PRIMEIRA AVALIAÇÃO PARCIAL – SEGUNDA CHAMADA - PIDM – 2019.2
Nome:

Usando a biblioteca "**stack**", "**switch**" e "**async-storage**" do react-native, implemente as seguintes telas:

0 – AuthLoadingScreen.js

- (0.5 pts) Essa tela deverá mostrar um <ActivityIndicator /> de carregamento. Ela verifica, em seu método **componentDidMount,** se existe algum usuário logado, ou seja, você deverá verificar o `AsyncStorage.getItem('logado')` retorna um objeto ou `null`. Caso seja **`null`**, você deverá encaminhar o usuário para **PILHA Login** (por isso o uso do navegador switch, veja última aula). Caso contrário, você deverá encaminhar o usuário para a **PILHA Aplicação.**

1 – CadastroScreen.js

- (1.5 pts) Nessa tela você deverá entrar com <login>, <senha> e <email>. Esses dados deverão ser **salvos** no `AsyncStorage.setItem("logado",<novo_objeto_json>)`. Infelizmente, <novo_objeto_json> só pode ser do tipo **String**! Então, você terá que transformar seu objeto JSON em um String. Como? Veja:

```
const logado = {'login':'aluno','senha':'ufc','email':'aluno@ufc.br'};
await AsyncStorage.setItem('logado', JSON.stringify(logado) );
```

  Ao clicar em **Enviar,** os dados do this.state deverão ser salvos AsyncStorage como no exemplo acima e o usuário deverá ser enviado para a tela de login (**LoginScreen.js**).

2 - LoginScreen.js

- (1.5 pts) O usuário deve entrar com o <login> e <senha>. Ao pressionar em **ENTRAR,** você deverá pegar o objeto salvo em **CadastroScreen.js** dentro do **AsyncStorage.** No entanto, você quer um JSON e o **AsyncStorage** só tem Strings! Como então retornar o valor do usuário logado como um objeto JSON? Veja:

```
const aluno_logado = await AsyncStorage.getItem('logado') //string!
//cuidado que  aluno_logado pode ser null (nenhum aluno cadastrado ainda).
//teste antes se é diferente de nulo para poder avançar...
let novoAluno = JSON.parse(aluno_logado); //agora é um objeto!
```

  Uma vez de posse do objeto `"novoAluno"`, você irá agora acessar o login e senha dele, comparando-os com <login> e <senha> digitados pelo usuário. Se estiver tudo certo, encaminhe o usuário para **PILHA aplicação**. Caso contrário, envie o usário para a tela **LoginErroScreen.js (MODAL).**

3 - LoginErroScreen.js

- (1.0 pts) Essa tela deve ser implementada como um **MODAL.** Ela irá exibir a mensagem substituindo o <login> pelo <login> dado como entrada na tela

LoginScreen.js. Ao pressionar **VOLTAR,** o usuário voltara para a tela **LoginScreen.js,** para poder tentar novamente o login e senha.

4 – LembrarSenhaScreen.js

- (0.5 pts)Essa tela recebe um e-mail como parâmetro, apenas, e não faz absolutamente nada. Ao pressionar em Enviar, o usuário deverá ser redirecionado para tela **LoginScreen.js**.

5 - MenuScreen.js

- (0.5 pts) Tem quatro botões: **IMC**, o qual leva para a tela IMCScreen.js e **HORÓSCOPO**, o qual leva para a tela HoroscopoScreen.js, **LOVE MONEY**, o qual leva para a tela LoveMoneyScreen.js e **SAIR** o qual **apaga** o objeto "`logado`" do AsyncStorage e redireciona para a página de login.

6 - IMCScreen.js e **ResultadoIMCScreen.js**

- (0.5 pts) Assim como nos exercícios anteriores, essa tela deverá ter dois <TextInput>: um para o <peso> e outro para a <altura>. Ao apertar em **ENVIAR,** os dados deverão ser enviados para a tela **ResultadoScreen.js,** o qual deverá ser implementada como uma **MODAL. IMPLEMENTE DE UMA FORMA QUE A PENAS O TECLADO NUMÉRICO SEJA EXIBIDO**

- (0.5 pts) Em **ResultadoScreen.js (MODAL),** você deverá calcular o IMC e mostrar a categoria de peso de acordo com o quadro (pressione **MENU** para voltar a tela **MenuScreen.js** ou **VOLTAR** para voltar para tela **IMCScreen.js)**:

Resultado	Situação
Abaixo de 17	Muito abaixo do *peso*
Entre 17 e 18,49	Abaixo do *peso*
Entre 18,5 e 24,99	*Peso* normal
Entre 25 e 29,99	Acima do *peso*
Entre 30 e 34,99	*Obesidade* I
Entre 35 e 39,99	*Obesidade* II (severa)
Acima de 40	*Obesidade* III (mórbida)

O aluno é livre para usar mais classes do que as apresentadas nesta prova para poder resolver o problema do IMC.

7 – HoróscopoScreen.js e **SorteScreen.js**

- (0.5 pts) Nessa tela, entre com <dia>, <mes> e <ano> de nascimento (ou outra data que você queira). Ao apertar em **ENVIAR,** redirecione os dados para a tela **SorteScreen.js.**
- (0.5 pts) Na tela **SorteScreen.js** você deverá exibir a mensagem de sorte de acordo com as seguintes regras (sinta-se livre para criar outras mensagens):

- Se o <dia> for PAR
  - Se o <mes> for PAR
    - Se o <ano> for PAR
      - ESCREVA **"MUITO DINHEIRO O ANO TODO."**
    - Se o <ano> for ÍMPAR
      - ESCREVA **"VAI ENCONTRAR A PESSOA AMADA"**
  - Se o <mes> for ÍMPAR
    - Se o <ano> for PAR
      - ESCREVA **"AQUELE SEU AMIGO VAI DAR UM CHOCOLATE"**
    - Se o <ano> for ÍMPAR
      - ESCREVA **"CUIDADO COM PROJETO INTEGRADO"**
- Se o <dia> for ÍMPAR
  - Se o <mes> for PAR
    - Se o <ano> for PAR
      - ESCREVA **"A LISEIRA VAI SER GRANDE."**
    - Se o <ano> for ÍMPAR
      - ESCREVA **"VÃO LHE EMPRESTAR A SENHA DO NETFLIX"**
  - Se o <mes> for ÍMPAR
    - Se o <ano> for PAR
      - ESCREVA **"DOR DE BARRIGA EMINENTE!"**
    - Se o <ano> for ÍMPAR
      - ESCREVA **"ESTUDE UM POUCO MAIS."**

Pressione **MENU** para voltar a tela **MenuScreen.js** ou **VOLTAR** para voltar para tela **HoróscopoScreen.js). SorteScreen.js deverá ser implementada como uma MODAL.**

8 – (1.5 pts) **LoveMoneyScreen.js** e **ContaScreen.js**

- Nessa tela, o usuário entrar com 3 valores: <Salário A>, <Salário B> e <Conta>. A ideia é que num jantar a dois, cada um pague a conta proporcionalmente ao seu salário. Por exemplo
  - Se A ganha 2000.00 por mês e B ganha 3000.00 por mês, então B deve pagar 60% da conta (60% = (2000+3000)/5000) e A deve pagar 40% da conta. Ao pressionar em **Enviar,** esse calculo deverá ser feito e os resultado enviados para **ContaScreen.js.**
  - **ContaScreen.js** deve ser implementado como um MODAL. Pressione **MENU** para voltar a tela **MenuScreen.js** ou **VOLTAR** para voltar para tela **LoveMoneyScreen.js**

9 - (0.5 pts) **Use estilos**: use algum estilo em suas telas. Coloque Títulos nos cabeçalhos (headers) de cada tela e aplique estilos lá também. Dica: crie um arquivo separado.

Snack: caso não tenha o ambiente ainda instalado na sua máquina use o site do snack (https://snack.expo.io/) e teste sua aplicação no seu dispositivo móvel lendo o QR Code (necessita baixar o Expo).

ATENÇÃO 1: NÃO serão tiradas dúvidas referentes a instalação, bibliotecas, código, interfaces de desenvolvimentos, erros de compilação/execução, etc. As dúvidas permitidas são apenas referentes **À PROVA (apenas após 30 minutos do início)**.

ATENÇÃO 2: a prova deve ser entregue pelo SIPPA **(AP1)** até as 19:30 de 03/10. **NÃO ENVIE A PASTA node_modules**. Compacte o seu projeto. O seu código **DEVE** ter algo executando.

0	1	2	3	4	5	6	7	8	9
0.5	1.5	1.5	1.0	0.5	0.5	0.5 0.5	0.5 0.5	2.0	0.5

Fluxo de Telas: