01

Criando a janela sobre

Transcrição

Neste capítulo, entenderemos um pouco melhor sobre as diferenças de processos e como eles se comunicam. Veremos mais sobre as diferenças do processo `main`, que é o processo principal, que controla o ciclo de vida da aplicação, capaz de criar janelas; e do processo de `renderer`, que é a nossa janela, o visual que o usuário interage.

Veremos também como algo que acontece no `renderer` pode ser comunicar com algo do `main`, por exemplo. Só conseguimos utilizar o `BrowserWindow` dentro o **main.js**, e o processo de `renderer` é o único que consegue criar janelas de navegação entre os sistemas, como as janelas de busca de arquivo. Então, como do processo `main`, pedimos uma janela de buscar arquivo, ou do processo de `renderer`, pedimos para criar uma nova janela?

É isso que veremos nesse capítulo.

Comunicação entre processos

Para vermos como fazemos isso, vamos criar uma janela de **Sobre** no nosso projeto, que é uma janela que explica resumidamente o que é o software.

Na janela inicial da aplicação, o usuário irá clicar em um link, que irá abrir a janela de **Sobre**. Então, no **index.html**, vamos adicionar um link, logo abaixo do `h1`:

```
<!-- index.html -->
<a href="#" id="link-sobre">Sobre</a>
```

Agora, ao clicar nesse link, uma nova janela é aberta com informações da nossa aplicação, como sua versão, autor, etc. Então, se queremos detectar um evento de `click`, devemos fazer isso no JavaScript, mais precisamente no **renderer.js**, que ficará escutando o evento:

```
// renderer.js

let linkSobre = document.querySelector('#link-sobre');

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

});
```

Ao clicar no link, queremos abrir uma janela, mas sabemos que o único responsável por criar janelas é o **main.js**, que permite usar `BrowserWindow`, que efetivamente criará a janela. O `BrowserWindow` é uma das características que só podem ser feitas no **processo principal**, então não conseguimos utilizá-lo no **renderer.js**, já que só o processo principal consegue criar novas janelas.

Então, no `renderer`, temos que **pedir** para o processo principal criar uma nova janela. Fazemos essa comunicação entre o processos utilizando o **IPC** (*Inter-Process Communication*).

O IPC é uma técnica de comunicação de processos, que provavelmente o nosso sistema operacional usa, mas que aqui vamos utilizar sua biblioteca do Electron. Essa biblioteca de IPC do Electron fará com que o processo `renderer` consiga se comunicar com o processo principal, e vice-versa.

Para utilizar essa biblioteca do Electron, fazemos o mesmo que fizemos quando queríamos usar o sub-módulo de `app` ou de `BrowserWindow`, ou seja, importamos de `electron` o `ipcRenderer`:

```
// renderer.js

const { ipcRenderer } = require('electron');

let linkSobre = document.querySelector('#link-sobre');

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

});
```

É o `ipcRenderer` que iremos utilizar para realizar a comunicação com o processo principal, pedindo para criar uma nova janela.

Utilizando o ipcRenderer

O `ipcRenderer` se comunica através de eventos, ou seja, ele envia um evento para o processo principal, que por sua vez fica escutando esse evento. Então, quando o link Sobre for clicado, enviamos um evento através da função `send`. Chamaremos o evento de `abrir-janela-sobre`:

```
// renderer.js

const { ipcRenderer } = require('electron');

let linkSobre = document.querySelector('#link-sobre');

linkSobre.addEventListener('click', function() {
    ipcRenderer.send('abrir-janela-sobre');
});
```

Agora, no processo principal, precisamos escutar o evento `abrir-janela-sobre`. Se temos o `ipcRenderer` para enviar o evento, temos o `ipcMain` para escutá-lo. Então vamos importá-lo em main.js:

```
// main.js

const { app, BrowserWindow, ipcMain }  = require('electron');
```

E para escutar o evento, chamamos a função `on` do `ipcMain`, passando o evento que queremos escutar e uma função a ser executada:

```
// main.js

ipcMain.on('abrir-janela-sobre', () => {
```

```
});
```

Dentro dessa função, podemos criar uma janela. Como estamos no processo principal, podemos inicializar um `BrowseWindow`:

```
// main.js

ipcMain.on('abrir-janela-sobre', () => {
    let sobreWindow = new BrowserWindow({
        width: 300,
        height: 220
    });
});
```

Quando criamos uma janela, precisamos carregar um arquivo, que será o HTML daquela janela. Como ainda não temos o HTML, vamos criar o arquivo **sobre.html**, dentro da pasta **app**, com o seguinte conteúdo:

```
<!-- sobre.html -->

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sobre o Alura Timer</title>
</head>
<body>
    <p>O Alura timer é para você guardar seus tempos de estudo na Alura.</p>
</body>
</html>
```

Com o HTML criado, podemos carregá-lo no processo principal:

```
// main.js

ipcMain.on('abrir-janela-sobre', () => {
    let sobreWindow = new BrowserWindow({
        width: 300,
        height: 220
    });
    sobreWindow.loadURL(`file://${__dirname}/app/sobre.html`);
});
```

Ao testar, vemos que clicando no link **Sobre**, a nossa janela é aberta. Mas o que acontece se clicamos várias vezes no link? Várias janelas são abertas, e isso não é um comportamento legal em um software desktop. A partir do momento que a janela **Sobre** está aberta, ao clicar no link, não deveria abrir novas janelas.

Vamos ver como lidar com múltiplas janelas no próximo vídeo!