplebarra ::= LLAVE_I CATEGORIA DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO
LLAVE_D COMA simplebarra
        | LLAVE_I CATEGORIA DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO LLAVE_D
CORCHETE_D
        ;

expandidobarra ::= LLAVE_I CATEGORIA DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO
COMA COLOR DOS_PUNTOS NUMERO_COLOR LLAVE_D COMA expandidobarra
    | LLAVE_I CATEGORIA DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO COMA COLOR
DOS_PUNTOS NUMERO_COLOR LLAVE_D CORCHETE_D COMA extra1
        ;



/* ---------------------------------- Pastel ----------------------------------
-- */
simplepastel ::= LLAVE_I LABEL DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO
LLAVE_D COMA simplepastel
    | LLAVE_I LABEL DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO LLAVE_D
CORCHETE_D
    ;

expandidopastel ::= LLAVE_I LABEL DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO
COMA COLOR DOS_PUNTOS NUMERO_COLOR LLAVE_D COMA expandidopastel
    | LLAVE_I LABEL DOS_PUNTOS TIPO COMA VALOR DOS_PUNTOS NUMERO COMA COLOR
DOS_PUNTOS NUMERO_COLOR LLAVE_D CORCHETE_D COMA extra2
    ;

/* PENDIENTE DE VER SI HAY ESPECIFICACIONES PARA EL LUGAR=PALABRA (BOTTOM) */
extra2 ::= CHART DOS_PUNTOS LLAVE_I TITULO DOS_PUNTOS PALABRA COMA POSICION
DOS_PUNTOS PALABRA LLAVE_D
    ;




/* ---------------------------------- Puntos ----------------------------------
-- */
simplepunto ::= LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO LLAVE_D COMA
simplepunto
```

```
        |       LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO LLAVE_D
CORCHETE_D
        ;

expandidopunto ::= LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO COMA
TAMANIO DOS_PUNTOS NUMERO COMA COLOR DOS_PUNTOS NUMERO_COLOR LLAVE_D COMA
expandidopunto
        | LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO COMA TAMANIO
DOS_PUNTOS NUMERO COMA COLOR DOS_PUNTOS NUMERO_COLOR LLAVE_D CORCHETE_D COMA
extra1
        ;




/* ---------------------------------- Parte en Común entre el Comienzo, Barras
y Puntos ---------------------------------- */
extra1 ::= CHART DOS_PUNTOS LLAVE_I TITULO DOS_PUNTOS PALABRA COMA X_BARRA
DOS_PUNTOS PALABRA COMA Y_BARRA DOS_PUNTOS PALABRA LLAVE_D
    ;




/* ---------------------------------- Lineas ----------------------------------
-- */
simplelinea ::= LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO LLAVE_D COMA
simplelinea
        | LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO LLAVE_D
        ;

expandidolinea ::= LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO COMA
LABEL DOS_PUNTOS PALABRA LLAVE_D COMA expandidolinea
        | LLAVE_I X DOS_PUNTOS NUMERO COMA Y DOS_PUNTOS NUMERO COMA LABEL
DOS_PUNTOS PALABRA LLAVE_D
        ;

extra3 ::= COLOR DOS_PUNTOS NUMERO_COLOR COMA LINEA_ESTILO DOS_PUNTOS PALABRA
        ;
```