Criando um Bloco de Notas

Nessa aula vamos criar um bloco de notas e para isso vamos criar um projeto. Abra o VS Code na pasta onde você deseja criar o seu projeto e em seguida abra o terminal do Vs Code e digite o comando:

npx create-expo-app –template

Na hora de colar no terminal, certifique-se que tem dois traços (--) antes da palavra "template", para não dar erro na hora da instalação.

Em seguida escolha a opção "**Blank**" e dê um nome ao seu projeto.

Depois de criado o seu projeto, **entre na pasta do seu projeto para realizar a instalação das dependências** que serão utilizadas no projeto, vá até o arquivo **package.json**:

package.json

```
{
  "name": "animated-todo",
  "description": "A smoothly animated ToDo app built with React Native",
  "version": "1.0.0",
  "repository": {
    "type": "git",
    "url": "git+https://github.com/craftzdog/react-native-animated-todo.git"
  },
  "main": "node_modules/expo/AppEntry.js",
  "scripts": {
    "start": "expo start",
    "android": "expo start --android",
    "ios": "expo start --ios",
    "web": "expo start --web",
    "eject": "expo eject"
  },
```

```
"dependencies": {
  "@react-navigation/drawer": "^6.1.8",
  "@react-navigation/native": "^6.0.6",
  "expo": "^46.0.9",
  "expo-linking": "~3.2.2",
  "expo-modules-core": "^0.11.4",
  "expo-status-bar": "~1.4.0",
  "moti": "^0.16.1",
  "native-base": "^3.2.1",
  "react": "^18.0.0",
  "react-dom": "^18.0.0",
  "react-native": "^0.69.5",
  "react-native-checkbox-reanimated": "^0.1.0",
  "react-native-gesture-handler": "~2.5.0",
  "react-native-reanimated": "~2.9.1",
  "react-native-safe-area-context": "^4.3.1",
  "react-native-screens": "~3.15.0",
  "react-native-svg": "^12.3.0",
  "react-native-web": "^0.18.8",
  "react-stately": "^3.17.0",
  "shortid": "^2.2.16",
  "styled-components": "^5.3.3",
  "styled-system": "^5.1.5"
},
"devDependencies": {
  "@babel/core": "^7.12.9",
  "@types/react": "~17.0.21",
  "@types/react-native": "~0.64.12",
```

```
    "@types/shortid": "^0.0.29",
    "prettier": "^2.4.1",
    "typescript": "~4.3.5"
  },
  "private": true
}
```

Agora vamos iniciar a configuração do nosso projeto. Para isso, abra o arquivo **tsconfig.json.**

./tsconfig.json

```
{
  "extends": "expo/tsconfig.base",
  "compilerOptions": {
    "strict": true,
    "allowSyntheticDefaultImports": true,
    "jsx": "react-native",
    "lib": [
      "dom",
      "esnext"
    ],
    "moduleResolution": "node",
    "noEmit": true,
    "skipLibCheck": true,
    "resolveJsonModule": true,
  }
}
```

Em seguida vamos configurar o arquivo **babel.config.js**

./babel.config.js

```
module.exports = function (api) {
  api.cache(true)
  return {
    presets: ['babel-preset-expo'],
    plugins: ['react-native-reanimated/plugin']
  }
}
```

O próximo passo é configurar o arquivo **prettier.config.js**, caso o arquivo não exista crie-o.

./prettier.config.js

```
const options ={
  arrowParens: 'avoid',
  singleQuote: true,
  backetSpacing: true,
  endOfLine: 'lf',
  semi: false,
  tabWidth: 2,
  trailingComma: 'none'
}
module.exports = options
```

Agora na raiz do projeto cole uma pasta chamada **"doc"** que será disponibilizada em sala contendo arquivos necessários para o projeto.

Em seguida, na raiz do projeto crie a pasta **src** e dentro dela cole uma outra pasta chamada **assets** onde ficará armazendo as nossas imagens. Após isso vamos criar a nossa pasta **components** e nela os nossos arquivos.

./src/components/animated-color-box.tsx

```
import React, { useEffect } from 'react'
import { Box, useToken } from 'native-base'
import usePrevious from '../utils/use-previous'
import Animated, {
  useSharedValue,
  useAnimatedStyle,
  withTiming,
  interpolateColor
} from 'react-native-reanimated'

const AnimatedBox = Animated.createAnimatedComponent(Box)

const AnimatedColorBox = ({ bg, ...props }: any) => {
  const hexBg = useToken('colors', bg)
  const prevHexBg = usePrevious(hexBg)
  const progress = useSharedValue(0)

  useEffect(() => {
    progress.value = 0
  }, [hexBg])
```

```
  const animatedStyles = useAnimatedStyle(() => {
    progress.value = withTiming(1, { duration: 200 })
    return {
      backgroundColor: interpolateColor(
        progress.value,
        [0, 1],
        [prevHexBg || hexBg, hexBg]
      )
    }
  }, [hexBg])
  return <AnimatedBox {...props} style={animatedStyles} />
}


export default AnimatedColorBox
```

O próximo componente será **animated-task-label.tsx.**

./src/components/animated-task-label.tsx

```
import React, { useEffect, memo } from 'react'
import { Pressable, Text, HStack, Box } from 'native-base'
import Animated, {
  Easing,
  useSharedValue,
  useAnimatedStyle,
  withTiming,
  withSequence,
  withDelay,
```

```
  interpolateColor
} from 'react-native-reanimated'

interface Props {
  strikethrough: boolean
  textColor: string
  inactiveTextColor: string
  onPress?: () => void
  children?: React.ReactNode
}

const AnimatedBox = Animated.createAnimatedComponent(Box)
const AnimatedHStack = Animated.createAnimatedComponent(HStack)
const AnimatedText = Animated.createAnimatedComponent(Text)

const AnimatedTaskLabel = memo((props: Props) => {
  const { strikethrough, textColor, inactiveTextColor, onPress, children } =
    props

  const hstackOffset = useSharedValue(0)
  const hstackAnimatedStyles = useAnimatedStyle(
    () => ({
      transform: [{ translateX: hstackOffset.value }]
    }),
    [strikethrough]
  )
```

```
const textColorProgress = useSharedValue(0)
const textColorAnimatedStyles = useAnimatedStyle(
  () => ({
    color: interpolateColor(
      textColorProgress.value,
      [0, 1],
      [textColor, inactiveTextColor]
    )
  }),
  [strikethrough, textColor, inactiveTextColor]
)
const strikethroughWidth = useSharedValue(0)
const strikethroughAnimatedStyles = useAnimatedStyle(
  () => ({
    width: `${strikethroughWidth.value * 100}%`,
    borderBottomColor: interpolateColor(
      textColorProgress.value,
      [0, 1],
      [textColor, inactiveTextColor]
    )
  }),
  [strikethrough, textColor, inactiveTextColor]
)

useEffect(() => {
  const easing = Easing.out(Easing.quad)
```

```
  if (strikethrough) {
    hstackOffset.value = withSequence(
      withTiming(4, { duration: 200, easing }),
      withTiming(0, { duration: 200, easing })
    )
    strikethroughWidth.value = withTiming(1, { duration: 400, easing })
    textColorProgress.value = withDelay(
      1000,
      withTiming(1, { duration: 400, easing })
    )
  } else {
    strikethroughWidth.value = withTiming(0, { duration: 400, easing })
    textColorProgress.value = withTiming(0, { duration: 400, easing })
  }
})

return (
  <Pressable onPress={onPress}>
    <AnimatedHStack alignItems="center" style={[hstackAnimatedStyles]}>
      <AnimatedText
        fontSize={19}
        noOfLines={1}
        isTruncated
        px={1}
        style={[textColorAnimatedStyles]}
      >
```

```
          {children}
        </AnimatedText>
        <AnimatedBox
          position="absolute"
          h={1}
          borderBottomWidth={1}
          style={[strikethroughAnimatedStyles]}
        />
      </AnimatedHStack>
    </Pressable>
  )
})

export default AnimatedTaskLabel
```

O próximo componente será **app-container.tsx.**

./src/components/app-container.tsx

```
import * as React from 'react'
import { NavigationContainer } from '@react-navigation/native'
import { NativeBaseProvider } from 'native-base'
import theme from '../theme'

type Props = {
  children: React.ReactNode
}
```

```
export default function AppContainer(props: Props) {
  return (
    <NavigationContainer>
      <NativeBaseProvider theme={theme}>{props.children}</NativeBaseProvider>
    </NavigationContainer>
  )
}
```

Agora vamos para o seguinte que será o **link-button.tsx.**

./src/components/link-button.tsx

```
import React, { useCallback } from 'react'
import * as Linking from 'expo-linking'
import { Button, IButtonProps } from 'native-base'

interface Props extends IButtonProps {
  href: string
}

const LinkButton = ({ href, ...props }: Props) => {
  const handlePress = useCallback(() => {
    Linking.openURL(href)
  }, [href])

  return <Button {...props} onPress={handlePress} />
}
```

```
export default LinkButton
```

Vamos seguir com **masthead.tsx**.

./src/components/masthead.tsx

```
import React from 'react'
import { ImageSourcePropType } from 'react-native'
import { Box, VStack, Heading, Image } from 'native-base'

interface Props {
  title: string
  image: ImageSourcePropType
  children: React.ReactNode
}

const Masthead = ({ title, image, children }: Props) => {
  return (
    <VStack h="300px" pb={5}>
      <Image
        position="absolute"
        left={0}
        right={0}
        bottom={0}
        w="full"
        h="300px"
        resizeMode="cover"
        source={image}
```

```
        alt="masthead image"
      />
      {children}
      <Box flex={1} />
      <Heading color="white" p={6} size="xl">
        {title}
      </Heading>
    </VStack>
  )
}

export default Masthead
```

O nosso próximo componente será **menu-button.tsx**

./src/components/menu-button.tsx

```
import React from 'react'
import { Button, Icon, IButtonProps } from 'native-base'
import { Feather } from '@expo/vector-icons'

interface Props extends IButtonProps {
  active: boolean
  icon: string
  children: React.ReactNode
}

const MenuButton = ({ active, icon, children, ...props }: Props) => {
```

```
return (
  <Button
    size="lg"
    _light={{
      colorScheme: 'blue',
      _pressed: {
        bg: 'primary.100'
      },
      _text: {
        color: active ? 'blue.50' : 'blue.500'
      }
    }}
    _dark={{
      colorScheme: 'darkBlue',
      _pressed: {
        bg: 'primary.600'
      },
      _text: {
        color: active ? 'blue.50' : undefined
      }
    }}
    bg={active ? undefined : 'transparent'}
    variant="solid"
    justifyContent="flex-start"
    leftIcon={<Icon as={Feather} name={icon} size="sm" opacity={0.5} />}
    {...props}
```

```
    >
      {children}
    </Button>
  )
}

export default MenuButton
```

Agora vamos criar o componente **navbar.tsx**

./src/components/navbar.tsx

```
import React, { useCallback } from 'react'
import { HStack, IconButton } from 'native-base'
import { Feather } from '@expo/vector-icons'
import { useNavigation } from '@react-navigation/native'
import { DrawerNavigationProp } from '@react-navigation/drawer'

const NavBar = () => {
  const navigation = useNavigation<DrawerNavigationProp<{}>>()
  const handlePressMenuButton = useCallback(() => {
    navigation.openDrawer()
  }, [navigation])

  return (
    <HStack w="full" h={40} alignItems="center" alignContent="center" p={4}>
      <IconButton
```

```
        onPress={handlePressMenuButton}
        borderRadius={100}
        _icon={{
          as: Feather,
          name: 'menu',
          size: 6,
          color: 'white'
        }}
      />
    </HStack>
  )
}

export default NavBar
```

Vamos continuar criando o componente **sidebar.tsx**

./src/components/sidebar.tsx

```
import React, { useCallback } from 'react'
import {
  HStack,
  VStack,
  Center,
  Avatar,
  Heading,
  IconButton,
  useColorModeValue
```

```
} from 'native-base'
import { DrawerContentComponentProps } from '@react-navigation/drawer'
import AnimatedColorBox from './animated-color-box'
import ThemeToggle from './theme-toggle'
import { Feather } from '@expo/vector-icons'
import MenuButton from './menu-button'

const Sidebar = (props: DrawerContentComponentProps) => {
  const { state, navigation } = props
  const currentRoute = state.routeNames[state.index]

  const handlePressBackButton = useCallback(() => {
    navigation.closeDrawer()
  }, [navigation])
  const handlePressMenuMain = useCallback(() => {
    navigation.navigate('Main')
  }, [navigation])

  return (
    <AnimatedColorBox
      safeArea
      flex={1}
      bg={useColorModeValue('blue.50', 'darkBlue.800')}
      p={7}
    >
      <VStack flex={1} space={2}>
```

```
<HStack justifyContent="flex-end">
  <IconButton
    onPress={handlePressBackButton}
    borderRadius={100}
    variant="outline"
    borderColor={useColorModeValue('blue.300', 'darkBlue.700')}
    _icon={{
      as: Feather,
      name: 'chevron-left',
      size: 6,
      color: useColorModeValue('blue.800', 'darkBlue.700')
    }}
  />
</HStack>
<Avatar
  source={require('../assets/profile-image.png')}
  size="xl"
  borderRadius={100}
  mb={6}
  borderColor="secondary.500"
  borderWidth={3}
/>
<Heading mb={4} size="xl">
  Nome do Usuário
</Heading>
<MenuButton
```

```
          active={currentRoute === 'Main'}
          onPress={handlePressMenuMain}
          icon="inbox"
        >
          Tarefas
        </MenuButton>
      </VStack>
      <Center>
        <ThemeToggle />
      </Center>
    </AnimatedColorBox>
  )
}

export default Sidebar
```

Vamos continuar criando o componente **swipable-view.tsx**

./src/components/swipable-view.tsx

```
import React from 'react'
import { Dimensions } from 'react-native'
import {
  PanGestureHandler,
  PanGestureHandlerGestureEvent,
  PanGestureHandlerProps
} from 'react-native-gesture-handler'
import Animated, {
```

```
  useAnimatedGestureHandler,
  useSharedValue,
  useAnimatedStyle,
  withTiming,
  runOnJS
} from 'react-native-reanimated'
import { Box } from 'native-base'
import { makeStyledComponent } from '../utils/styled'

const StyledView = makeStyledComponent(Animated.View)

interface Props extends Pick<PanGestureHandlerProps, 'simultaneousHandlers'> {
  children: React.ReactNode
  backView?: React.ReactNode
  onSwipeLeft?: () => void
}

const { width: SCREEN_WIDTH } = Dimensions.get('window')
const SWIPE_THRESHOLD = -SCREEN_WIDTH * 0.2

const SwipeView = (props: Props) => {
  const { children, backView, onSwipeLeft, simultaneousHandlers } = props
  const translateX = useSharedValue(0)

  const panGesture =
useAnimatedGestureHandler<PanGestureHandlerGestureEvent>({
```

```
  onActive: event => {
    translateX.value = Math.max(-128, Math.min(0, event.translationX))
  },
  onEnd: () => {
    const shouldBeDismissed = translateX.value < SWIPE_THRESHOLD
    if (shouldBeDismissed) {
      translateX.value = withTiming(-SCREEN_WIDTH)
      onSwipeLeft && runOnJS(onSwipeLeft)()
    } else {
      translateX.value = withTiming(0)
    }
  }
})

const facadeStyle = useAnimatedStyle(() => ({
  transform: [
    {
      translateX: translateX.value
    }
  ]
}))

return (
  <StyledView w="full">
    {backView && (
      <Box position="absolute" left={0} right={0} top={0} bottom={0}>
```

```
          {backView}
        </Box>
      )}
      <PanGestureHandler
        simultaneousHandlers={simultaneousHandlers}
        onGestureEvent={panGesture}
      >
        <StyledView style={facadeStyle}>{children}</StyledView>
      </PanGestureHandler>
    </StyledView>
  )
}

export default SwipeView
```

Vamos continuar criando o componente **task-item.tsx**

./src/components/task-item.tsx

```
import React, { useCallback } from 'react'
import { PanGestureHandlerProps } from 'react-native-gesture-handler'
import { NativeSyntheticEvent, TextInputChangeEventData } from 'react-native'
import {
  Pressable,
  Box,
  HStack,
  useColorModeValue,
  Icon,
```

```
  Input,
  useToken
} from 'native-base'
import AnimatedCheckbox from 'react-native-checkbox-reanimated'
import AnimatedTaskLabel from './animated-task-label'
import SwipableView from './swipable-view'
import { Feather } from '@expo/vector-icons'

interface Props extends Pick<PanGestureHandlerProps, 'simultaneousHandlers'> {
  isEditing: boolean
  isDone: boolean
  onToggleCheckbox?: () => void
  onPressLabel?: () => void
  onRemove?: () => void
  onChangeSubject?: (subject: string) => void
  onFinishEditing?: () => void
  subject: string
}

const TaskItem = (props: Props) => {
  const {
    isEditing,
    isDone,
    onToggleCheckbox,
    subject,
    onPressLabel,
```

```
  onRemove,
  onChangeSubject,
  onFinishEditing,
  simultaneousHandlers
} = props

const highlightColor = useToken(
  'colors',
  useColorModeValue('blue.500', 'blue.400')
)
const boxStroke = useToken(
  'colors',
  useColorModeValue('muted.300', 'muted.500')
)

const checkmarkColor = useToken('colors', useColorModeValue('white', 'white'))

const activeTextColor = useToken(
  'colors',
  useColorModeValue('darkText', 'lightText')
)
const doneTextColor = useToken(
  'colors',
  useColorModeValue('muted.400', 'muted.600')
)
```

```
const handleChangeSubject = useCallback(
  (e: NativeSyntheticEvent<TextInputChangeEventData>) => {
    onChangeSubject && onChangeSubject(e.nativeEvent.text)
  },
  [onChangeSubject]
)

return (
  <SwipableView
    simultaneousHandlers={simultaneousHandlers}
    onSwipeLeft={onRemove}
    backView={
      <Box
        w="full"
        h="full"
        bg="red.500"
        alignItems="flex-end"
        justifyContent="center"
        pr={4}
      >
        <Icon color="white" as={<Feather name="trash-2" />} size="sm" />
      </Box>
    }
  >
    <HStack
      alignItems="center"
```

```
  w="full"
  px={4}
  py={2}
  bg={useColorModeValue('warmGray.50', 'primary.900')}
>
  <Box width={30} height={30} mr={2}>
    <Pressable onPress={onToggleCheckbox}>
      <AnimatedCheckbox
        highlightColor={highlightColor}
        checkmarkColor={checkmarkColor}
        boxOutlineColor={boxStroke}
        checked={isDone}
      />
    </Pressable>
  </Box>
  {isEditing ? (
    <Input
      placeholder="Task"
      value={subject}
      variant="unstyled"
      fontSize={19}
      px={1}
      py={0}
      autoFocus
      blurOnSubmit
      onChange={handleChangeSubject}
```

```
          onBlur={onFinishEditing}
        />
      ) : (
        <AnimatedTaskLabel
          textColor={activeTextColor}
          inactiveTextColor={doneTextColor}
          strikethrough={isDone}
          onPress={onPressLabel}
        >
          {subject}
        </AnimatedTaskLabel>
      )}
    </HStack>
  </SwipableView>
 )
}

export default TaskItem
```

Vamos continuar criando o componente **task-list.tsx**

./src/components/task-list.tsx

```
import React, { useCallback, useRef } from 'react'
import { AnimatePresence, View } from 'moti'
import {
  PanGestureHandlerProps,
  ScrollView
```

```
} from 'react-native-gesture-handler'
import TaskItem from './task-item'
import { makeStyledComponent } from '../utils/styled'

const StyledView = makeStyledComponent(View)
const StyledScrollView = makeStyledComponent(ScrollView)

interface TaskItemData {
  id: string
  subject: string
  done: boolean
}

interface TaskListProps {
  data: Array<TaskItemData>
  editingItemId: string | null
  onToggleItem: (item: TaskItemData) => void
  onChangeSubject: (item: TaskItemData, newSubject: string) => void
  onFinishEditing: (item: TaskItemData) => void
  onPressLabel: (item: TaskItemData) => void
  onRemoveItem: (item: TaskItemData) => void
}

interface TaskItemProps
  extends Pick<PanGestureHandlerProps, 'simultaneousHandlers'> {
  data: TaskItemData
```

```
  isEditing: boolean
  onToggleItem: (item: TaskItemData) => void
  onChangeSubject: (item: TaskItemData, newSubject: string) => void
  onFinishEditing: (item: TaskItemData) => void
  onPressLabel: (item: TaskItemData) => void
  onRemove: (item: TaskItemData) => void
}

export const AnimatedTaskItem = (props: TaskItemProps) => {
  const {
    simultaneousHandlers,
    data,
    isEditing,
    onToggleItem,
    onChangeSubject,
    onFinishEditing,
    onPressLabel,
    onRemove
  } = props
  const handleToggleCheckbox = useCallback(() => {
    onToggleItem(data)
  }, [data, onToggleItem])
  const handleChangeSubject = useCallback(
    subject => {
      onChangeSubject(data, subject)
    },
```

```
  [data, onChangeSubject]
)
const handleFinishEditing = useCallback(() => {
  onFinishEditing(data)
}, [data, onFinishEditing])
const handlePressLabel = useCallback(() => {
  onPressLabel(data)
}, [data, onPressLabel])
const handleRemove = useCallback(() => {
  onRemove(data)
}, [data, onRemove])
return (
  <StyledView
    w="full"
    from={{
      opacity: 0,
      scale: 0.5,
      marginBottom: -46
    }}
    animate={{
      opacity: 1,
      scale: 1,
      marginBottom: 0
    }}
    exit={{
      opacity: 0,
```

```
        scale: 0.5,
        marginBottom: -46
      }}
    >
      <TaskItem
        simultaneousHandlers={simultaneousHandlers}
        subject={data.subject}
        isDone={data.done}
        isEditing={isEditing}
        onToggleCheckbox={handleToggleCheckbox}
        onChangeSubject={handleChangeSubject}
        onFinishEditing={handleFinishEditing}
        onPressLabel={handlePressLabel}
        onRemove={handleRemove}
      />
    </StyledView>
  )
}

export default function TaskList(props: TaskListProps) {
  const {
    data,
    editingItemId,
    onToggleItem,
    onChangeSubject,
    onFinishEditing,
```

```
    onPressLabel,
    onRemoveItem
  } = props
  const refScrollView = useRef(null)

  return (
    <StyledScrollView ref={refScrollView} w="full">
      <AnimatePresence>
        {data.map(item => (
          <AnimatedTaskItem
            key={item.id}
            data={item}
            simultaneousHandlers={refScrollView}
            isEditing={item.id === editingItemId}
            onToggleItem={onToggleItem}
            onChangeSubject={onChangeSubject}
            onFinishEditing={onFinishEditing}
            onPressLabel={onPressLabel}
            onRemove={onRemoveItem}
          />
        ))}
      </AnimatePresence>
    </StyledScrollView>
  )
}
```

E por fim o último componente que precisamos para o nosso projeto **theme-toggle.tsx**

./src/components/theme-toggle.tsx

```
import React from 'react'
import { Text, HStack, Switch, useColorMode } from 'native-base'


export default function ThemeToggle() {
  const { colorMode, toggleColorMode } = useColorMode()
  return (
    <HStack space={2} alignItems="center">
      <Text>Escuro</Text>
      <Switch
        isChecked={colorMode === 'light'}
        onToggle={toggleColorMode}
      ></Switch>
      <Text>Claro</Text>
    </HStack>
  )
}
```

Com os componentes finalizados crie dentro da pasta src uma outra pasta chamada **screens,** iremos criar o arquivo que será responsável pela nossa página principal.

./src/screens/main-screen.tsx

```
import React, { useCallback, useState } from 'react'
import { Icon, VStack, useColorModeValue, Fab } from 'native-base'
import { AntDesign } from '@expo/vector-icons'
```

```
import AnimatedColorBox from '../components/animated-color-box'
import TaskList from '../components/task-list'
import shortid from 'shortid'
import Masthead from '../components/masthead'
import NavBar from '../components/navbar'

const initialData = [
  {
    id: shortid.generate(),
    subject: 'Comprar ingresso para o cinema',
    done: false
  },
  {
    id: shortid.generate(),
    subject: 'Estudar react native amanhã',
    done: false
  }
]

export default function MainScreen() {
  const [data, setData] = useState(initialData)
  const [editingItemId, setEditingItemId] = useState<string | null>(null)

  const handleToggleTaskItem = useCallback(item => {
    setData(prevData => {
      const newData = [...prevData]
```

```
    const index = prevData.indexOf(item)
    newData[index] = {
      ...item,
      done: !item.done
    }
    return newData
  })
}, [])
const handleChangeTaskItemSubject = useCallback((item, newSubject) => {
  setData(prevData => {
    const newData = [...prevData]
    const index = prevData.indexOf(item)
    newData[index] = {
      ...item,
      subject: newSubject
    }
    return newData
  })
}, [])
const handleFinishEditingTaskItem = useCallback(_item => {
  setEditingItemId(null)
}, [])
const handlePressTaskItemLabel = useCallback(item => {
  setEditingItemId(item.id)
}, [])
const handleRemoveItem = useCallback(item => {
```

```
  setData(prevData => {
    const newData = prevData.filter(i => i !== item)
    return newData
  })
}, [])

return (
  <AnimatedColorBox
    flex={1}
    bg={useColorModeValue('warmGray.50', 'primary.900')}
    w="full"
  >
    <Masthead
      title="E aí, Usuário!"
      image={require('../assets/masthead.png')}
    >
      <NavBar />
    </Masthead>
    <VStack
      flex={1}
      space={1}
      bg={useColorModeValue('warmGray.50', 'primary.900')}
      mt="-20px"
      borderTopLeftRadius="20px"
      borderTopRightRadius="20px"
      pt="20px"
```

```
>
  <TaskList
    data={data}
    onToggleItem={handleToggleTaskItem}
    onChangeSubject={handleChangeTaskItemSubject}
    onFinishEditing={handleFinishEditingTaskItem}
    onPressLabel={handlePressTaskItemLabel}
    onRemoveItem={handleRemoveItem}
    editingItemId={editingItemId}
  />
</VStack>
<Fab
  position="absolute"
  renderInPortal={false}
  size="sm"
  icon={<Icon color="white" as={<AntDesign name="plus" />} size="sm" />}
  colorScheme={useColorModeValue('blue', 'darkBlue')}
  bg={useColorModeValue('blue.500', 'blue.400')}
  onPress={() => {
    const id = shortid.generate()
    setData([
      {
        id,
        subject: '',
        done: false
      },
```

```
            ...data
          ])
          setEditingItemId(id)
        }}
      />
    </AnimatedColorBox>
  )
}
```

Agora dentro de src crie uma nova pasta chamada utils onde configuraremos nosso style.

./src/utils/styled.tsx

```
import React from 'react'
import { useStyledSystemPropsResolver } from 'native-base'

export const makeStyledComponent = (Comp: any) => {
  return React.forwardRef((props: any, ref: any) => {
    const [style, restProps] = useStyledSystemPropsResolver(props)
    return (
      <Comp {...restProps} style={style} ref={ref}>
        {props.children}
      </Comp>
    )
  })
}
```

./src/utils/use-previous.ts

```
import { useEffect, useRef } from 'react'


export default function usePrevious(value: any) {
  const ref = useRef()
  useEffect(() => {
    ref.current = value
  })
  return ref.current
}
```

E agora dentro de src crie o arquivo **index.tsx** e **theme.ts**

./src/utils/index.tsx

```
import React from 'react'
import { createDrawerNavigator } from '@react-navigation/drawer'
import MainScreen from './screens/main-screen'
import Sidebar from './components/sidebar'


const Drawer = createDrawerNavigator()


const App = () => {
  return (
    <Drawer.Navigator
      initialRouteName="Main"
      drawerContent={props => <Sidebar {...props} />}
```

```
      screenOptions={{
        headerShown: false,
        drawerType: 'back',
        overlayColor: '#00000000'
      }}
    >
      <Drawer.Screen name="Main" component={MainScreen} />
    </Drawer.Navigator>
  )
}

export default App
```

./src/utils/theme.ts

```
import { extendTheme } from 'native-base'

const config = {
  useSystemColorMode: false,
  initialColorMode: 'light'
}

const colors = {
  primary: {
    50: '#EEF2F6',
    100: '#CFD9E7',
    200: '#B1C1D8',
```

```
    300: '#92A9C9',
    400: '#7491B9',
    500: '#5578AA',
    600: '#446088',
    700: '#334866',
    800: '#223044',
    900: '#111822'
  }
}

export default extendTheme({ config, colors })
```

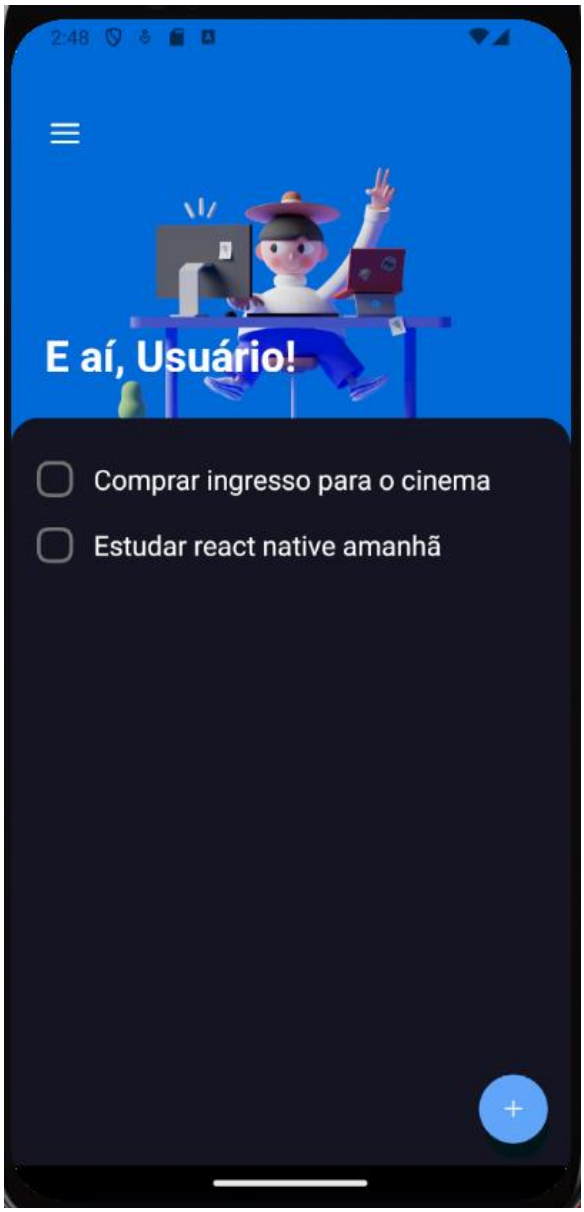
E agora para finalizar o nosso projeto vamos apenas configurar o arquivo **App.tsx** que fica na raiz do projeto para chamar o restante da nosso aplicação.

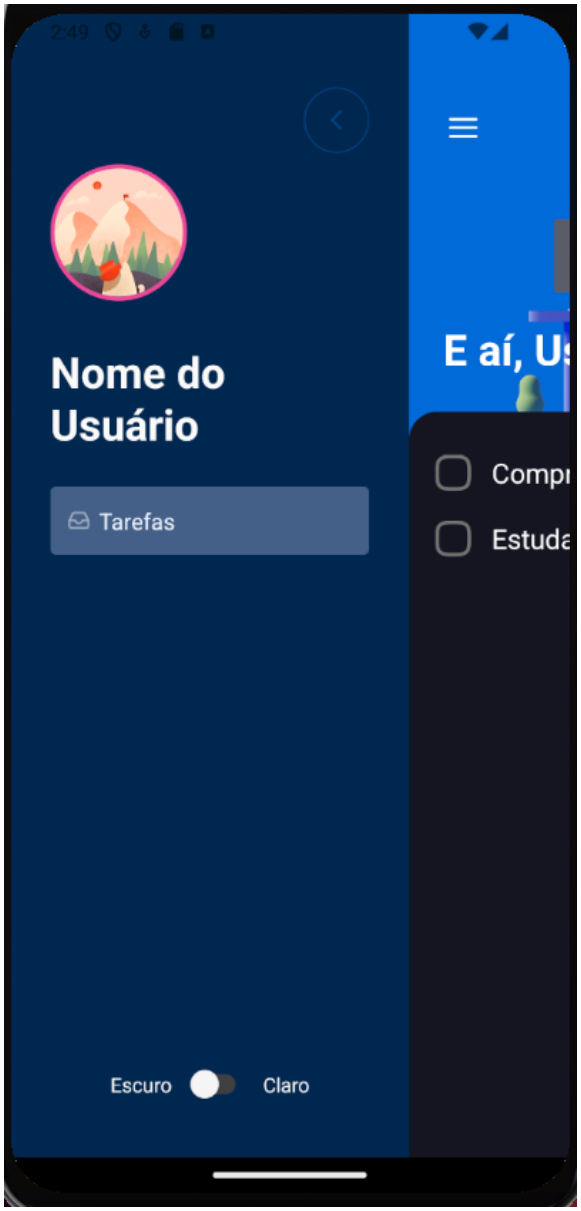
./App.tsx

```
import React from 'react'
import AppContainer from './src/components/app-container'
import Navigator from './src/'

export default function App() {
  return (
    <AppContainer>
      <Navigator />
    </AppContainer>
  )
}
```

Agora inicie o servidor com **npm start** para testar se sua aplicação está funcionando corretamente.




2:49
Nome do Usuário
Tarefas
Escuro
Claro
E aí, U
Compr
Estuda