Ministerul Educaţiei, Tineretului şi Sportului al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul Informatica si Ingineria sistemelor

RAPORT

Lucrare de laborator nr.4

la Programarea aplicațiilor mobile

Tema: **Custom Progress/Loading Bar**

A efectuat: st. gr. FI-181

Damean Alxandra

A verificat: prof.

Rusu Cristian

Chişinău 2020

**Scopul lucrării de laborator:**

* *Realizarea unui control grafic de vizualizare a progresului cu implementarea lui în practică.*
* *Se va alege la dorința dezvoltatorului o animație GIF sau Custom care va fi implementata în cod.*

1. **Elementele grafice rasteriale vor fi convertate în elemente de tip vectorial.**

**SVG** - un limbaj de markup Scalable Vector Graphics creat de World Wide Web Consortium (W3C) și inclus în subsetul limbajului de markup extensibil XML , este folosit pentru a descrie vectorul bidimensional și vectorul / rasterul mixtGrafică XML. Suportă atât grafică interactivă, cât și animată - sau, cu alte cuvinte, declarativă și scriptată.



 return (

    <View *style*={[{ alignItems: 'center', justifyContent: 'center' }]}>

      <Svg *height*="58" *width*="58" *viewBox*="0 0 58 58">

        <G *fill*="none" *fillRule*="evenodd">

          <G *transform*="translate(2 1)" *stroke*={props.backgroundColor} *stroke-width*="1.5">

            <Circle

*cx*="42.601"

*cy*="11.462"

*r*="5"

*fillOpacity*={String(newArr[0])}

*fill*={props.backgroundColor}

            ></Circle>

            <Circle

*cx*="49.063"

*cy*="27.063"

*r*="5"

*fillOpacity*={String(newArr[1])}

*fill*={props.backgroundColor}

            ></Circle>

            <Circle

*cx*="42.601"

*cy*="42.663"

*r*="5"

*fillOpacity*={String(newArr[2])}

*fill*={props.backgroundColor}

            ></Circle>

            <Circle

*cx*="27"

*cy*="49.125"

*r*="5"

*fillOpacity*={String(newArr[3])}

*fill*={props.backgroundColor}

            ></Circle>

            <Circle

*cx*="11.399"

*cy*="42.663"

*r*="5"

*fillOpacity*={String(newArr[4])}

*fill*={props.backgroundColor}

            ></Circle>

            <Circle

*cx*="4.938"

*cy*="27.063"

*r*="5"

*fillOpacity*={String(newArr[5])}

*fill*={props.backgroundColor}

            ></Circle>

            <Circle

*cx*="11.399"

*cy*="11.462"

*r*="5"

*fillOpacity*={String(newArr[6])}

*fill*={props.backgroundColor}

            ></Circle>

            <Circle

*cx*="27"

*cy*="5"

*r*="5"

*fillOpacity*={String(newArr[7])}

*fill*={props.backgroundColor}

            ></Circle>

          </G>

        </G>

      </Svg>

      <Text *style*={styles.textPercents}>{percent}%</Text>

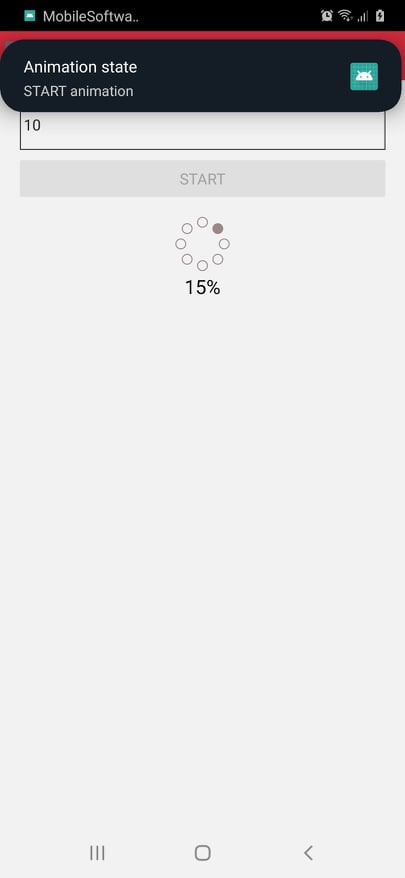
    </View>

  )

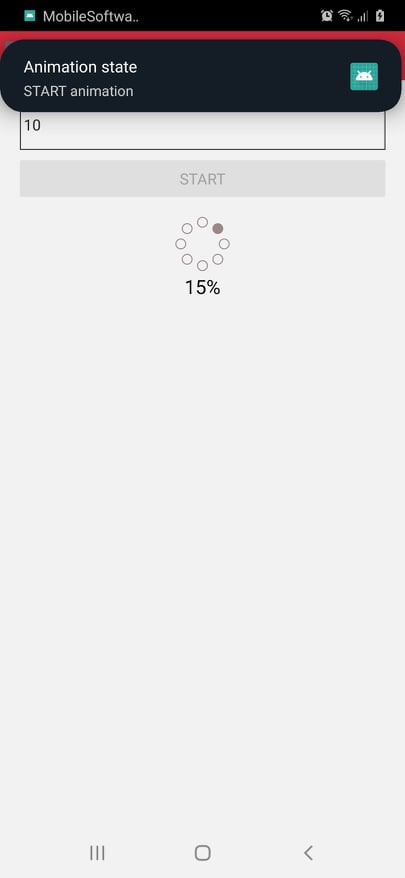
}

* Cotrolul grafic va fi suprapus pe activitatea curentă care la apelat.
* Controlul are implementat minim 3 evenimente (START, PROGRESS[%], STOP)

**Start/Progress:**



**Stop:**



export const Spinner: FC<SpinnerPropType> = (props: SpinnerPropType): ReactElement => {

  const percent = props.percent >= 100 ? 100 : props.percent

  const points = Math.floor(percent / (100.0 / MAX\_DIVISIONS)) || 1

  const [offset, setOffset] = useState(0)

  const divisions = divisionsArrayGenerator(points)

  useEffect(() => {

    const timer = setInterval(() => {

      setOffset((offs) => (offs + 1 >= MAX\_DIVISIONS ? 0 : offs + 1))

    }, 120)

    return () => {

      clearInterval(timer)

    }

  }, [])

  let newArr = divisions

  if (offset > 0) {

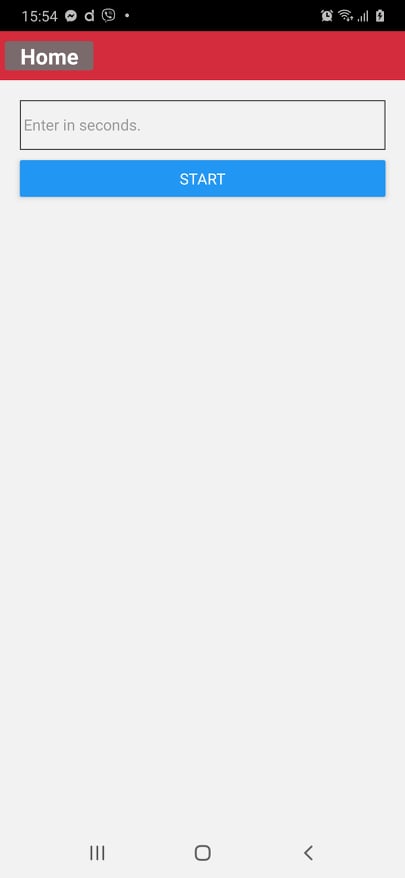
    const newArr1 = divisions.slice(0, offset)

    const newArr2 = divisions.slice(offset, divisions.length)

    newArr = [...newArr2, ...newArr1]

  }

**3.Testarea si prezentarea implementării va fi efectuată în cadrul unui TASK ASYNC cu un delay de execuție de maxim 5-10s.**



 intervalDelayCallback(): void {

    const DELAY = 0.05

    const newPosVal = this.state.position + (this.state.isIncrement ? DELAY : -DELAY)

    if (newPosVal >= 1.0) {

      this.setState({ position: 1.0, isIncrement: false })

      return

    }

    if (newPosVal <= 0.0) {

      this.setState({ position: 0.0, isIncrement: true })

      return

    }