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.3

la Programarea aplicațiilor mobile

Tema: Simple HTTP Client (Web Service Emulation)

A efectuat: st. gr. FI-181

Damean Alxandra

A verificat: prof.

Rusu Cristian

Chişinău 2020

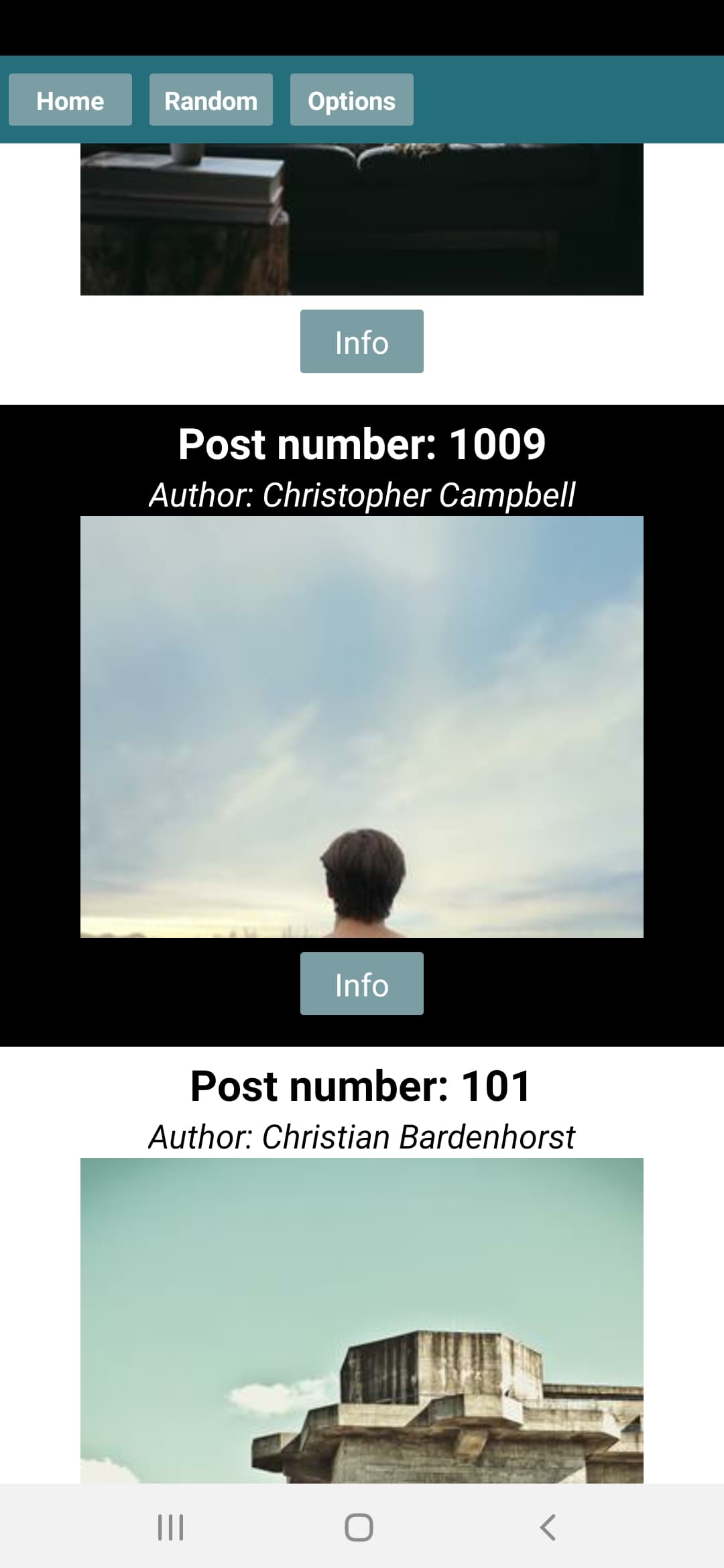
**Scopul lucrării de laborator:**De realizat emularea programatica a unui serviciu web.

Se va alege una din aceste puncte:

1. De realizat o aplicație ce va încărca de pe o resursă web (preferată, ex: https://news.yam.md/ro/rss ) fluxul RSS al acesteia.

Serviciul web va fi emulat programatic utilizând metodele protocolului HTTP (GET; )

1. posibilitate de adaugare 2 sau mai multe fluxuri RSS



type WebViewScreenPropType = {

  navigation: NavigationProp<ParamListBase>

  route: {

    params: {

      urlPath: string

    }

  }

}

class WebViewScreen *extends* Component<WebViewScreenPropType> {

  constructor(props: WebViewScreenPropType) {

    super(props)

  }

  render(): ReactElement {

    const uri: string = this.props.route.params.urlPath

    return <WebView *source*={{ uri }} />

  }

}

export default WebViewScreen

1. posibilitate de a naviga catre postarea din fluxul incarcat

*async* load(key: string): Promise<T | null> {

    const path = RNFS.ExternalDirectoryPath + `/${key}.txt`

    let result = null

    try {

      const jsonExists = await RNFS.exists(path)

      if (!jsonExists) {

        throw new Error()

      }

      const jsonString = await RNFS.readFile(path, 'utf8')

      result = JSON.parse(jsonString) as T

      console.log(`FileJSONStorage: File ${key} loaded with success.`)

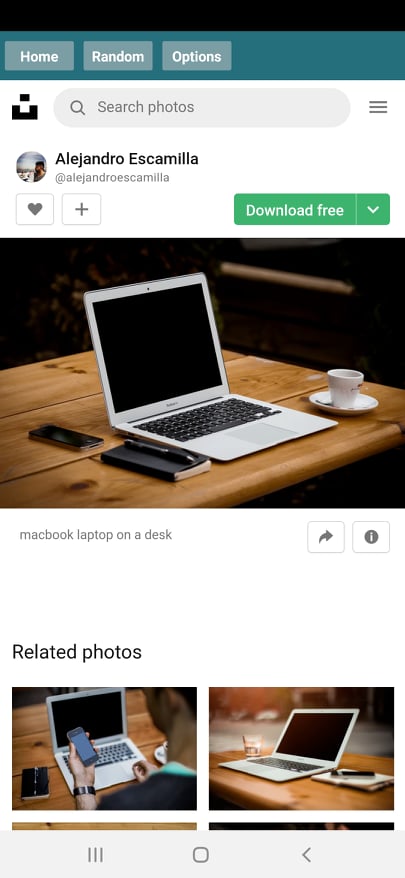
      return result

    } catch (error) {}

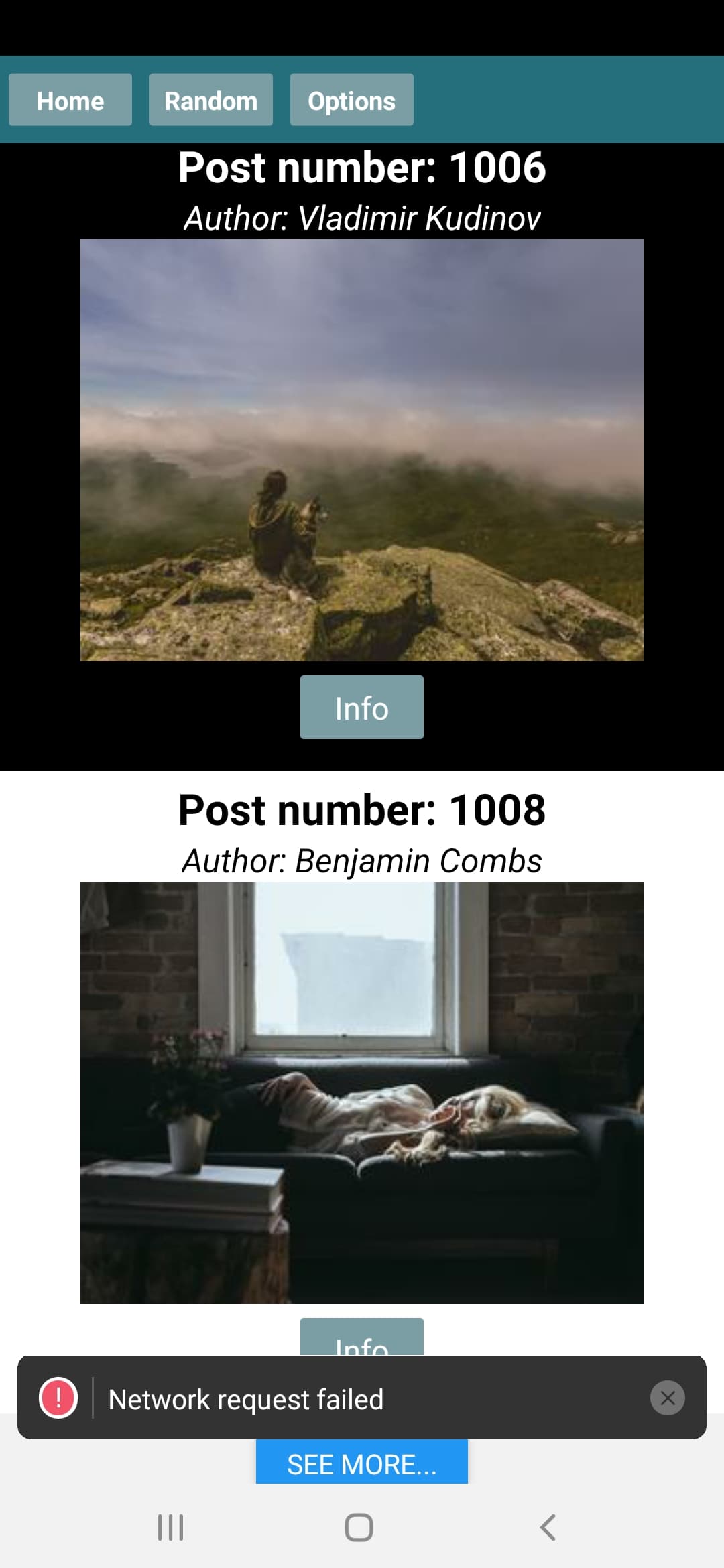
    return result

  }

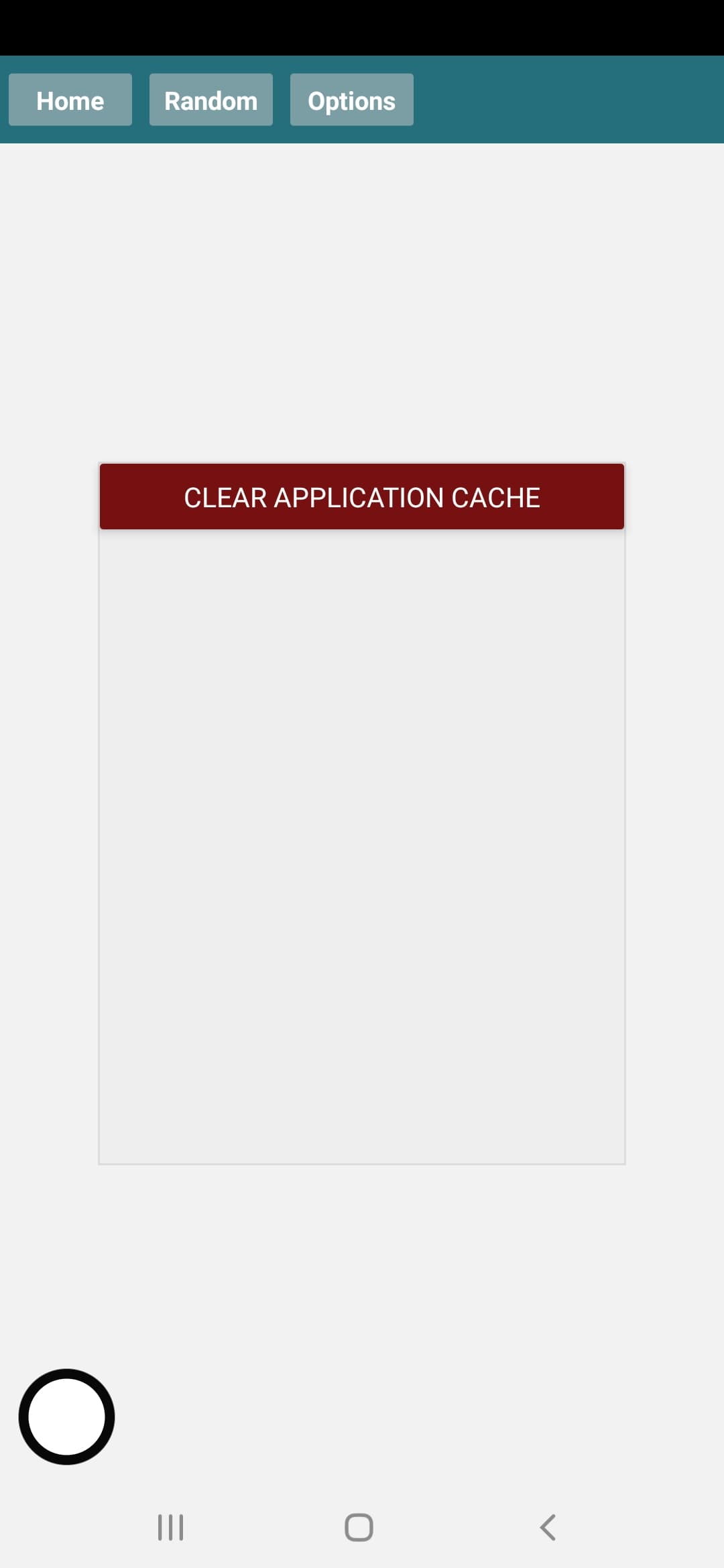
}



1. salvarea locala a fluxului cu păstrarea sa pînă utilizatorul nu o va distruge



Pagina pentru ștergere(după ce ștergem, apoi nu mai rămâne cash, și fără internet nu se incarcă nimic):



function clearApplicationCache() {

  Alert.alert(

    'Confirm',

    `Remove app cache ???`,

    [

      {

        text: 'Yes',

        onPress: *async* () => {

          const storage = new FileJSONStorage<unknown>()

          await storage.remove('data')

        },

      },

      {

        text: 'Cancel',

        onPress: () => {},

        style: 'cancel',

      },

    ],

    { cancelable: false },

  )

}

export const OptionsScreen: FC = (): ReactElement => {

  return (

    <View *style*={styles.container}>

      <View *style*={styles.childContainer}>

        <Button *color*="#711" *onPress*={clearApplicationCache} *title*="Clear application cache" />

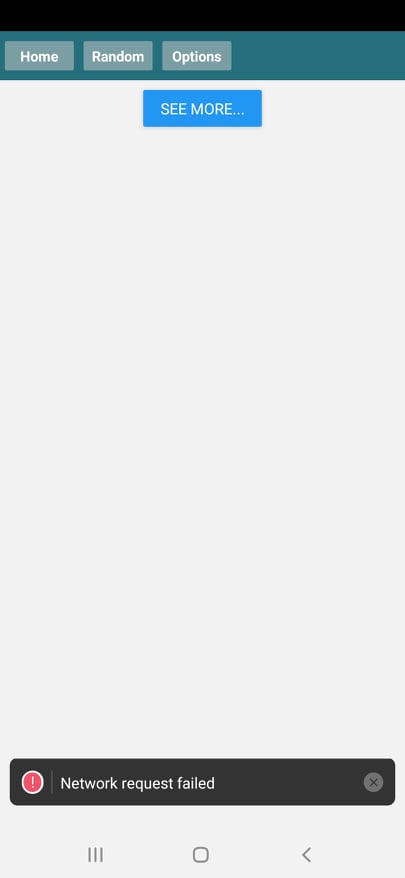
      </View>

    </View>

  )

}

Rezulatul:



**Random:**

type RandomScreenPropType = {

  navigation: NavigationProp<ParamListBase>

}

*async* function fetchRandomPostCallback(): Promise<Post> {

  try {

    const post: Post = await fetchRandomPost()

    return post

  } catch (error) {

    console.warn('FETCH RANDOM POST ERROR: ', error)

  }

  return {

    author: '',

    base64Image: '',

    downloadImageUrl: '',

    id: 0,

    webUrl: '',

  }

}

export default function RandomScreen(props: RandomScreenPropType): ReactElement {

  const componentIsMounted = useRef(true)

  const [post, setPost] = useState<Post>({

    author: '',

    base64Image: '',

    downloadImageUrl: '',

    id: 0,

    webUrl: '',

  })

  const onNewRandomButtonClick = useCallback(() => {

    fetchRandomPostCallback().then((post: Post) => {

      if (componentIsMounted.current) {

        setPost({ ...post })

      }

    })

  }, [])

  useEffect(() => {

    fetchRandomPostCallback().then((post: Post) => {

      if (componentIsMounted.current) {

        setPost({ ...post })

      }

    })

    return () => {

      componentIsMounted.current = false

    }

  }, [])

  return (

    <View>

      <Card

*id*={post.id}

*author*={post.author}

*base64Image*={post.base64Image}

*onInfoClick*={() =>

          props.navigation.navigate(routeNames.WebView, {

            urlPath: post.webUrl,

          })

        }

      />

      <View *style*={styles.buttonContainer}>

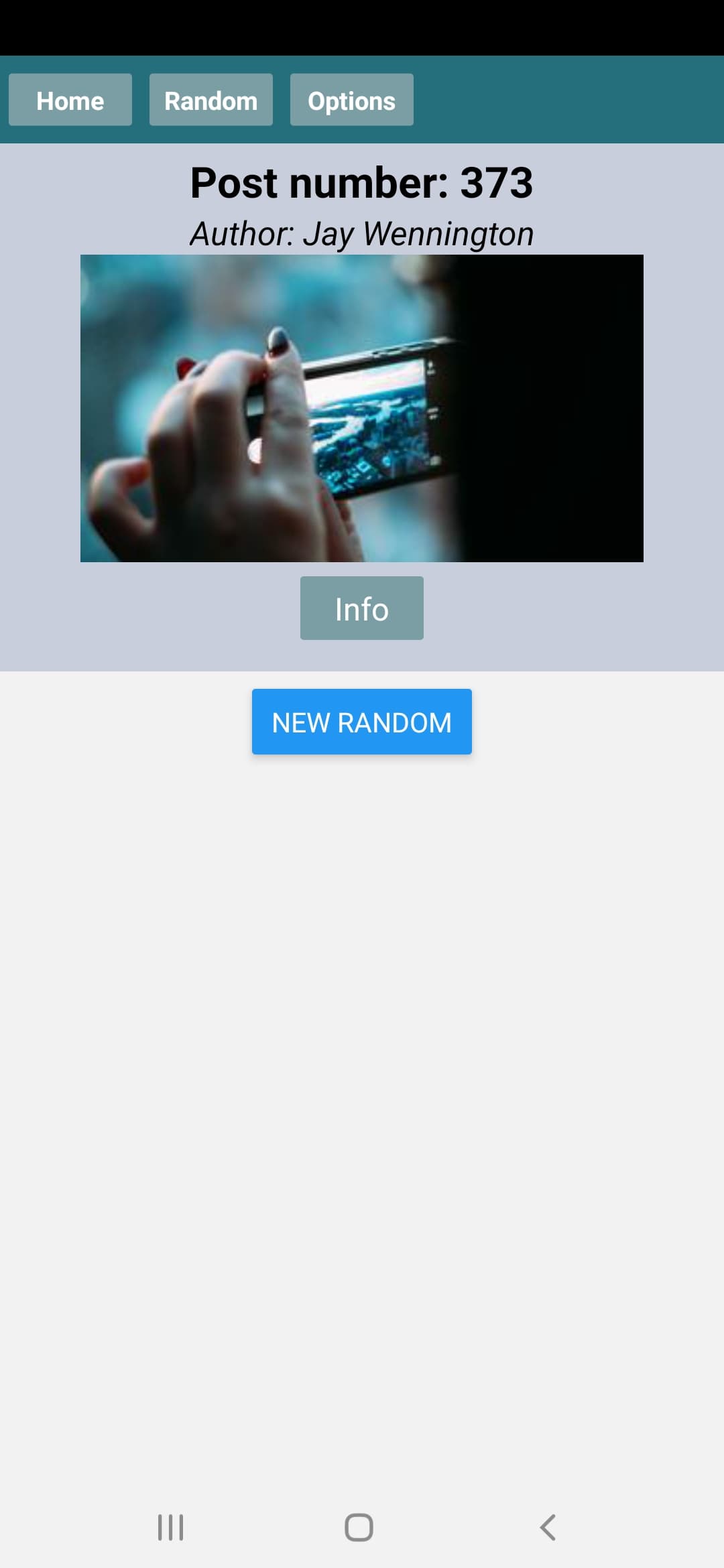
        <Button *title*="New random" *onPress*={onNewRandomButtonClick} />

      </View>

    </View>

  )

}

****