Objetivos generales

- Aplicar conocimientos de análisis léxico y sintáctico, así como el manejo de errores para crear herramientas útiles.

Objectivos específicos

- Crear expresiones regulares y gramaticales libres de contexto complejas.
- Implementar las fases de análisis léxico y sintáctico de un compilador.
- Combinar la funcionalidad de JFlex y Cup en aplicaciones reales.
- Adquirir conocimientos sobre aplicaciones móviles

Descripción de la actividad

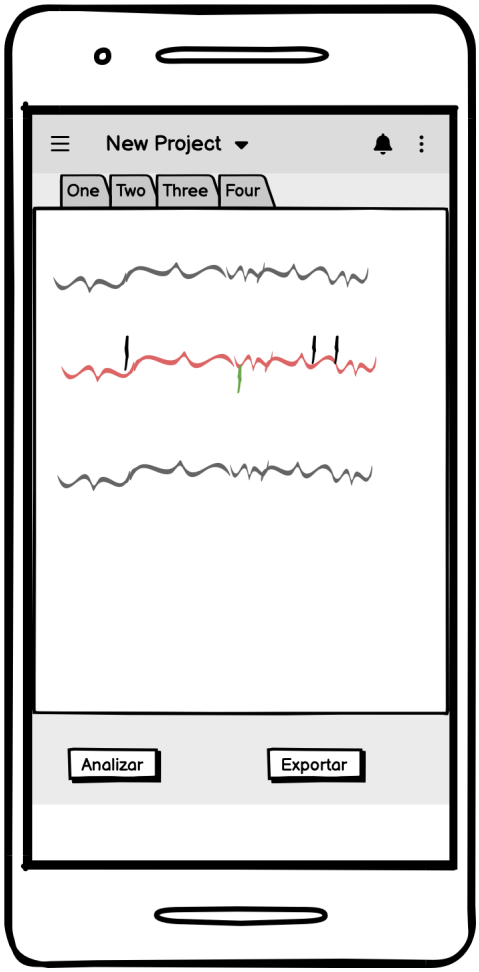
En la actualidad, existen aplicaciones móviles por todos lados, con la posibilidad que tienen los teléfonos inteligentes podemos tener aplicaciones que desarrollen tareas por nosotros en todos momento en la palma de nuestra mano. Esto nos facilita bastante el trabajo sin necesidad de tener un dispositivo de gran tamaño con nosotros. Sin embargo a pesar de la gran cantidad de aplicaciones que existen para móviles no se cuenta con un ide al estilo de programación para la generación de gráficas por lo que es necesario se construya una con las siguientes características.

Interfaz gráfica

La interfaz gráfica consta con una área de texto para el ingreso de código para la generación de gráfica además de un menú en el que permite abrir nuevos documentos, colocar nombres a los archivos o crear nuevos documentos con la extensión .gh no habrá botón para guardar ya que al momento de que se escriba en el documento este se deberá de guardar automáticamente. Además debe de haber un botón para realizar tanto el análisis de código como la ejecución de este para poder realizar las distintas gráficas que el usuario haya agregado. De haber errores que no se pueden solucionar se deberá notificar al usuario. Recuerde que es una aplicación móvil por lo que la interfaz gráfica debe estar adaptada a una pantalla pequeña.

Puede utilizar la siguiente imagen de referencia:

Mobile Apps



Tome en cuenta que se pueden tener varios documentos de distintos proyectos a la vez como se muestra en la imagen cada uno tiene una pestaña diferente.

Codificación

La aplicación será un IDE adaptado a dispositivos móviles en los que los usuarios puedan colocar su código con el cual podrán generar Dashboards generados como páginas web que van a contener las distintas páginas que así lo requieran. La forma de describir las gráficas es por medio de una estructura similar a la JSON pero con algunas variantes como se muestra a continuación.

```
{
```

```
 "title": "Mi Página Web",
 "description": "Una página web de ejemplo",
 "keywords": ["página web", "ejemplo", "JSON"],
 "header": { "title": "Bienvenido a mi sitio web" },
 "footer": { "copyright": "© 2024" },
 "backgroundColor": "#f0f0f0",
 "fontFamily": "Arial, sans-serif",
 "fontSize": "16px"


{
 "data": [
   { "category": "A", "value": 30 },
   { "category": "B", "value": 50 },
   { "category": "C", "value": 20 },
   { "category": "D", "value": 45 },
   { "category": "E", "value": 60 }
 ]
}

}
```

Todo nuestro código escrito siempre tiene que ir dentro de llaves, Antes de comenzar a describir nuestras gráficas se deben dar ciertos atributos a nuestra página como por ejemplo un título del Dashboard, una descripción que será opcional, palabras claves para la búsqueda, un header y un footer para agregar mayor información a nuestra página, la familia de la fuente que utilizaremos y el tamaño de letra que se utilizará en todos los textos por último un color de fondo si así lo quiere el usuario de lo contrario se agrega el color blanco hueso.

Para iniciar nuestro código podemos hacerlo directamente escribiendo nuestras características variables o código, o agregando todo esto dentro de una pareja de llaves.

Gráficas

Gráfica de barras

Habrá dos formas de generar una gráfica de barras una será de forma simple y la otra de forma expandida, la forma simple únicamente se le enviaran los datos a graficar y la aplicación deberá generarla, para la segunda forma que es la extendida el usuario podrá agregar características extra como las siguientes.

```
{
 "data": [
   { "category": "A", "value": 30 },
   { "category": "B", "value": 50 },
   { "category": "C", "value": 20 },
   { "category": "D", "value": 45 },
   { "category": "E", "value": 60 }
 ]
}
```

En este ejemplo:

- "data" es un arreglo que contiene objetos representando cada categoría y su valor.
- Cada objeto tiene dos propiedades:
  - "category": La etiqueta o nombre de la categoría.
  - "value": El valor numérico asociado con esa categoría, que determinará la altura de la barra en la gráfica.

```
{
 "data": [
   { "category": "A", "value": 30, "color": "#ff5733" },
   { "category": "B", "value": 50, "color": "#33ff57" },
   { "category": "C", "value": 20, "color": "#5733ff" },
   { "category": "D", "value": 45, "color": "#ffff33" },
   { "category": "E", "value": 60, "color": "#33ffff" }
 ],
 "chart": {
   "title": "Gráfica de Barras",
   "xAxisLabel": "Categorías",
   "yAxisLabel": "Valores"
```

```
  }
}
```

En este ejemplo extendido:

- Se han añadido propiedades adicionales a cada objeto en el arreglo "data":
  - "color": Define el color de cada barra en la gráfica.
- Se ha añadido un objeto "chart" para proporcionar metadatos adicionales sobre la gráfica:
  - "title": El título de la gráfica.
  - "xAxisLabel": La etiqueta para el eje X.
  - "yAxisLabel": La etiqueta para el eje Y.

Gráfica de pastel

Para esta grafica de pastel igualmente habran dos formas para poder elaborar la grafica una forma simple y la otra extandida, un ejemplo de cada forma es la siguiente:

```
{
  "data": [
    { "label": "A", "value": 30 },
    { "label": "B", "value": 50 },
    { "label": "C", "value": 20 }
  ]
}
```

En este ejemplo:

- "data" es un arreglo que contiene objetos representando cada segmento del pastel y su valor.
- Cada objeto tiene dos propiedades:
  - "label": La etiqueta o nombre del segmento.
  - "value": El valor numérico asociado con ese segmento, que determinará su tamaño proporcional en la gráfica de pastel.

```
{
```

```
 "data": [
   { "label": "A", "value": 30, "color": "#ff5733" },
   { "label": "B", "value": 50, "color": "#33ff57" },
   { "label": "C", "value": 20, "color": "#5733ff" }
 ],
 "chart": {
   "title": "Gráfica de Pastel",
   "legendPosition": "bottom"
 }
}
```

En este ejemplo extendido:

- Se han añadido propiedades adicionales a cada objeto en el arreglo "data":
  - "color": Define el color de cada segmento en la gráfica de pastel.
- Se ha añadido un objeto "chart" para proporcionar metadatos adicionales sobre la gráfica:
  - "title": El título de la gráfica de pastel.
  - "legendPosition": La posición de la leyenda en la gráfica (en este caso, en la parte inferior).

Gráfica de puntos

Para poder dibujar esta gráfica se deberá de crear una estructura como la siguiente donde de igual forma habrá dos maneras de poder generar estas gráficas.

```
{
 "data": [
   { "x": 10, "y": 20 },
   { "x": 30, "y": 40 },
   { "x": 50, "y": 60 },
   { "x": 70, "y": 80 },
   { "x": 90, "y": 100 }
 ]
}
```

En este ejemplo:

- "data" es un arreglo que contiene objetos representando cada punto en el gráfico de puntos.
- Cada objeto tiene dos propiedades:

- "x": El valor numérico de la coordenada en el eje X.
- "y": El valor numérico de la coordenada en el eje Y.

```
{
 "data": [
   { "x": 10, "y": 20, "size": 5, "color": "#ff5733" },
   { "x": 30, "y": 40, "size": 8, "color": "#33ff57" },
   { "x": 50, "y": 60, "size": 10, "color": "#5733ff" },
   { "x": 70, "y": 80, "size": 6, "color": "#ffff33" },
   { "x": 90, "y": 100, "size": 7, "color": "#33ffff" }
 ],
 "chart": {
   "title": "Gráfico de Puntos",
   "xAxisLabel": "Eje X",
   "yAxisLabel": "Eje Y"
 }
}
```

En este ejemplo extendido:

- Se han añadido propiedades adicionales a cada objeto en el arreglo "data":
  - "size": Define el tamaño del punto en el gráfico.
  - "color": Define el color del punto en el gráfico.
- Se ha añadido un objeto "chart" para proporcionar metadatos adicionales sobre la gráfica:
  - "title": El título del gráfico de puntos.
  - "xAxisLabel": La etiqueta para el eje X.
  - "yAxisLabel": La etiqueta para el eje Y.

Gráfica de líneas

Para este tipo de gráfico tenemos como en los demás dos opciones pero además podemos poner en comparación dos grupos de datos en el mismo gráfico el ejemplo es el siguiente.

```
{
 "data": [
   {
```

```
    "name": "Serie 1",
    "points": [
      { "x": 1, "y": 10 },
      { "x": 2, "y": 20 },
      { "x": 3, "y": 15 },
      { "x": 4, "y": 25 },
      { "x": 5, "y": 30 }
    ]
   },
   {
    "name": "Serie 2",
    "points": [
      { "x": 1, "y": 15 },
      { "x": 2, "y": 25 },
      { "x": 3, "y": 20 },
      { "x": 4, "y": 30 },
      { "x": 5, "y": 35 }
    ]
   }
  ],
  "chart": {
   "title": "Gráfico de Líneas",
   "xAxisLabel": "Eje X",
   "yAxisLabel": "Eje Y"
  }
}
```

En este ejemplo:

- "data" es un arreglo que contiene objetos representando cada serie de datos en la gráfica de líneas.
- Cada objeto de serie tiene dos propiedades:
  - "name": El nombre de la serie.
  - "points": Un arreglo de objetos representando los puntos de coordenadas (x, y) de la serie.
- Cada objeto "points" contiene dos propiedades:
  - "x": El valor numérico de la coordenada en el eje X.
  - "y": El valor numérico de la coordenada en el eje Y.

```
{
 "data": [
   {
    "name": "Serie 1",
    "points": [
      { "x": 1, "y": 10, "label": "Punto 1" },
      { "x": 2, "y": 20, "label": "Punto 2" },
      { "x": 3, "y": 15, "label": "Punto 3" },
      { "x": 4, "y": 25, "label": "Punto 4" },
      { "x": 5, "y": 30, "label": "Punto 5" }
    ],
    "color": "#ff5733",
    "lineStyle": "solid"
   },
   {
    "name": "Serie 2",
    "points": [
      { "x": 1, "y": 15, "label": "Punto 1" },
      { "x": 2, "y": 25, "label": "Punto 2" },
      { "x": 3, "y": 20, "label": "Punto 3" },
      { "x": 4, "y": 30, "label": "Punto 4" },
      { "x": 5, "y": 35, "label": "Punto 5" }
    ],
    "color": "#33ff57",
    "lineStyle": "dashed"
   }
 ],
 "chart": {
   "title": "Gráfico de Líneas",
   "xAxisLabel": "Eje X",
   "yAxisLabel": "Eje Y"
 }
}
```

En este ejemplo extendido:

- Se han añadido propiedades adicionales a cada serie en el arreglo "data":
    - "color": Define el color de la línea de la serie.

    - "lineStyle": Define el estilo de línea de la serie (por ejemplo, "solid" para sólida, "dashed" para discontinua).
- Cada punto en la serie también tiene una propiedad "label" para proporcionar etiquetas a los puntos individuales.

Tarjetas de información

Estas tarjetas servirán para mostrar información adicional de un solo dato su estructura es la siguiente.

```
{
 "data": {
  "value": 75,
  "label": "Porcentaje completado",
  "description": "Este es el porcentaje de completado de la tarea actual."
 }
}
```

En este ejemplo:

- "data" es un objeto que contiene la información del solo dato a mostrar en la tarjeta.
- El objeto "data" tiene varias propiedades:
    - "value": El valor numérico que se muestra en la tarjeta (por ejemplo, un porcentaje, una cantidad, etc.).
    - "label": Una etiqueta descriptiva para el dato (por ejemplo, el título de la tarjeta).
    - "description": Una descripción opcional que proporciona más información sobre el dato mostrado en la tarjeta.

```
{
 "data": {
  "value": 42,
  "label": "Puntos obtenidos",
  "description": "Estos son los puntos obtenidos en la última evaluación.",
  "icon": "fa-star",
  "color": "#FFD700",
  "link": "https://example.com/more-info"
 }
}
```

- Se han añadido propiedades adicionales al objeto "data":
  - "icon": El nombre de un ícono (por ejemplo, de Font Awesome) que podría representar visualmente el tipo de dato. También puede ser el enlace de alguna imagen.
  - "color": Un color específico para resaltar el dato en la tarjeta.
  - "link": Un enlace opcional que podría llevar a más detalles o información relacionada con el dato.

Sentencias de código

Variables

Dentro de nuestro código nosotros podremos declarar variables de tipo entero, cadena, decimal y booleano, Para poder nombrar nuestras variables tenemos que seguir la siguiente estructura: Podemos utilizar cualquier letra o número, Podemos usar mayúsculas o minúsculas pero siempre tiene que terminar con la palabra reservada id como se ve en los ejemplos:

```
n1Id = 2;
numero6ID = 45.6;
algomasid = true;
ID = “cadena”;
```

Además se pueden realizar operaciones sencillas entre enteros y decimales como los siguientes:

```
2 * 6;
ID + 6;
46.3 / ID;
5 - ID;
ID++;
Pid - - ;
valueID += 5;
valueID -= 5;
valueID *= 5;
valueID /= 5;
```

Comparadores

Con el fin de poder hacer comparaciones dentro de nuestras sentencias vamos a tener símbolos para poder realizar distintas comparaciones dentro de nuestro código las aceptadas por nuestro lenguaje son las siguientes:

==
!=
>
<
<=
>=

If y else

Se podrán crear estructuras de código que ayudarán al usuario a generar gráficas, estas estructuras de código funcionarán como en un lenguaje de programación y podrán interactuar con las gráficas estando dentro de nuestros archivos .gh de la siguiente manera:

```
n1Id = 2;

if (n1Id == 2) {
 {
 "data": [
   { "category": "A", "value": 30, "color": "#ff5733" },
   { "category": "B", "value": 50, "color": "#33ff57" },
   { "category": "C", "value": 20, "color": "#5733ff" },
   { "category": "D", "value": 45, "color": "#ffff33" },
   { "category": "E", "value": 60, "color": "#33ffff" }
 ],
 "chart": {
   "title": "Gráfica de Barras",
   "xAxisLabel": "Categorías",
   "yAxisLabel": "Valores"
 }
}

} else {
  {
 "data": [
```

```
    { "category": "A", "value": 30, "color": "#ff5733" },
    { "category": "B", "value": 50, "color": "#33ff57" },
    { "category": "C", "value": 20, "color": "#5733ff" },
    { "category": "D", "value": 45, "color": "#ffff33" },
    { "category": "E", "value": 60, "color": "#33ffff" }
  ]
}
}
```

NOTA: tomar en cuenta que cada una de nuestras estructuras de gráfica siempre va dentro de un par de llaves.

For

Para este ciclo tomar en cuenta que si aún no ha sido declarada la variable id declarar únicamente para este caso si ya está declarada usará la misma variable.

```
for ( id = 1; id <= 5; id++) {
  {
 "data": [
    { "category": "A", "value": 30, "color": "#ff5733" },
    { "category": "B", "value": 50, "color": "#33ff57" },
    { "category": "C", "value": 20, "color": "#5733ff" },
    { "category": "D", "value": 45, "color": "#ffff33" },
    { "category": "E", "value": 60, "color": "#33ffff" }
  ],
  "chart": {
    "title": "Gráfica de Barras",
    "xAxisLabel": "Categorías",
    "yAxisLabel": "Valores"
  }
}

}
```

While y do while

Estos bucles también serán necesarios para nuestros desarrollos, a continuación puede ver un ejemplo de cómo usarlos correctamente:

```
valueId = 2;
```

```
while (valueId <= 64) {
 {
 "data": [
   { "category": "A", "value": 30, "color": "#ff5733" },
   { "category": "B", "value": 50, "color": "#33ff57" },
   { "category": "C", "value": 20, "color": "#5733ff" },
   { "category": "D", "value": 45, "color": "#ffff33" },
   { "category": "E", "value": 60, "color": "#33ffff" }
 ]
}

  valueId ++;
}
```

NOTA: al igual que en el ciclo for podemos usar el ++ para sumar un número a la variable.

```
valueID = 5;
multiplierID = 2;

do {
  {
 "data": [
   { "category": "A", "value": 30 },
   { "category": "B", "value": 50 },
   { "category": "C", "value": 20 * multiplierID },
   { "category": "D", "value": 45 },
   { "category": "E", "value": 60 }
 ]
}

  valueID += 5;
} while (valueID <= 25);
```

IDE

Color

Para mejor comprensión del código se deberá colorear el código dentro del ide de las siguiente forma:

```
{
  "data": {
    "value": 42,
    "label": "Puntos obtenidos",
    "description": "Estos son los puntos obtenidos en la última evaluación.",
    "icon": "fa-star",
    "color": "#FFD700",
    "link": "https://example.com/more-info"
  }
}
```

Las llaves son de color rosado, los nombres de los datos antes de los dos puntos de color naranja, los numeros enteros decimales o porcentajes de color verde, losa cadenas entre comillas luego de los dos puntos de color morado y por ultimo si detecta enlaces, colores o correos en color azul, para todo los demas en que no este contemplado colocarlo en color blanco.

Comparadores de color Azul claro o celeste:

==
!=
>
<
<=
>=

Palabras reservadas de código amarillo:

if
else
do
while
for

NOTA: Realizar el ide en colores oscuros tanto el tema de menús como el fondo de donde se escribe el texto.

Indentación

Se debe tener alguna opción para que el usuario pueda genera la indentación de código con el fin de tener un mejor orden del código, para esto puede apoyarse de los elementos como: {} [] () para poder generar una sangría en las líneas dentro de estos bloques de código o al momento de que superen la pantalla del dispositivo dándonos un resultado como el siguiente:

```
{
  "data": {
    "value": 42,
    "label": "Puntos obtenidos",
    "description": "Estos son los puntos obtenidos

          en la última evaluación.",
    "icon": "fa-star",
    "color": "#FFD700",
    "link": "https://example.com/more-info"
  }
}
```

Insercion de codigo

Con la finalidad de ayudar al usuario, la aplicación deberá tener un menú para poder insertar código que facilite la creación de los gráficos. Este menú deberá tener opción de insertar plantillas para los Cuatro tipos de gráficos que se pueden crear pero también para las tarjetas de información, cuando el usuario de clic sobre alguno de los gráficos que quiera insertar el aplicativo deberá insertar código donde se encuentre el puntero dentro de nuestra área de texto, con datos ficticios que el usuario pueda cambiar pero si no los cambia el programa debería funcionar sin ningún problema con datos de prueba.

Soporte ante errores

El ide debe ser capaz de detectar errores al momento de ir escribiendo. Al momento de encontrar uno deberá colorear en color rojo la parte que contiene el error como se muestra a continuación.

```
{
  "data": {
    "value": 42,
    "label": "Puntos obtenidos",
    "description": "Estos son los puntos obtenidos en la última evaluación.,
    "icon": "fa-star",
```

```
    "color": "#FFD700",
    "link": "https://example.com/more-info"
  }
}
```

En este caso falta las comillas de que finalizan la cadena por ende debe marcar el error al momento de ejecutar el botón para analizar el documento mostrará información específica del error en este caso “HACEN FALTA COMILLAS AL FINAL DEL TEXTO” además de mostrar la ubicación exacta del error.

Para que el usuario encuentre mucho más fácil el error el área de texto debe poder mostrar las filas numeradas además de mostrar el número de fila o columna en el que se encuentra el puntero.

El analizador léxico como sintáctico debe ser capaz de poder recuperarse ante errores hasta terminar de leer el documento y marcar todos los errores que este contiene.

Recuerde que los errores deben de estar ubicados en un area al que se pueda acceder constantemente ya que si existen varios errores el usuario estará viendo este apartado en cada momento para poder solucionar todos los errores sin tener que estar analizando cada vez el documento.

Exportaciones

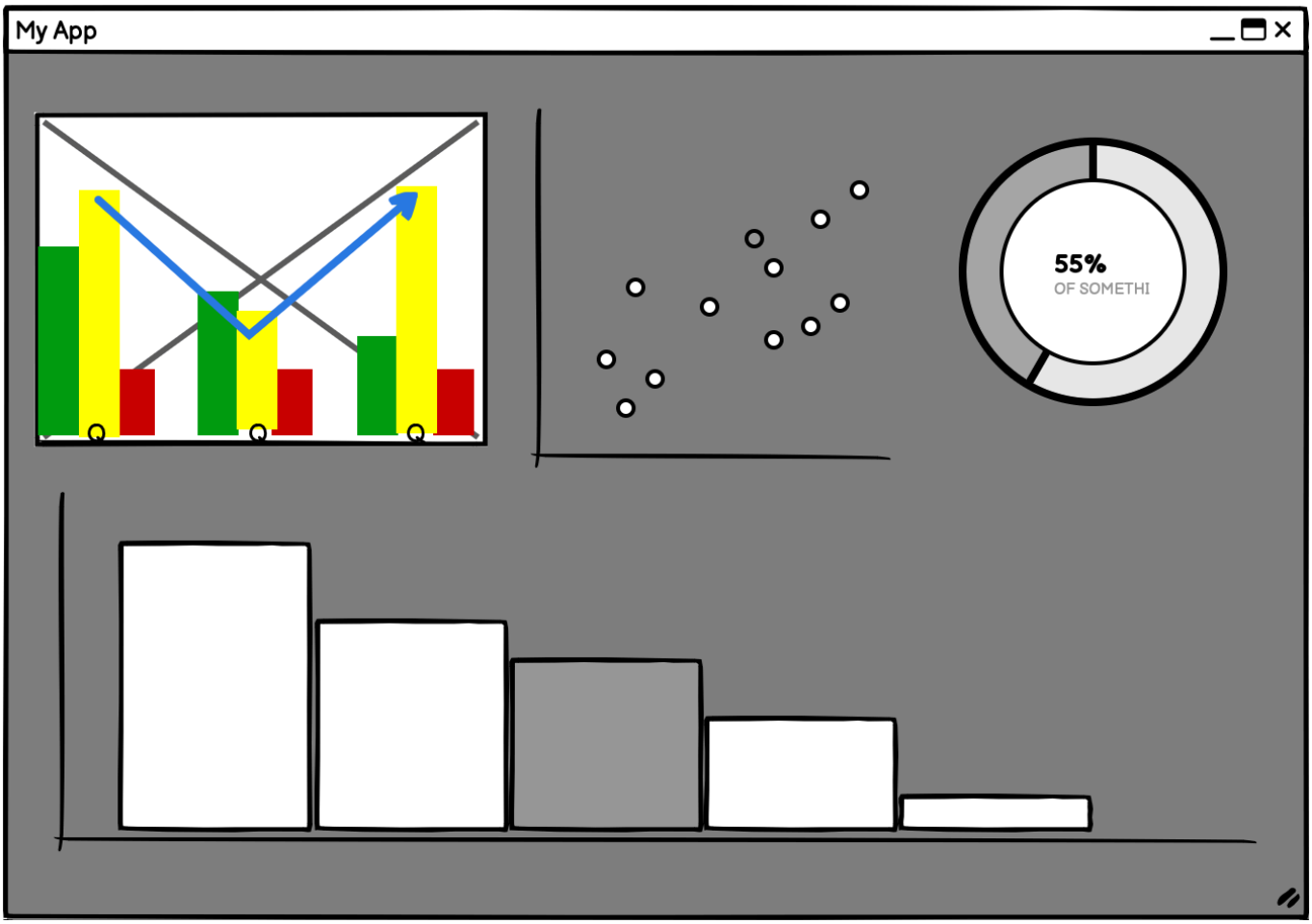
Para la exportación vamos a tener tres opciones de las cuales podemos escoger dentro del menú de aplicación, estas opciones únicamente deberán ejecutarse si no se encuentran errores dentro de nuestro documento,Las tres formas posibles son las siguientes:

- Página web
- Archivo PDF
- Imagen PNG

La forma principal es la página web que deberá darnos archivos HTML, CSS y/o Java script, ya sea uno o los tres, un ejemplo mínimo de lo que debería contener la página es lo siguiente:

CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN DE CIENCIAS DE LA INGENIERÍA
ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1
PRIMER SEMESTRE 2024
PROYECTO 2

Web Apps



Importante

- Usar lenguaje de programación Kotlin
- Usar herramientas Jflex y Cup para cualquier tipo de análisis lexico y sintáctico.
- Práctica obligatoria para tener derecho al siguiente proyecto.
- Las copias obtendrán nota de cero y se notificará a coordinación.
- Si se va a utilizar código de internet, entender la funcionalidad para que se tome como válido.

Entrega

La fecha de entrega es el día miercoles 15 de mayo a las 02:00 pm. Los componentes a entregar son:

- Código fuente (enlace del repo de github)
- Ejecutables (APK)
- Manual técnico incluyendo una sección con las expresiones regulares y las gramáticas usadas.
- Manual de usuario.

Calificación

Se calificará en la computadora de cada alumno por lo que es necesario que tengan instalado y configurado todo lo necesario.

```
}
"chart": {
    "title": "Gráfica de Barras",
    "xAxisLabel": "Categorías",
    "yAxisLabel": "Valores"

  }
,
"data": [
    { "category": "A", "value": 30, "color": "#ff5733" },
    { "category": "B", "value": 50, "color": "#33ff57" },
    { "category": "C", "value": 20, "color": "#5733ff" },
    { "category": "D", "value": 45, "color": "#ffff33" },
    { "category": "E", "value": 60, "color": "#33ffff" }
  ]
  }
```