01

Finalizando o projeto

Transcrição

Chegamos ao último capítulo do nosso treinamento de Electron. Aqui, iremos "embrulhar" a aplicação para realizarmos o *deploy* para os três sistemas operacionais, e resolveremos alguns *bugs* da aplicação, foco deste vídeo.

Evitando que um curso em branco seja adicionado

Um dos *bugs* da nossa aplicação é a possibilidade de podermos criar cursos em branco! Na aplicação, podemos simplesmente clicar no botão de adição de cursos, que um curso em branco é adicionado. Além disso, um item "fantasma" é criado no nosso menu.

Para resolver isso, no **renderer.js**, quando o campo estiver vazio, nós não devemos adicionar o curso. Então, se o campo estiver vazio, vamos imprimir uma mensagem no console e sair da função, através do `return`:

```
// renderer.js

// restante do código omitido

botaoAdicionar.addEventListener('click', function() {

    // verificando se o campo está em branco
    if(campoAdicionar.value == ''){
        console.log('Não posso adicionar um curso com nome vazio');
        return;
    }

    let novoCurso = campoAdicionar.value;
    curso.textContent = novoCurso;
    tempo.textContent = '00:00:00';
    campoAdicionar.value = '';
    ipcRenderer.send('curso-adicionado', novoCurso);
});
```

Com isso, eliminamos esse primeiro *bug*, já que agora um curso em branco não pode ser mais adicionado.

Curso sem arquivo JSON com dados de outro curso

Outro *bug* da aplicação é que, ao adicionar um novo curso e imediatamente trocar por um curso já existente, se estudarmos um pouco, ou seja, apertarmos o botão de *play* e depois o de *stop*, e trocarmos de volta para o curso criado inicialmente, o tempo desse curso não estará zerado. Por exemplo: adicionamos o curso de Photoshop, e trocamos para o curso de JavaScript, estudamos um pouco, e voltamos para o curso de Photoshop, ele não terá o seu tempo zerado e sim igual ao tempo do curso de JavaScript.

Isso acontece pois como não começamos o curso novo (apertamos o botão de *play*), com isso o arquivo JSON dele não foi criado! Logo, não há nada para carregar no *timer* e ele fica com os dados carregados do curso anterior. Isso até gera um erro, que podemos ver no console.

Então, para corrigir isso, na hora de carregar os dados do curso, se houver um erro, nós imprimimos uma mensagem e zeramos o *timer*

```
// renderer.js

// restante do código omitido

ipcRenderer.on('curso-trocado', (event,nomeCurso) => {
    data.pegaDados(nomeCurso)
        .then((dados) => {
            tempo.textContent = dados.tempo;
        }).catch((err) => {
            console.log('O curso ainda não possui um JSON');
            tempo.textContent = "00:00:00";
        })
    curso.textContent = nomeCurso;
});
```

Com isso, corrigimos mais um *bug*.

Parando o timer antes de trocar o curso

Mais *bug* que veremos, é quando trocamos de curso com o *timer* rodando, sem apertar o botão de *stop* antes da troca. Quando isso acontece, o tempo que está rodando continua sendo do curso inicial.

O que podemos fazer é parar o *timer* antes que o curso seja trocado:

```
// renderer.js

// restante do código omitido

ipcRenderer.on('curso-trocado', (event,nomeCurso) => {
    // parando o timer antes de carregar os dados do curso
    timer.parar(curso.textContent);
    data.pegaDados(nomeCurso)
        .then((dados) => {
            tempo.textContent = dados.tempo;
        }).catch((err) => {
            console.log('O curso ainda não possui um JSON');
            tempo.textContent = "00:00:00";
        })
    curso.textContent = nomeCurso;
});
```

Agora, quando trocamos o curso, paramos o *timer* do curso inicial, nos precavendo caso o usuário esqueça de pará-lo.

Prevenindo o valor undefined para segundos

Por último, com essa alteração que fizemos anteriormente, quando adicionamos um curso, não o inicializamos, e trocamos de curso, é impresso um erro no console.

O que acontece é que estamos chamando a função `parar` no momento em que trocamos o curso, e essa função passa a variável `segundos` para outra função, a `segundosParaTempo`, que faz algo com essa variável. Mas como o *timer* não foi inicializado, `segundos` terá o valor `undefined`, e é aí que acontece o erro.

Para evitar esse erro, vamos iniciar a variável `segundos` com o valor `0`:

```
// timer.js

const { ipcRenderer } = require('electron');
const moment = require('moment');
// inicializando a variável segundos com o valor 0
let segundos = 0;
let timer;
let tempo;

module.exports = {
    // código do módulo omitido
}
```

Com isso, prevenimos que o valor da variável `segundos` seja `undefined`.

Com a nossa aplicação um pouco mais redonda, vamos continuar com o objetivo de realizar o *deploy* para os três sistemas operacionais.