



# PRACTICA NUMERO 1

28.2.2021

—

Carlos Benjamín Pac Flores

Centro Universitario de Occidente (CUNOC)

Ingeniería en Ciencias y Sistemas

Febrero 2022

<b>Introducción</b>	<b>2</b>
<b>Plataforma de Desarrollo:</b>	<b>3</b>
<b>Alcances Tecnicos:</b>	<b>3</b>
<b>Requisitos De Software:</b>	<b>3</b>
<b>Detalle del Sistema:</b>	<b>4</b>
Estructura del analisis Lexico:	4
<b>Estructura del Análisis Sintáctico:</b>	<b>6</b>
Símbolos Terminales:	6
Símbolos No Terminales:	6
Reglas de Producción:	6
Validación de los valores específicos (análisis semántico):	8
<b>Código de Ejemplo:</b>	<b>9</b>
<b>Estructura del Proyecto:</b>	<b>10</b>

## Introducción

El siguiente documento da presenta una descripción de la estructura del software de la aplicación desarrollada para el entorno de Android, que por medio de una serie de un lenguaje formal el software es capaz de graficar los gráficos de Barras y de Pie respectivamente, el procesado del código es por medio de un analizador lexico y sintactico y por medio de este último poder resolver parámetros basados en operaciones entre números tales como sumas, restas, multiplicaciones y divisiones y agrupaciones.

## Plataforma de Desarrollo:

- Sistema de gestión de proyecto Grandle.
- Android Studio
- Librería para análisis léxico JFLEX
- Librería para análisis sintáctico CUP
- Lenguaje Kotlin y Java

## Alcances Tecnicos:

Poder visualizar Gráficas de Pie o de Barras por medio de un lenguaje formal y con ayuda de la librería MP Android Chart en su versión 3.1.0 ayuda a visualizar las gráficas ya que ofrece una configuración personalizable, y ser amigable con el usuario.

## Requisitos De Software:

- Android en su version minima 4.2
- Ram recomendada 3 GB

## Detalle del Sistema:

### Estructura del analisis Lexico:

Para el funcionamiento del analizador léxico se establecen las siguientes expresiones regulares para el reconocimiento de los TOKENS del texto.

- **LineTerminator** = `[\r|\n|\r\n]+`  
Reconoce los fines de línea en el texto, como retornos de carro y nuevas líneas.
- **WhiteSpace** = `[\t\n]+`  
Reconoce los espacios entre textos, como los ya mencionados espacios y las tabulaciones.
- **Comment** = `#[^\n]*`  
Reconoce los comentarios incluidos en el código y su alcance máximo es una línea
- **Def** = `def|Def`  
Reconoce la instrucción para la definición de una gráfica.
- **GraphicBarras** = `Barras`  
Reconoce la instrucción para la definición de una gráfica de Barras.
- **GraphicPie** = `Pie`  
Reconoce la instrucción para la definición de una gráfica de Pie.
- **Tittle** = `titulo`  
Reconoce el parámetro título de una gráfica tanto de barras o de pie.
- **EjeX** = `ejex`  
Reconoce el parámetro ejex de una gráfica de barras.
- **EjeY** = `ejey`  
Reconoce el parámetro ejey de una gráfica de barras.
- **Label** = `etiquetas`  
Reconoce el parámetro etiqueta de una gráfica de pie.
- **Values** = `valores`

Reconoce el parámetro valores de una gráfica de pie.

- **Link = unir**

Reconoce el parámetro unir de una gráfica tanto de barras o de pie.

- **Type = tipo**

Reconoce el parámetro tipo de una gráfica de pie.

- **TypeValue = Porcentaje | Cantidad**

Reconoce el valor del parámetro tipo de una gráfica de pie.

- **Total = total**

Reconoce el parámetro total de una gráfica de pie.

- **Extra = extra**

Reconoce el parámetro extra de una gráfica de pie.

- **Execute = Ejecutar**

Reconoce la instrucción ejecutar del lenguaje.

- **Numbers = [0-9]+**

Reconoce un valor entero escrito en el lenguaje.

- **Decimal = [0-9]+.[0-9]+**

Reconoce un valor decimal escrito en el lenguaje.

- **[""](^\\n\\r\\")+[""]**

Reconoce una cadena de datos escrita en el lenguaje.

- **{ + }**

Reconoce el símbolo de suma.

- **{ - }**

Reconoce el símbolo de resta.

- **{ \* }**

Reconoce el símbolo de multiplicación.

- **{ / }**

Reconoce el símbolo de división.

- **{ ( }**

Reconoce el paréntesis de apertura.

- **{ ) }**  
Reconoce el paréntesis de cierre.
- **{ { }**  
Reconoce la llave de apertura.
- **{ } }**  
Reconoce la llave de cierre.
- **{ [ ] }**  
Reconoce el corchete de apertura.
- **{ [ ] }**  
Reconoce el corchete de cierre.
- **{ , }**  
Reconoce el separador de texto denominado COMA.
- **{ ; }**  
Reconoce el fin de una instrucción denominada PUNTO Y COMA.
- **{ : }**  
Reconoce la asignación de un parámetro de una gráfica.

## Estructura del Análisis Sintáctico:

Símbolos Terminales:

**DEF, GRAPHICBARRA, GRAPHICPIE, TITTLE, EJEX, EJEY, LABEL, VALUES, LINK, TYPE, TYPEVALUE, TOTAL, EXTRA, EXECUTE, MAS, MENOS, MUL, DIV, PA\_A, PA\_C, CO\_A, CO\_C, LLA\_A, LLA\_C, COMA, DOSPUNTOS, PUNTOCOMA, STRING, NUMBERS, DECIMAL**

Símbolos No Terminales:

**s , ej, EjeGraf, DefgrafPie, DefgrafBarra, contPie, contBarra, contEjeY, contEjeY2, contEjeX, contEjeX2, contUnir, contUnir2, e ,ee, t, f, te, fe**

Reglas de Producción:

El símbolo no terminal “ej” es donde se inicia la producción de gramática.

Con esta producción se reconoce por medio de recursividad por la izquierda la instrucciones de ejecutar o ninguna de ellas, con el fin de que al momento que sea reconocida se puedan realizar operaciones sobre las mismas.

**ej ::= ej EjeGraf**

| **s**

Con esta producción se reconoce por medio de recursividad por la izquierda la definiciones de gráficas, de igual manera que las ejecuciones se utiliza la recursividad por la izquierda para poder evaluar los resultados de una manera más fácil y en orden de tipado.

**s ::= s DefgrafBarra**

| **s DefgrafPie**

| **DefgrafBarra**

| **DefgrafPie**

Con las siguientes 3 producciones podemos reconocer la estructura principal para la definición de un gráfico.

**EjeGraf ::= EXECUTE PA\_A STRING PA\_C PUNTOCOMA**

**DefgrafBarra ::= DEF GRAPHICBARRA contBarra LLA\_C**

**DefgrafPie ::= DEF GRAPHICPIE contPie LLA\_C**

Con las producciones e,t,f son utilizadas para reconocer y realizar operaciones tales como la suma, resta,multiplicación, división, en números enteros o decimales.

**e ::= e MAS t**

| **e MENOS t**

| **t**

**t ::= t MUL f**

| **t DIV f**

| **f**

**f ::= NUMBERS**

| **DECIMAL**

| **PA\_A e PA\_C**

Con estas dos producciones podemos recuperar los valores numéricos pertenecientes al arreglo de asignación a la propiedad ejey de la gráfica de barras o valores para la gráfica de pie.

**contEjeY ::= CO\_A e contEjeY2**

**contEjeY2 ::= COMA e contEjeY2**



| **CO\_C**

Con estas dos producciones podemos recuperar los valores numéricos pertenecientes al arreglo de asignación a la propiedad eje de la gráfica de barras o etiquetas para la gráfica de pie.

**contEjeX ::= CO\_A STRING contEjeX2**

**contEjeX2 ::= COMA STRING contEjeX2**

| **CO\_C**

Con estas dos producciones podemos recuperar la pareja de valores numéricos enteros utilizados para hacer una referencia de utilización para las propiedades ejex y ejey del gráfico de barras y valores y etiquetas para la gráfica de pie.

**contUnir ::= LLA\_A ee COMA ee LLA\_C contUnir2**

**contUnir2 ::= COMA LLA\_A ee COMA ee LLA\_C contUnir2**

| **CO\_C**

Con las últimas 3 producciones ee,et,fe son utilizadas para reconocer y realizar operaciones tales como la suma, resta, multiplicación, división, en números enteros de exclusividad para el parámetro unir de las gráficas.

**ee ::= ee MAS te**

| **ee MENOS te**

| **te**

**te ::= te MUL fe**

| **te DIV fe**

| **fe**

**fe ::= NUMBERS**

| **PA\_A e PA\_C**

## Validación de los valores específicos (análisis semántico):

El lenguaje al ser restringido por ciertos valores que pueden o no estar se utiliza lógica por medio de clases no ajenas al análisis léxico y sintáctico en lenguaje Java; como el verificar que los valores introducidos en los parámetros de unir sean enteros y que estén en el rango de acceso a los mismos ya que hacen referencia a una casilla de un array de datos,

otro a mencionar es los parámetros de configuración de un gráfica en el tipo de gráfica de pie ya que dependiendo de su valor usamos una parte de los parámetros permitidos y otro no lo son.

## Código de Ejemplo:

#Inicio del archivo de ejemplo

def Barras{

    titulo: "Grafica 1";

    ejex:["item1", "item2"];

    ejey:[5 + (3\*3),1+34-2];

    unir:[{0,1}, {1,0}];

    #Fin de definicion de grafica

}

Def Pie{

    titulo: "Grafica 2";

    tipo: Cantidad;

    etiquetas: ["Compi1", "Compi2"];

    valores:[5, 10];

    total: 25;

    unir:[{0,1}, {1,0}];

    extra: "Resto";

}

Def Pie{

    titulo: "Grafica 3";

    tipo: Porcentaje;

    etiquetas: ["Compi1","Compi2"];

    valores:[70,120];

    unir:[{0,1},{1,0}];

    extra: "Resto";

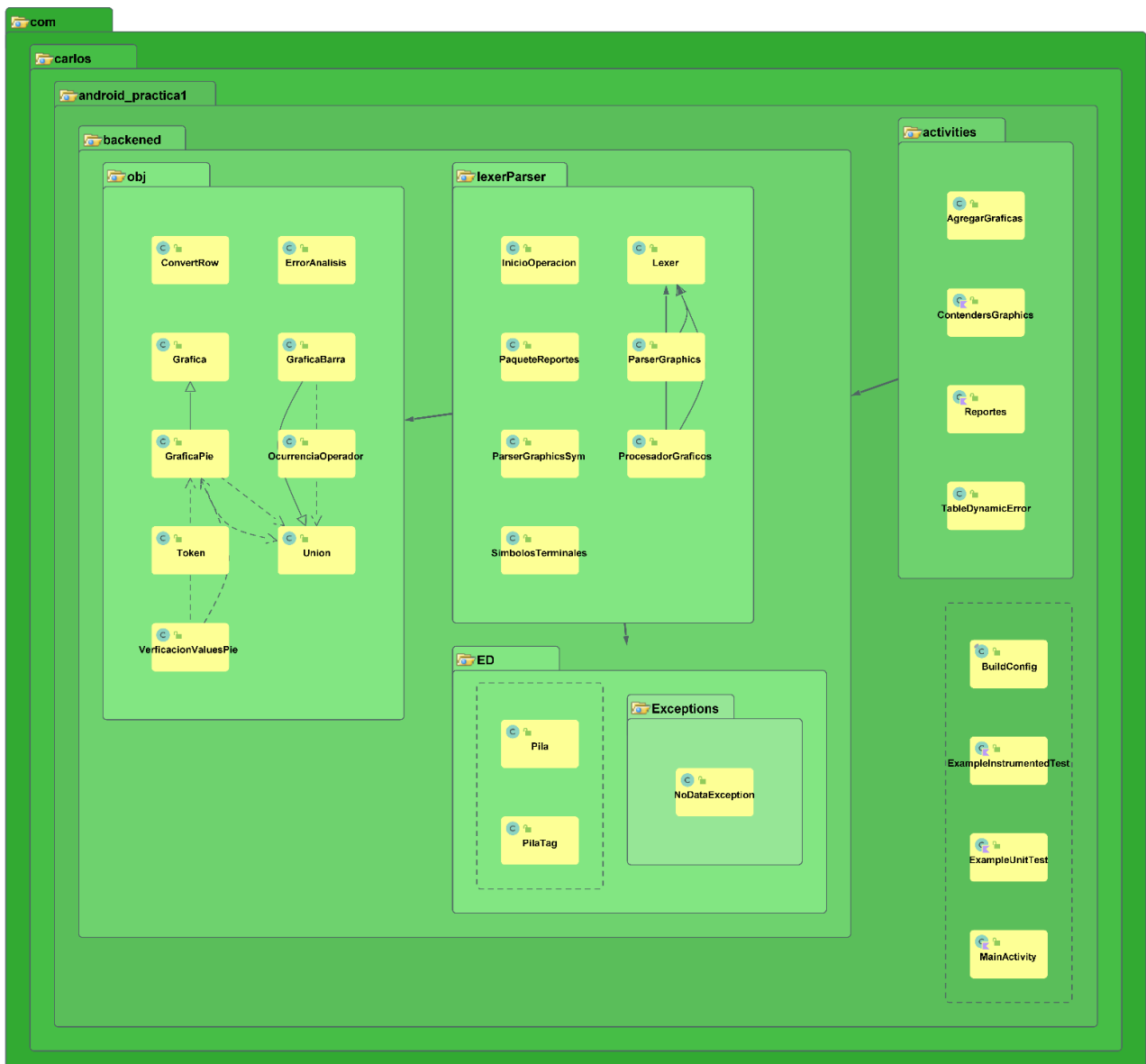
}

## Estructura del Proyecto:

Modulos:

**Android-Practica1.app**

Vista de Paquetes:



## Vista de Clases:

