05

Novas funcionalidades no Menu

Transcrição

Vamos agora organizar o código responsável de criar o Menu para que ele não fique todo na função principal de nosso programa.

O primeiro passo é criar a função `geraMenuPrincipalTemplate()`, em nosso `template.js`, que será quem é capaz de retornar o template do Menu de nossa aplicação. O nosso menu inicial terá o menu de sobre, que deve exibir a janela de sobre do nosso sistema.

```
//template.js

module.exports = {
    ...
    geraMenuPrincipalTemplate(app){
        let templateMenu = [
            {
                label: 'Sobre',
                submenu: [
                    {
                        label: 'Sobre o Alura Timer'
                    }
            }
        ];
        templateMenu.unshift(
        if( process.platform == 'darwin'){
                {
                    label: app.getName(),
                    submenu: [
                        {
                            label: 'Estou rodando no Mac!'
                        }
                    ]
            })
         }
         return templateMenu;
    }
}
```

Repare que como fazemos o uso do `app`, precisamos recebê-lo como parâmetro.

Em seguida, vamos no `main.js` chamar a nossa função:

```
//main.js
app.on('ready', () => {
    ...

    let templateMenu = templateGenerator.geraMenuPrincipalTemplate(app);
    let menuPrincipal = Menu.buildFromTemplate(templateMenu);
    Menu.setApplicationMenu(menuPrincipal);
```

```
    ...
});
```

Quando carregarmos a aplicação o novo Menu deve surgir lá perfeitamente, agora só falta conseguirmos detectar um clique no "Sobre o Alura Timer" , e que este clique execute a função que abra a janela de sobre.

Para detectar um click é simples, basta adicionarmos a propriedade click no objeto construtor do nosso Menu:

```
//template.js
const { ipcMain } = require('electron');

module.exports = {
    ...
    geraMenuPrincipalTemplate(app){
        let templateMenu = [
            {
                label: 'Sobre',
                submenu: [
                    {
                        label: 'Sobre o Alura Timer',
                        click: () => {
                    //Detecção de clique!
                        }
                    }
                ]
            }
        ];

        ...

        return templateMenu;
    }
}
```

Agora para fazermos com que este click abra a janela de sobre, precisamos que aquele evento que foi criado anteriormente seja executado, o evento de `abrir-janela-sobre` , que está no `main.js` .

Até agora tínhamos visto como emitir eventos do Processo Principal para o Processo de Renderer, e vice versa. Mas agora precisamos de algo diferente, o `template.js` faz parte do processo principal, e o listener do evento `abrir-janela-sobre` também, ou seja, precisamos emitir um evento no Processo Principal **para** o Processo Principal.

Como o `ipcMain` é uma extensão de um módulo importante do Node.js que é o `EventEmitter` ,podemos alcançar isto através da função `.emit()` , que emite um determinado evento no próprio `ipcMain` . Então fazendo como abaixo:

```
//template.js
const { ipcMain } = require('electron');

module.exports = {
    ...
    geraMenuPrincipalTemplate(app){
        let templateMenu = [
            {
```

```
                label: 'Sobre',
                submenu: [
                    {
                        label: 'Sobre o Alura Timer',
                        click: () => {
                            ipcMain.emit('abrir-janela-sobre');
                        }
                    }
                ]
            }
        ];

        ...

        return templateMenu;
    }
}
//Não esqueça de importar o ipcMain!
```

Conseguimos com que o nosso evento seja emitido no processo principal e ao testar, vemos que a janela de sobre é aberta!

Adicionando controles padrões em nossos Menus

Quando criamos um aplicativo padrão do Electron, vimos que ele implementa algumas funcionalidades comuns para gente, como : *Maximizar, Minimizar, Copiar, Colar...* ; Estas funções básicas já tem comportamentos definidos pelo Sistema Operacional e não precisamos reimplementá-las caso queiramos utilizá-las em nosso projeto. Podemos apenas criar um novo menu e falar que o **role** dele é uma destas funcionalidades. Na documentação (https://electron.atom.io/docs/api/menu/#examples) dos Menus do Electron, podemos ver que é muito fácil utilizar estes roles prontos, basta apenas inserir um novo sub menu com as funcionalidades desejadas.

Vamos então adicionar novos submenus de *View* e *Window* em nosso projeto, com as funções de controlar a janela e poder voltar a exibir o Dev Tools. Olhando na documentação conseguimos extrair os roles destas funcionalidades e adicionar ao Menu já criado:

```
//template.js
...
    geraMenuPrincipalTemplate(app){
        let templateMenu = [
            {
                label: 'View',
                submenu: [{
                    role: 'reload'
                },
                {
                    role: 'toggledevtools'
                  }
                ]
            },
            {
                label: 'Window',
                submenu: [
                    {
                        role: 'minimize'
```

```
                    },
                    {
                        role: 'close'
                    }
                ]
            },
            {
                label: 'Sobre',
                submenu: [
                    {
                        label: 'Sobre o Alura Timer',
                        click: () => {
                            ipcMain.emit('abrir-janela-sobre');
                        }
                    }
                ]
            }
        ];
        if( process.platform == 'darwin'){
            templateMenu.unshift({
                label: app.getName(),
                submenu: [
                    {
                        label: 'Estou rodando no Mac!'
                    }
                ]
            })
         }
         return templateMenu;
    }
...
```

Agora você pode testar a funcionalidade de Minimizar ou Dev Tools pelo Menu da aplicação, que foi implementada sem uma linha de código nossa! Bem prático, não ?