Modificações no projeto caqp.dev.br

A implementação de outros tipos de devices

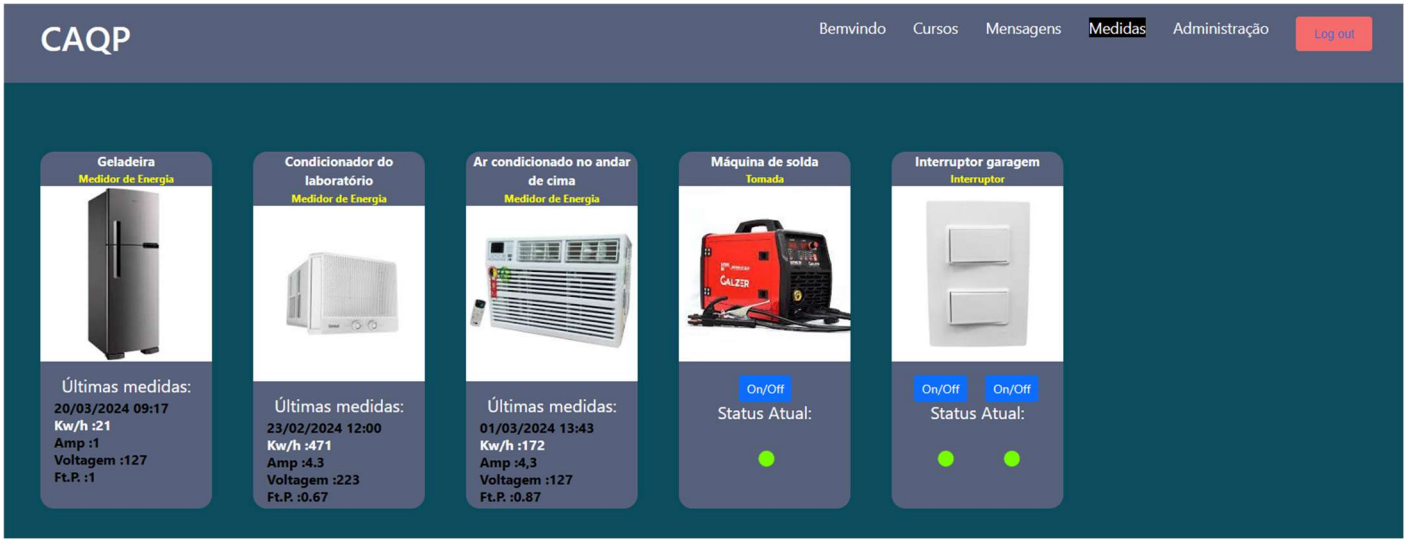
O fluxo do sistema se inicia com a criação de um device. Neste ponto o novo device recebe nome, email, descrição e uma foto que ilustra o eletrodoméstico em que ele está conectado, pois, este único device, é um medidor de energia. Para implementar devices diferentes deste tipo será necessário alterar a etapa de cadastro dos novos devices:

```
const NovoDevice = ()=>{
    const [nome, setNome] = useState()
    const [descricao, setDescricao] = useState()
    const [email, setEmail] = useState()
    const [imagem, setImagem] = useState()
    const [tipo, setTipo] = useState()

    const data = {nome, descricao, email, imagem, tipo}

    function click (){
        addRegistro(data)
    }
```

Além dos tópicos já existentes, incluímos o tipo do device.
Para o tipo um iremos ter um medidor de energia.
Para o tipo dois um elemento interruptor com uma seção.
Para o tipo três um elemento interruptor com duas seções
Para o tipo quatro um elemento interruptor com três seções
Para o tipo cinco um elemento também liga desliga com controle para altas correntes.

```
    return (
        <Container>
            <StyleForm>
                <h5>Formulário para cadastro de um novo device. Todos os campos são necessários.</h5>
                <Form>
                <Form.Group className="mb-3" controlId="formBasicTipo">
                        <Form.Label><h6>Tipo de device</h6></Form.Label>
                            <Form.Control
                                type="text"
                                value={tipo}
                                placeholder="Tipo de device - 1, 2 ou 3"
                                onChange={(e)=> setTipo(e.target.value)}
                            />
                    </Form.Group>
```

Na parte JSX incluímos no formulário o tipo do dispositivo. Inicialmente estamos propondo os tipos 1 2 e 3.
1 – Medidor de energia
2 – Interruptor com 2 seções
3 – Interruptor para corrente elevada



Uma vez que um novo device passa por etapa, é preciso prever para ele dados na base MongoDB ou ter em mãos o device físico preparado para enviar dados a está mesma base de dados.
O problema usar a opção Medidas sem que este novo devices tenha dados cadastradas no array medidas permanece. Isso irá provocar uma tela em branco na seção Medidas e um erro no inspecionar.
Este problema terá uma solução em futura oportunidade.
A página Medidas não foi alterada e dentro do JSX dela temos um map para a leitura da base de dados sendo passada para o componente Cards:

```
    return(
        <Listadevices>
            <Listacards>

                {data?.message?.map(projeto =>{
                    return(<Card key={projeto._id} projeto={projeto}/>)
                })}
            </Listacards>
        </Listadevices>
    )
}
export default Medidas
```

Já no componente temos algumas alterações:

```
72  const Card = ({projeto})=>{
73      var ultimo = projeto.medidas.length
74
75      let posicao
76      let posicao1
77      let statusTomada
78      let data
79      let hora
80
81      const [OnOff, setOnOff] = React.useState()
82      const [OnOff1, setOnOff1] = React.useState()
83      const [OnOffTomada, setOnOffTomada] = React.useState()
84      const [datalocal, setDataLocal] = useState()
85      const [horalocal, setHoraLocal] = useState()
```

Montamos um esquema com useState para alocar algumas variáveis que iremos usar.

```
86
87      function Inicio(){
88          setOnOff(projeto.medidas[ultimo-1].posicao)
89          setOnOff1(projeto.medidas[ultimo-1].posicao1)
90          setOnOffTomada(projeto.medidas[ultimo-1].statusTomada)
91          setDataLocal(projeto.medidas[ultimo-1].data)
92          setHoraLocal(projeto.medidas[ultimo-1].hora)
93      }
94
95      const Interruptor = ()=>{
96          console.log("OnOff antes", OnOff)
97          let inv = !OnOff
98          setOnOff(inv)
99          console.log("OnOff depois", inv)
100         posicao = inv
101         posicao1 = OnOff1
102         data = datalocal.toString()
103         hora = horalocal.toString()
104         let id = projeto._id
105         putmedidas(id, {medidas: {data, hora, posicao, posicao1}})
106     }
```

A seguir uma função se encarrega de ajustar as variáveis para que elas assumam os últimos valores obtidos da leitura da base de dados. Estes valores estão condicionados ao id do elemento que foi clickado na tela dos Cards. Por exemplo, se foi clickado e elemento máquina de solda, os valores serão os correspondentes a este elemento, vez que a seleção de qual elemento foi clickado é feita com base no id deste elemento.
Na figura acima, estamos mostrando o processamento para o click no elemento interruptor duplo.
As linhas 96 e 99 são apenas para fins de debug e podem ser retiradas.
A linha 97 cria uma variável, inv, que vai receber o valor presente em OnOff invertido. Este é um valor booleano.
Lembre-se que este valor foi ajustado para o último valor presente na base de dados quando a página foi carregada.

O trecho denominado Inicio será executado uma única vez sempre que a página for carregada e isso fará os ajustes daquelas variáveis para os valores mais recentes existentes na base de dados.

A seguir, valor de inv é setado na variável OnOff. Este valor vai ajustar a cor dos LEDs presentes no Cards:

```
161             {projeto.tipo === 2 &&
162                 <div>
163                     <Titulo1>{projeto.nome}</Titulo1>
164                     <Titulo>Interruptor</Titulo>
165                     <Link to={`/medidas/${projeto._id}`}>
166                         <img src= {projeto.imagem} alt="Description"/>
167                     </Link>      
168                     <button type="button" class="btn btn-primary btn-sm" onClick={Interruptor}>On/Off</button>
169                          
170                     <button type="button" class="btn btn-primary btn-sm" onClick={Interruptor1}>On/Off</button>
171                     <br/>
172                     <h5>Status Atual:</h5>
173                     <br/>
174                                   
175                     {OnOff === true ? (<img src='circuloVerde_resized.png' thumbnail alt='imagem'></img>) : (<img src='circuloVermelho_resized.png' thumbnail alt='imagem'></img>)}
176                                    
177                     {OnOff1 === true ? (<img src='circuloVerde_resized.png' thumbnail alt='imagem'></img>) : (<img src='circuloVermelho_resized.png' thumbnail alt='imagem'></img>)}
178
179                 </div>
180             }
```

Este é um trecho do JSX de Cards que monta os Cards na tela.
Na linha 161 estamos selecionando os devices do tipo 2. Antes o Card inteiro era clickavel, mas, agora somente a foto do device é usada para iniciar a visualização da tabela de detalhes.
Logo abaixo da tela colocamos dois botões, um para cada interruptor e para cada botão um LED.
Isso pode ser visto nas linhas 168 a 170.

Note que na linha 168 o evento de click chama a função Interruptor:

```
95    const Interruptor = ()=>{
96        console.log("OnOff antes", OnOff)
97        let inv = !OnOff
98        setOnOff(inv)
99        console.log("OnOff depois", inv)
100       posicao = inv
101       posicao1 = OnOff1
102       data = datalocal.toString()
103       hora = horalocal.toString()
104       let id = projeto._id
105       putmedidas(id, {medidas: {data, hora, posicao, posicao1}})
106   }
```

Essa função, que estávamos examinando acima, irá inverter o estado da variável boolena OnOff via uma outra variável inv. A seguir o conjunto OnOff e OnOff1, que representam os interruptores, a OnOff já com seu valor invertido, junto com a data e hora que serão passadas junto a partir do ESP32 presente no device, serão enviadas para a base de dados, linha 105.
Assim, o envio pode ser feito pressionando o botão na tela ou via o ESP32 no device.
Para que o envio com o uso do botão na tela possa passar data e hora corretamente, deveremos ajustar este valores com a data obtido externamente, via Date.now(), por exemplo.
Para o envio via ESP32 utilizamos o recurso de uma biblioteca que retorna a data e a hora.

O mesmo irá acontecer para o segundo interruptor e para o interruptor para correntes altas:

```
108   const Interruptor1 = ()=>{
109       console.log("OnOff1 antes", OnOff)
110       let inv = !OnOff1
111       setOnOff1(inv)
112       console.log("OnOff1 depois", inv)
113       posicao = OnOff
114       posicao1 = inv
115       data = datalocal.toString()
116       hora = horalocal.toString()
117       let id = projeto._id
118       putmedidas(id, {medidas: {data, hora, posicao, posicao1}})
119
120   }
121
122   const InterruptorTomada = ()=>{
123       console.log("OnOffTomada antes", OnOffTomada)
124       let inv = !OnOffTomada
125       setOnOffTomada(inv)
126       console.log("OnOffTomada depois", inv)
127       statusTomada = inv
128       data = datalocal.toString()
129       hora = horalocal.toString()
130       let id = projeto._id
131       putmedidas(id, {medidas: {data, hora, statusTomada}})
132
133   }
```

No início desta análise, foi dito que a função Inicio será executada uma única vez quando a página é carregada. Isso é feito via um useEffect:

```
135   useEffect(()=>{
136       Inicio()
137   },[])
```

O useEffect usado desta forma, será executa apenas uma vez quando a página for carregada e nesta única vez ela irá chamar a função Inicio.
O JSX para a tomada de alta corrente com interruptor é mostrado a seguir:

```
{projeto.tipo === 3 &&
    <div>
        <Titulo1>{projeto.nome}</Titulo1>
        <Titulo>Tomada</Titulo>
        <Link to={`/medidas/${projeto._id}`}>
            <img src= {projeto.imagem} alt="Description"/>
        </Link>                
        <button type="button" class="btn btn-primary btn-sm" onClick={InterruptorTomada}>On/Off</button>
        <h5>Status Atual:</h5>
        <br/>
                                
        {OnOffTomada ? (<img src='circuloVerde_resized.png' thumbnail alt='imagem'></img>) : (<img src='circuloVermelho_resized.png' thumbnail alt='imagem'></img>)}
    </div>
}
</Content>
</div>
</StyledCard>
```

Finalmente, para que tudo funcione, precisamos modificar o componente DetalhesDevices:

```
return(
    <Container>
        <Titulo><h4>{data?.message?.nome}</h4></Titulo>
        <Table striped bordered hover variant="dark">
            {data?.message?.tipo === 1 &&
                <thead>
                    <tr>
                        <th>Kw/H</th>
                        <th>Corrente</th>
                        <th>Voltagem</th>
                        <th>Fator de Potência</th>
                        <th>Data</th>
                        <th>Hora</th>
                    </tr>
                </thead>
            }
            {data?.message?.tipo === 1 &&
                <tbody>
                    {data?.message?.medidas?.map(med =>{
                        return(<tr>
                            <td style={med.kwh > 1500 ? {color:"red"}:{color:"white"}}>{med.kwh}</td>
                            <td>{med.corrente}</td>
                            <td>{med.voltagem}</td>
                            <td>{med.fp}</td>
                            <td>{med.data}</td>
                            <td>{med.hora}</td>
                        </tr>)
                    })}
                </tbody>
            }
```

De novo, estamos, na linha 22 verificando se o device cuja foto recebeu clcik, é do tipo 1.
Foi necessário colocar a verificação para os rótulos e também para os valores para obedecer as regras de sintax do JSX.

Caso o device que recebeu click seja do tipo 2:

```
48
49              {data?.message?.tipo === 2 &&
50                  <thead>
51                      <tr>
52                          <th>Data</th>
53                          <th>Hora</th>
54                          <th>Interruptor 1</th>
55                          <th>Interruptor 2</th>
56                          <th>Interruptor 3</th>
57                      </tr>
58                  </thead>
59              }
60              {data?.message?.tipo === 2 &&
61                  <tbody>
62                      {data?.message?.medidas?.map(med =>{
63                          return(<tr>
64                              <td>{med.data}</td>
65                              <td>{med.hora}</td>
66                              {med.posicao === true &&
67                                  <td>Ligada</td>
68                              }
69                              {med.posicao === false &&
70                                  <td>Desligada</td>
71                              }
72                              {med.posicao1 === true &&
73                                  <td>Ligada</td>
74                              }
75                              {med.posicao1 === false &&
76                                  <td>Desligada</td>
77                              }
78                          </tr>)
79                      })}
80                  </tbody>
81              }
```

E caso seja do tipo 3:

```
83                  {data?.message?.tipo === 3 &&
84                      <thead>
85                          <tr>
86                              <th>Data</th>
87                              <th>Hora</th>
88                              <th>Interruptor</th>
89                          </tr>
90                      </thead>
91                  }
92                  {data?.message?.tipo === 3 &&
93                      <tbody>
94                          {data?.message?.medidas?.map(med =>{
95                              return(<tr>
96                                  <td>{med.data}</td>
97                                  <td>{med.hora}'</td>
98                                  {med.statusTomada === true &&
99                                      <td>Ligada</td>
100                                 }
101                                 {med.statusTomada === false &&
102                                     <td>Desligada</td>
103                                 }
104                             </tr>)
105                         })}
106                     </tbody>
107                 }
108             </Table>
109         </Container>
110     )
111 }
```

Tanto para o tipo 2 como para o tipo 3 foi necessário montar a estrutura para converter os níveis booleanos em palavras que representem o seu estado.