



Mateus Gomes Pereira - SP3060357  
Denysson Max Oliveira dos Santos - SP3063763



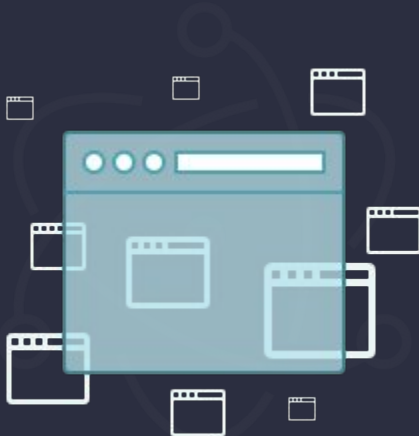
# O que é o Electron?

Electron é um framework que permite criar aplicações desktop com JavaScript, HTML e CSS.

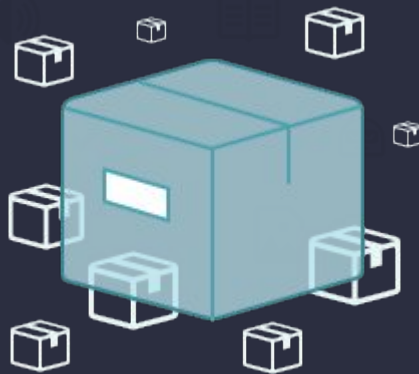
Utiliza o Chromium e o Node.js no binário, permitindo a criação de aplicações multiplataforma (Windows, Linux e MacOS) com um mesmo código fonte.



**Código Aberto**



**Tecnologias Web**



**Multiplataforma**



# Electron Fiddle

Electron Fiddle é um aplicativo sandbox escrito com Electron e suportado por mantenedores do Electron. É uma ferramenta de aprendizagem para experiências com APIs do Electron, ou para protótipos de recursos durante o desenvolvimento.



[Visitar o site...](#)



# Pré-requisitos

- ✓ Conhecimentos em Javascript, HTML5, CSS3 e Node.js.
- ✓ Node.js na última versão LTS disponível (recomendado).

**Observação:** o Electron incorpora o Node.js ao binário, por isso a versão instalada no sistema pode não ser a mesma que executa o código.



# Pré-requisitos

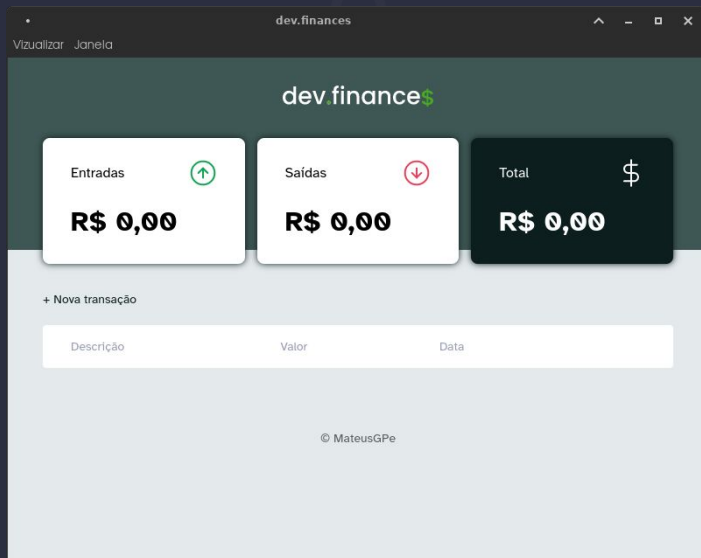
```
npm config set proxy "http://10.100.1.253:3128/"  
npm config set https-proxy "http://10.100.1.253:3128/"
```



# Criar um aplicativo Electron

Utilizando a Atividade 4 de DW2A4.

- <https://github.com/mateusGPr>
- <https://github.com/DenyssonMax>





# 1 - Criar e selecionar a pasta do projeto

```
Terminal - mateusgp@mush-bedroom: ~/projects/a4-electron ^ _ □ ×  
Arquivo Editar Ver Terminal Abas Ajuda  
mateusgp@mush-bedroom:~/projects$ mkdir a4-electron  
mateusgp@mush-bedroom:~/projects$ cd a4-electron  
mateusgp@mush-bedroom:~/projects/a4-electron$
```



## 2 - Criar o projeto

O **entry point** deve ser *main.js*

**description** e **author** não podem ser vazios.

npm init

```
Terminal - mateusgp@mush-bedroom: ~/projects/a4-electron
Arquivo  Editar  Ver  Terminal  Abas  Ajuda

mateusgp@mush-bedroom:~/projects/a4-electron$ yarn init
yarn init v1.22.10

question name (a4-electron):
question version (1.0.0):
question description: Aplicação Electron da Atividade 4 de DW2A4.
question entry point (index.js): main.js
question repository url:
question author: Mateus e Denysson
question license (MIT):
question private:
success Saved package.json

Done in 54.70s.

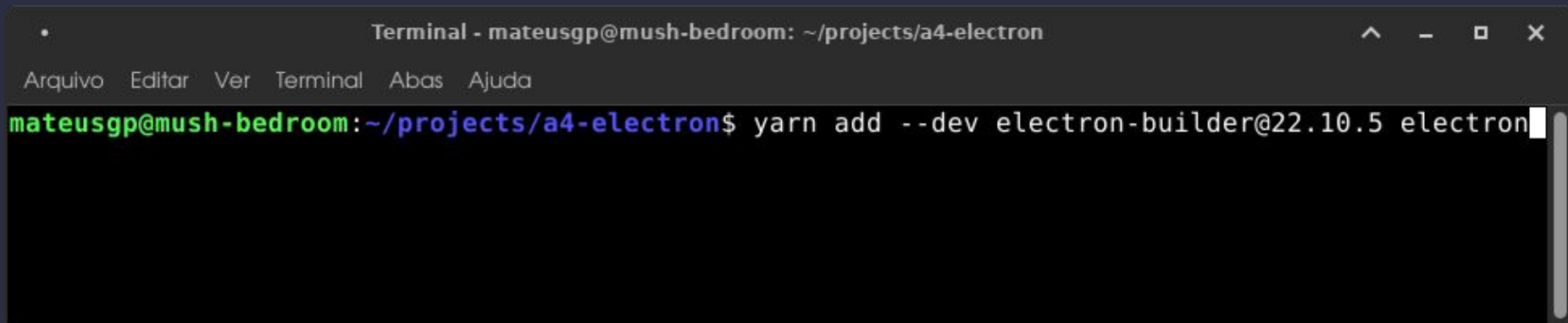
mateusgp@mush-bedroom:~/projects/a4-electron$
```





## 3 - Instalar as dependências

**Observação:** a versão **22.10.5** é para compatibilidade com versões anteriores ao **Node.js 18**.  
Se **node -v** for maior ou igual a 18, pode ser omitida a versão (**@22.10.5**).



```
Terminal - mateusgp@mush-bedroom: ~/projects/a4-electron
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
mateusgp@mush-bedroom:~/projects/a4-electron$ yarn add --dev electron-builder@22.10.5 electron
```

```
npx cross-env ELECTRON_GET_USE_PROXY=true GLOBAL_AGENT_HTTPS_PROXY=http://10.100.1.253:3128/
npm install -D electron@latest electron-builder@22.10.5
```



## 4 - Criar o arquivo *main.js*

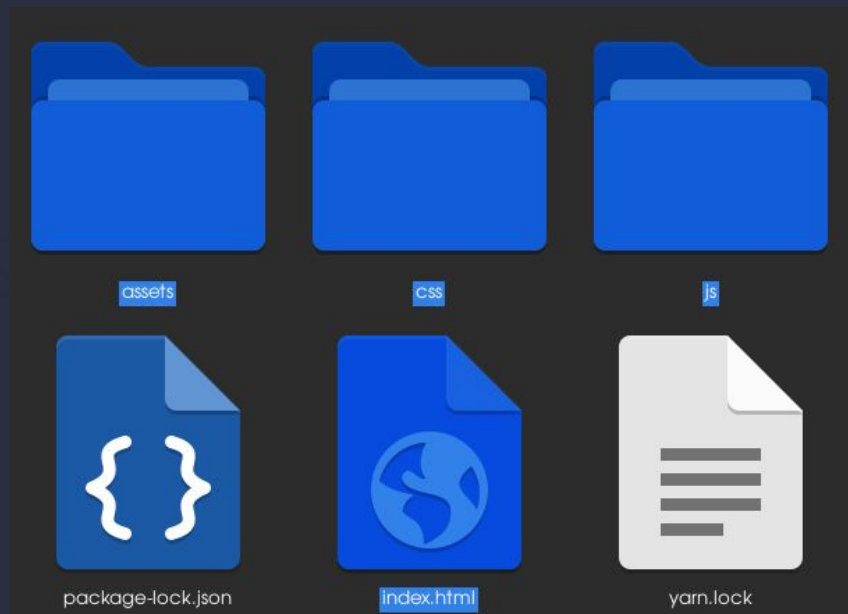
**Observação:** no Prompt de Comandos do Windows, somente *type nul >> "main.js"* é válido. O arquivo pode ser criado pelo Windows Explorer ou VS Code.

```
Terminal - mateusgp@mush-bedroom: ~/projects/a4-electron
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
mateusgp@mush-bedroom:~/projects/a4-electron$ touch main.js
mateusgp@mush-bedroom:~/projects/a4-electron$ type nul >> "main.js"
```



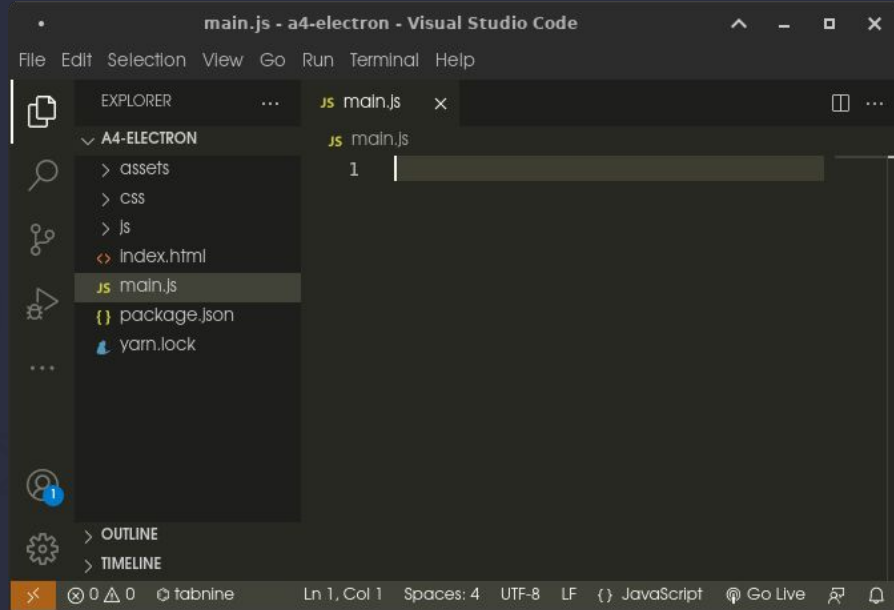
## 5 - Baixar os arquivos

Baixar os arquivos do GitHub da matéria e extrair os arquivos da atividade 4 na pasta a4-electron.





## 6 - Abrir a pasta no VS Code e selecionar *main.js*





## 7 - Importar os módulos do Electron JS

**app** - controla os eventos do ciclo de vida da aplicação.

**BrowserWindow** - cria e gerencia as janelas da aplicação.

**Menu** - cria e gerencia os menus.

```
js main.js > ...  
1  const { app, BrowserWindow, Menu } = require('electron')  
2
```



## 8 - Função para abrir a janela e carregar o HTML

`new BrowserWindow({ width: 800, height: 600 })` - cria uma janela de 800x600 pixels.

`mainWindow.loadFile('index.html')` - carrega a página *index.html*.

```
const createWindow = () => {  
  const mainWindow = new BrowserWindow({ width: 800, height: 600 })  
  mainWindow.loadFile('index.html')  
}
```



## 9 - Abrir a janela e carregar a página

`app.whenReady().then(/* Função */)` - executa a **Função** quando o Electron terminar a inicialização e estiver pronto para abrir a janela do **Web Browser**.

`createWindow(); setMainMenu();` - executa as funções para criar a janela, carregar a página e alterar o menu principal.

`app.on('activate', () => {if (BrowserWindow.getAllWindows().length === 0) createWindow()})` - No MacOS, cria uma nova janela, se todas forem fechadas, mas o ícone no **Dock** é clicado.

```
app.whenReady().then(() => {
  createWindow()
  setMainMenu()

  app.on('activate', () => {
    if (BrowserWindow.getAllWindows().length === 0) createWindow()
  })
})
```



# 10 - Finalizar o processo

`app.on('window-all-closed', /* Função */) - executa a Função quando todas as janelas do Web Browser são fechadas.`

`if (process.platform !== 'darwin') app.quit()` - verifica se o aplicativo está executando no **MacOS**, caso verdadeiro, o processo não é finalizado.

\*No Windows e no Linux, fechar todas as janelas geralmente encerra totalmente a aplicação. No MacOS a aplicação só é encerrada ao selecionar a opção na barra de menu ou pressionar Command (⌘)-Q

```
app.on('window-all-closed', () => {  
  if (process.platform !== 'darwin') app.quit()  
})
```





## 11 - Modificar o menu

**template** - Um objeto que contém o modelo do menu.

```
const menu = Menu.buildFromTemplate(template) -
```

cria um objeto a partir do modelo.

```
Menu.setApplicationMenu(menu) - altera o menu
```

padrão da aplicação para o criado.

```
function setMainMenu() {  
  const template = [  
    {  
      label: 'Vizualizar',  
      submenu: [  
        { role: 'reload' },  
        { role: 'toggledevtools' },  
        { type: 'separator' },  
        { role: 'resetzoom' },  
        { type: 'separator' }  
      ]  
    },  
    {  
      label: 'Janela', role: 'window',  
      submenu: [  
        { role: 'minimize' },  
        { role: 'close' }  
      ]  
    }  
  ];  
  
  const menu = Menu.buildFromTemplate(template)  
  Menu.setApplicationMenu(menu)  
}
```



## 12 - Editar *index.html*

Inserir as tags **meta** com as Políticas de Segurança de Conteúdo, permitir **scripts inline**, fontes e estilos no mesmo endereço e do **Google**.

- *Os URLs do Google só são necessários se são usadas fontes e/ou estilos destes.*

```
<!-- https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP -->  
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src  
'self' 'unsafe-inline'; style-src 'self' https://fonts.googleapis.com; font-src  
'self' https://fonts.gstatic.com;">  
<meta http-equiv="X-Content-Security-Policy" content="default-src 'self';  
script-src 'self' 'unsafe-inline'; style-src 'self' https://fonts.googleapis.com;  
font-src 'self' https://fonts.gstatic.com;">
```



## 12 - Editar *package.json*

Adicionar scripts para agilizar a execução das tarefas:

**start** - inicia a aplicação

**dist** - empacotar a aplicação.

**pack** - apenas comprime em um pacote, útil para testes.

```
{
  "name": "a4-electron",
  "version": "1.0.0",
  "description": "Aplicação Electron da Atividade 4",
  "main": "main.js",
  "author": "Mateus",
  "license": "MIT",
  "devDependencies": {
    "electron": "^18.2.3",
    "electron-builder": "22.10.5"
  },
  ▶ Debug
  "scripts": {
    "start": "electron .",
    "pack": "electron-builder --dir",
    "dist": "electron-builder"
  }
}
```



## 13 - Editar *package.json* - Configurar o Electron Builder

Uma configuração minimalista para empacotar para Windows e Linux, entretanto só será empacotado a aplicação na mesma plataforma.

São necessárias dependências adicionais para o empacotamento multiplataforma.

```
"build": {  
  "win": {  
    "target": "portable"  
  },  
  "linux": {  
    "target": "AppImage"  
  }  
},  
"postinstall": "electron-builder install-app-deps"
```



# 14 - Executar e empacotar

**yarn start** - executar a aplicação

**yarn dist** - empacotar a aplicação

```
mateusgp@mush-bedroom:~/projects/a4-electron$ yarn dist
```

```
yarn run v1.22.10
```

```
$ electron-builder
```

- electron-builder version=22.10.5 os=5.10.0-13-amd64
- loaded configuration file=package.json ("build" field)
- writing effective config file=dist/builder-effective-config.yaml
- packaging platform=linux arch=x64 electron=18.2.3 appOutDir=dist/linux-unpacked
- Unpacking electron zip zipPath=undefined
- building target=AppImage arch=x64 file=dist/a4-electron-1.0.0.AppImage
- application Linux category is set to default "Utility" reason=linux.category is not set and cannot map from macOS docs=<https://www.electron.build/configuration/linux>
- default Electron icon is used reason=application icon is not set

```
Done in 8.72s.
```

```
mateusgp@mush-bedroom:~/projects/a4-electron$
```

```
mateusgp@mush-bedroom:~/projects/a4-electron$ yarn start
yarn run v1.22.10
$ electron .
```