Desafio prático — *PhotoViewer Lite* (com janelas e IPC avançados)

> Objetivo: construir um app Electron diferente do exemplo anterior, com janela sem moldura (frameless + titlebar customizada), janela de pré-visualização flutuante, persistência de posição/tamanho, e IPC nos 3 sentidos (Renderer → Main send, Renderer → Main invoke, Main → Renderer send).

🚀 Ideia do Desafio

Criar um **aplicativo simples de visualização de imagens** chamado **PhotoViewer Lite**, mas com alguns temperos extras que obrigam você a usar vários recursos do Electron:

1. **Janela principal diferente**
   - Ela **não tem a moldura padrão** do Windows/macOS (sem a barra tradicional de fechar/minimizar/maximizar).
   - Você mesmo cria uma **barra de título personalizada** com botões que se comunicam com o **Main** via IPC.
   - Dentro dela, você tem um botão para **abrir imagens do computador**.
2. **Abrir imagens**
   - Ao clicar no botão, abre a janela nativa do sistema para escolher um arquivo.
   - O **Main** é quem abre essa janela (não o Renderer).
   - Depois ele devolve para o **Renderer** os detalhes da imagem (caminho, nome, tamanho, largura e altura).
   - O **Renderer** mostra a imagem e as informações na tela ou no terminal.
3. **Janela de pré-visualização**
   - Você pode abrir/fechar uma janelinha extra que mostra a mesma imagem em miniatura.
   - Essa janela abre usando um **atalho do teclado** (exemplo: `Ctrl+Shift+P`).
   - Essa janela fica sempre por cima (tipo “picture-in-picture”).
4. **Salvar posição e tamanho da janela**
   - Quando você move ou redimensiona a janela, o **Main** guarda esses valores em um arquivo.
   - Quando você abrir o app de novo, ele volta no mesmo lugar/tamanho.
   - Se não for possível (mudou de monitor, etc.), o app centraliza a janela sozinho.
5. **Comunicação entre Main e Renderer**
   - **Renderer → Main (send):** para mandar comandos como “fechar”, “minimizar”, “maximizar”.

    - **Renderer → Main (invoke):** para pedir algo e esperar resposta, por exemplo: “abre uma imagem e me devolve os dados”.
    - **Main → Renderer (send):** para avisar mudanças, como “a janela foi redimensionada” (e o Renderer mostra isso em um status bar).
6. **Atalhos extras**
    - `Ctrl+Alt+Left`: encaixa a janela na metade esquerda da tela.
    - `Ctrl+Alt+Right`: encaixa na metade direita.
    - `Ctrl+Alt+Up`: centraliza com tamanho 2/3 da tela.

Arquitetura e arquivos

```
photo-viewer-lite/
  ├─ package.json
  ├─ main.js
  ├─ preload.js
  ├─ index.html
  ├─ renderer.js
  ├─ about.html
  └─ styles.css        (opcional, para a titlebar e layout)
```

Canais IPC (exatos, para você implementar):

- Renderer → Main (fire-and-forget):

  `windowControls.minimize`, `windowControls.maximizeRestore`, `windowControls.close`

- Renderer → Main (request/response):

  `windowControls.openImage()` → retorna metadados e buffer/base64

- Main → Renderer (broadcast):

  `window-state-updated`, `theme-changed` (opcional)

✅ O que você precisa entregar (versão simples)

- Uma janela sem moldura com **botões de controle que funcionam**.
- Botão para **abrir imagem** e mostrar informações na tela.
- Uma janela de **pré-visualização** que abre/fecha com atalho.
- A janela **abre no mesmo lugar/tamanho** em que foi fechada.
- Atalhos de **encaixe/centralização** funcionando.

📦 Entrega (o que subir)

- Repositório no Github.
- **README.md** curto explicando:
    - Como rodar (`npm start`).
    - O que foi implementado (checklist dos requisitos).