Departamento de Computação
Uni-FACEF

Desenvolvimento de Software para Dispositivos Móveis II
Prof. Fausto G. Cintra
faustocintra@facef.br

Aula 06

📚 Formulários e entrada de dados no React Native (Expo)

1. Objetivos de aprendizagem

- Explorar `TextInput` de forma avançada.
- Usar `useState` e `useEffect` para tratar dados em tempo real.
- Aplicar **validações** e fornecer feedback ao usuário.
- Criar um formulário de **cadastro/*login*** funcional.

2. Criando e configurando um novo projeto para praticar

No terminal do VS Code, execute:

```
npx create-expo-app --template blank Aula06
```

Em seguida:

```
cd Aula06
```

Por fim, vamos executar o projeto:

```
npx expo start --tunnel
```

👉 Caso prefira visualizar seu projeto em modo Web, tecle `w` e instale os pacotes necessários.

3. Diferença entre entrada de dados na Web e no *mobile*

- Na Web → `<input>` é padrão, com eventos `onChange`, `onSubmit`, etc.
- No *mobile* (React Native) → usamos **`TextInput`**, com eventos adaptados para teclado virtual.
- Além disso, podemos controlar o **tipo de teclado**, algo inexistente no navegador (`numeric`, `email-address`, `phone-pad`).

4. Uso avançado do `TextInput`

📄 Exemplo com múltiplas propriedades:

```
<TextInput
  style={styles.input}
  placeholder="Digite seu e-mail"
  keyboardType="email-address"
  autoCapitalize="none"
  autoCorrect={false}
  value={email}
  onChangeText={setEmail}
  onSubmitEditing={() => console.log("Enviado!")}
/>
```

Destaques:

- `autoCapitalize="none"` → impede que o campo `email` comece com maiúscula.
- `autoCorrect={false}` → evita correções automáticas irritantes em *login*/senha.
- `onSubmitEditing` → executa ação quando o usuário pressiona "Enter" no teclado virtual.

5. Validação de formulários

📄 Exemplo simples:

```
import { useState } from 'react';
import { View, Text, TextInput, Button, StyleSheet } from 'react-native';

export default function CadastroScreen() {
  const [email, setEmail] = useState('');
  const [senha, setSenha] = useState('');
  const [erro, setErro] = useState('');

  const handleCadastro = () => {
    if (!email.includes('@')) {
      setErro('Digite um e-mail válido');
      return;
    }
    if (senha.length < 6) {
      setErro('A senha deve ter no mínimo 6 caracteres');
      return;
    }
    setErro('');
    alert('Cadastro realizado com sucesso!');
  };

  return (
    <View style={styles.container}>
      <Text style={styles.titulo}>Cadastro</Text>
```

```
      <TextInput
        style={styles.input}
        placeholder="E-mail"
        keyboardType="email-address"
        autoCapitalize="none"
        value={email}
        onChangeText={setEmail}
      />

      <TextInput
        style={styles.input}
        placeholder="Senha"
        secureTextEntry
        value={senha}
        onChangeText={setSenha}
      />

      {erro ? <Text style={styles.erro}>{erro}</Text> : null}

      <Button title="Cadastrar" onPress={handleCadastro} />
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', padding: 20 },
  titulo: { fontSize: 24, fontWeight: 'bold', marginBottom: 20, textAlign:
'center' },
  input: { borderWidth: 1, borderColor: '#ccc', marginBottom: 15, padding:
10, borderRadius: 8 },
  erro: { color: 'red', marginBottom: 10 }
});
```

6. Experiência de usuário (UX) em *mobile*

- Controlar o tipo de teclado (`numeric`, `phone-pad`, etc.) para facilitar a digitação.
- Mostrar mensagens de erro claras e próximas ao campo.
- Usar `secureTextEntry` para senhas.
- Dar *feedback* rápido (ex.: mostrar *loading* após clicar em "Entrar").

7. 🕹️ Exercício guiado

Criar uma tela de ***login*** que:

- Tenha dois campos: *e-mail* e senha.
- Valide o formato do *e-mail* e o tamanho da senha.
- Mostre erros diretamente abaixo dos campos.
- Ao logar com sucesso, navegue para uma outra tela com mensagem de boas-

vindas.

8. 👩‍💻 Desafio final

Criar um **formulário de cadastro mais completo**, com os campos:

- Nome completo
- *E-mail*
- Senha
- Confirmar senha
- Telefone

🔹 Regras:

1. O telefone deve aceitar apenas números (usar `keyboardType="numeric"`).
2. A senha e a confirmação devem ser iguais.
3. Mostrar erros específicos abaixo de cada campo inválido.
4. Ao finalizar, exibir os dados formatados em uma nova tela (pesquise como passar dados de uma tela a outra usando `navigation.navigate()`).