INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO ACRE

EZEQUIEL SOARES DA SILVA

Electron

Rio Branco – Acre
2022

Ezequiel Soares da Silva

Electron

Trabalho em caráter avaliativo para composição de nota na disciplina de Linguagens de Programação no Curso Tecnólogo em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia do Acre, Campus Rio Branco.

Rio Branco – Acre

2022

1. INTRODUÇÃO

Quando temos uma página web a sua vantagem é a sua onipresença, um site ele se torna muito amplo e abrange múltiplas plataforma apenas desenvolvendo uma vez. Mas isso tem uma limitação, no âmbito de sistema operacional ele se limita apenas com os recursos providos do navegador, assim não podendo criar aplicações que controlam ativamente sistemas de arquivos e mais. Numa aplicação WEB não conseguimos ler outro tipo de linguagem diferente do JS, e as APIs do sistema operacional não são acessadas diferentemente de aplicativos desktop. Outra questão é a conexão à internet, você só pode acessar um site online se estiver conectado a uma rede.

Nessa apostila iremos abordar o Electron, onde com ele podemos construir aplicativos desktop multiplataforma reaproveitando conhecimentos adquiridos no desenvolvimento web. Vimos algumas das desvantagens de uma aplicação pura na web, para o desenvolvimento nativo para aplicativos desktops enfrentamos também alguns problemas, já que muitas das vezes aprender novas linguagens e estruturas, assim tento que enfrentar uma longa curva de aprendizagem. Com o Electron podemos reaproveitar nossos conhecimentos e habilidades no desenvolvimento web para criar aplicativos desktop com muitos recursos de uma aplicação nativa.

O Electron é um runtime, onde já citamos, permite criar aplicativos desktop com linguagens de desenvolvimento web, HTML, CSS e JavaScript e também frameworks como VueJS, ReactJS e Angular.

1.1 Breve história

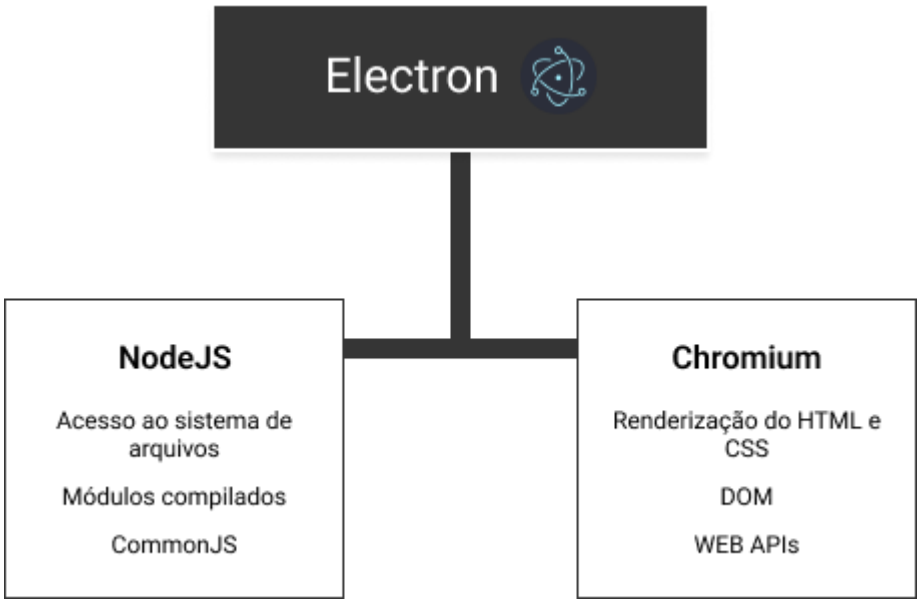
Seu desenvolvimento foi iniciado pelo engenheiro do GitHub Cheng Zhao em 2013, onde era a base do desenvolvimento de um aplicativo chamado Atom Shell, que era um editor de texto multiplataforma do GitHub construído com as tecnologias da web.

Após seu potencial ser descoberto, em 2014 se tornou Open Source, então qualquer pessoa poderia ver seu código fonte e submeter alterações.

2 DESENVOLVIMENTO

O Electron combina o módulo de conteúdo do Chromium (navegador da Google open source) e o runtime do NodeJS. Permitindo criar GUIs (Interface gráfica do utilizador) como páginas web, além de acessar recursos nativos de sistema operacional no Widows, macOS e Linux. O Electron reúne essas duas plataformas para criar uma aplicação totalmente nova, com todos os recursos disponíveis do JavaScript e NodeJS.

Como vimos o Electron não é apenas uma casca que roda uma aplicação web em ambiente de aplicativo desktop nativo, juntamente com o NodeJS você pode fazer muitas aplicações que já estão disponíveis nesse modelo com uma interface, como editar uma foto, onde é possível no NodeJS, mas sem uma GUI, onde para um usuário normal ficaria bem improvável sua utilização. Então com ele você só não pode construir aplicações web e também outros ótimos recursos poderosos, veja o esquema 2.1 abaixo:



2.1: O Electron combina os componentes principais da WEB do Chromium com o acesso de baixo nível do NodeJS

2.1 Conhecendo seus módulos: Chromium

O Chromium foi escolhido por ser o navegador do Google de código aberto, assim sendo flexível para mudanças e aprimoramentos mais rapidamente. Esse módulo faz com que podemos renderizar páginas web, utilizando a GPU, o *Blink rendering engine* e a engine de JavaScript V8. Esse módulo é o browser em si, mas sem algumas funcionalidades, como favoritos, salvar senha, autocomplete em inputs, histórico, etc. Seu principal papel é renderizar o HTML e CSS, integrando com JS fornecendo API, como a DOM e fetch.

2.2 NodeJS

O node foi uma solução de empregar o JavaScript do lado do servidor, antigamente só era possível ser executado em um interpretador que estava no navegador. O node usou o V8 da Google para interpretar o JavaScript, assim sendo usado para diversas funcionalidades, como acessar o sistema de arquivos, criar servidores e carregar módulos de código externos.

Ao amadurecimento dessa solução, ela foi empregada em diversos cenários, onde teve uma ampla variedade de aplicações, desde controlar robôs até criar aplicativos desktop.

2.3 Principais focos de aplicação da tecnologia

O Electron é utilizado em diversas empresas, sendo-as grandes e pequenas, para criar aplicativos desktop. Sua utilização criou o Visual Studio Code, uma IDE multiplataforma lançado pela Microsoft.

Ele é amplamente usado em soluções onde podemos ter uma compatibilidade e estabilidade para um software, diversas empresas exportam suas aplicações web como se fosse uma “demonstração” para explorar mais afundo na aplicação desktop. Podemos citar o Slack, popular aplicativo de mensagens empresarial, a versão desktop da Twitch é feita em Electron, o software de prototipagem Figma, o mais famoso aplicativo de comunicação em jogos, o Discord, além de outras aplicações para desenvolvedores usam o Electron.

2.4 Vantagens, desvantagens e limitações da tecnologia

Ao trabalhar com uma aplicação web padrão você pode ter vários cenários, um deles é que seus usuários podem estar utilizando a versão mais recente de um navegador como o Google Chrome, ou talvez um navegador desatualizado de muito tempo atrás, como o Internet Explorer. Isso pode fazer que as suas compatibilidades não sejam as mesmas, assim quebrando algumas funcionalidades, isso nos leva a implementar os recursos gradativamente, assim tendo que gerar fallbacks para funcionalidades não disponíveis em alguns navegadores.

Com o Electron esse processo é trivial, já que após o usuário instalar o seu aplicativo, a sua versão é específica na de qual foi desenvolvida, sendo empacotada para ser distribuída. O Electron permite que você use recursos de plataforma web de ponta porque inclui uma versão relativamente recente do Chromium.

Sua limitação e desvantagens são de performance, seu processo de compilação gera executáveis grandes, apenas o Chromium compilado dar um total de 30mb e sem falar da performance enquanto o aplicativo está sendo executado.

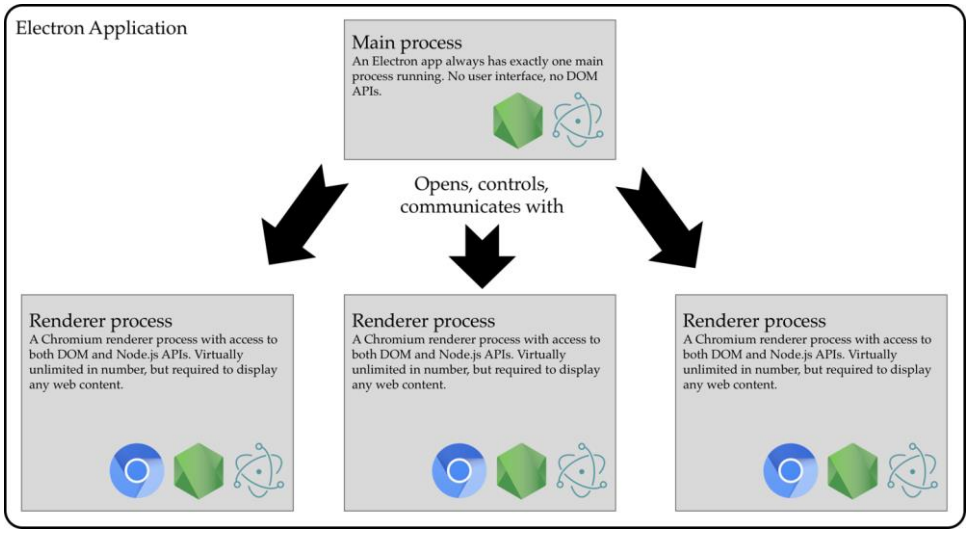
2.4 Entendendo como o Electron funciona

Ao contrário dos aplicativos web tradicionais, os aplicativos Electron não se limitam ao navegador. Você pode criar aplicativos que ficam na barra de menus ou na bandeja do sistema. Você pode até registrar atalhos globais para acionar esses aplicativos ou qualquer uma de suas habilidades com um toque de tecla especial de qualquer lugar do sistema operacional.

Os aplicativos feito em Electron têm acesso a informações no nível do sistema, como se o computador está com bateria ou conectado à uma fonte de energia. Eles também podem manter o sistema operacional acordado e impedir que ele entre no modo de economia de energia, se necessário.

O Electron funciona como um aplicativo desktop nativo, ficando na barra de tarefas no Windows, no dock no macOS, etc. Consumindo recursos da máquina que está o executando e criando processos. Com ele você pode ter acesso a maioria de recursos que um aplicativo nativo.

Uma aplicação em Electron tem dois tipos de processos: o principal (main process) e os processos de renderização (renderer process), veja os a imagem 2.4.1:



2.4.1: Sistema de processos do Electron

Como você pode ver em um nível mais baixo do Electron está conectado intimamente com o sistema operacional com o NodeJS, esse processo principal é onde irá começar nossa aplicação, onde devemos ditar seus atributos, como botões de ação (maximizar, minimizar e fechar), seu ícone, tamanho da tela, etc. O main process tem responsabilidade mais intima com os sistemas operacionais, é responsável pela comunicação com as APIs nativas. Se você deseja exibir uma caixa de diálogo para abrir ou salvar um arquivo, faça isso no processo principal.

Agora os processos de renderização é criado pelo processo princial, onde é estruturado e renderizado de acordo com seu ciclo de vida, esse processo é criado pelo Electron's BrowserWindow module, onde iremos configurar seus atributos já citados. Nesse processo você tem acesso a todos as APIs tanto da web e também do NodeJS, permitindo que você use módulos nativos e interações de sistema de nível inferior, mas cuidado, você não irá criar todo seu backend com NodeJS no Electron, isso ainda irá rodar no lado do usuário, então o mais recomendado é deixar seu banco de dados e regras de negócio isolada, do lado do servidor. As operações com NodeJS serve para operações a nível de sistema operacional, como acessar os sistemas de arquivo, manipular a área de transferência, etc.

2.6 O que preciso saber para usar Electron

Para começar a criar aplicações desktop com Electron você deve no mínimo ser familiarizado com ambiente de desenvolvimento do NodeJS, como módulos CommonJS, desejável já ter feito algum projeto utilizando um gerenciador de pacotes do NodeJS, como npm ou yarn.

Agora na parte do desenvolvimento web você deve ter conhecimento com HTML, CSS e JS, não precisa ser um especialista, mas a base da criação e relacionamento com os módulos são feitas com as tecnologias da web, se você quiser ser frontend esses conhecimentos são obrigatórios.

2.7 Recursos necessários para uso da tecnologia

Primeiramente devemos instalar a versão atual LTS do NodeJS, você pode acessar seu site oficial: https://nodejs.org/en/.

É apenas isso, o gerenciador de pacotes que iremos utilizar nessa apostila vem junto com o NodeJS, o npm.

2.8 Primeira aplicação

Vamos começar a nossa primeira aplicação, com os processos de configuração, instalação e início de um projeto em Electron. Iremos começar com o ponto de entrada do nosso aplicativo:

```
$ mkdir electron-aplicativo && cd electron aplicativo
```

```
$ npm init
```

Primeiramente criamos um diretório próprio para a nossa aplicação e navegamos para dentro dele. Após isso iniciaremos o gerenciador de pacotes npm, você pode ir confirmando as configurações.

Posteriormente devemos instalar o electron como dependência de desenvolvimento do nosso projeto:

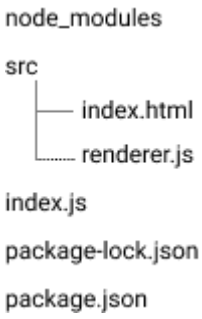
```
$ npm install --save-dev electron
```

Agora que instalamos o electron, temos que modificar o package.json, o arquivo gerado no início de um projeto npm.

```
{
  "name": "electron-aplicativo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
```

```
    "start": "electron ."
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "electron": "^16.0.7"
  }
}
```

No final o arquivo ficará desta forma, com a inserção do script para iniciar o projeto em electro, que claro, não foi configurado.



Aqui vemos o sistema de pastas da do nosso projeto, que começa com node_modules, que são os pacotes que o npm instalou, o diretório SRC irá conter as nossas visualizadoras, o HTML e os renderer que serão os arquivos que irão renderizar no documento HTML e como a gente viu, nos processos tem um processo principal e os processos de renderização, que será um utilizado nesse diretório. Depois temos o index.js que ele será a entrada para a para o nosso Electron, é nele que terá configurações e nele consiste no main process, devemos colocar configurações gerais como nossos ícones, formato da nossa janela e como arquivos de configurações. Agora o package-lock.json e package.json são arquivos do projeto iniciado do npm.

2.8.1 Configurando o Electron

Até agora estruturamos nossa aplicação, instalamos alguns pacotes e entendemos qual função ele exerce. Neste momento vamos ver a sua configuração na prática no arquivo index.js:

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

const createWindown = () => {
  const win = new BrowserWindow({
    backgroundColor: '#fff',
    maxWidth: 800,
    maxHeight: 600,
    resizable: true,
    roundedCorners: true,
    icon: './assets/icon.png',
    titleBarStyle: 'hidden',
    titleBarOverlay: {
      color: '#f3fcfc',
      symbolColor: '#181818'
    },
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false
    }
  })

  win.loadFile('./src/index.html')
}

app.whenReady().then(() => {
  createWindown()
})
```

O módulo app controla o ciclo de vida da nossa aplicação, como o whenReady indica se a aplicação está pronta para ser iniciada. O módulo BrowserWindow cria e gerencia as janelas da nossa aplicação.

A próximo passo é criar uma função que irá instanciar uma nova janela e também fazer a leitura do nosso documento HTML, na instancia do BrowserWindow passamos um objeto de configurações, que podem ser várias, verifique a documentação oficial para vê-las e aplicar o que se encaixa no seu contexto.

Após isso temos que verificar se o aplicativo está pronto para iniciar, e para isso temos o método do app, whenReady, que retorna um promisse, que será resolvida apenas quando o aplicativo for iniciado, e após isso devemos chamar a função para criar uma janela.

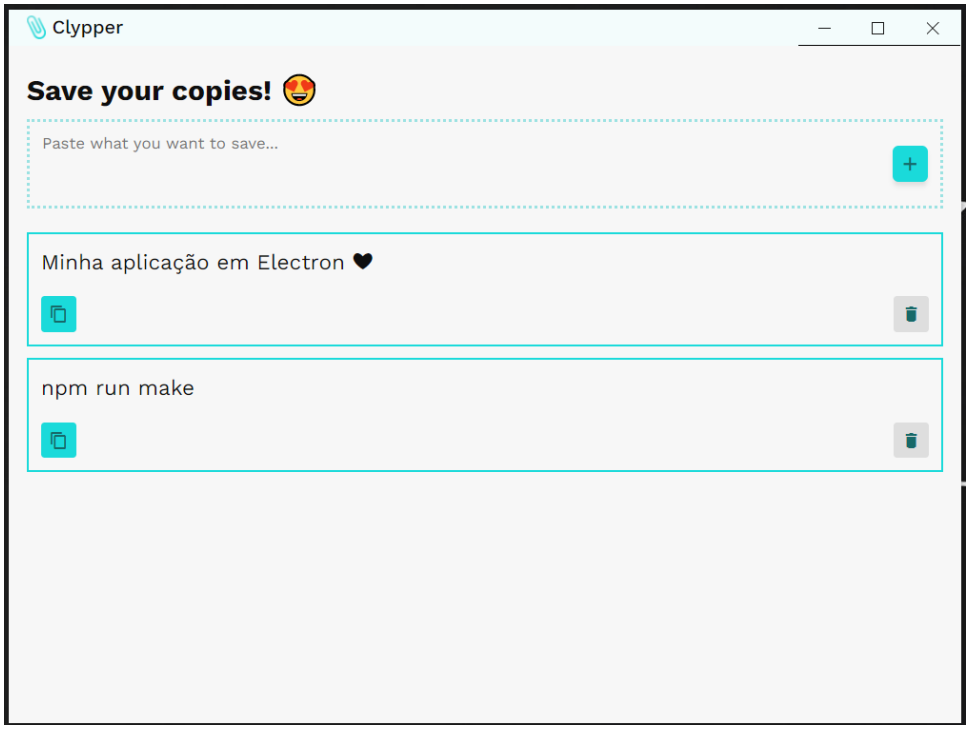
2.8.2 Nossa aplicação

Por enquanto é apenas isso que utilizaremos de electron, é basicamente isso, podemos fazer vários tipos de configurações no processo principal, também integrar com API nativas do sistema operacional com node, e renderizar isso tudo com HTML, CSS e JS.

Agora vamos para a parte do processo de renderização, a nossa aplicação é bem simples, iremos fazer um aplicativo para guardar cópias e copiar para a área de transferência caso o usuário queira. É bem simples, a parte de HTML, CSS e JS não tem nada demais, única coisa que utilizaremos própria do electron é a API clipboard:

```
const { clipboard } = required(“electron”)

// importando o clipboard do electron

clipboard.writeText(“Texto”)

// com o método writeText copiamos alguma string para área de transferência do usuário

clipboard.readText()

// ler o que tem na área de transferência do usuário
```

É essa API que iremos utilizar para produzir na aplicação, o código já pronto você tem acesso aqui: https://github.com/MrEzequiel/clypper. O código do aplicativo é trivial para quem conhece do assunto, e se você já quiser se aventurar nesse framework, considero conhecimentos como esses no desenvolvimento web, mas não se sinta desencorajado para aprender HTML, CSS e JS utilizando o electron para construir aplicações desktop.



2.8.3 Build da aplicação

Temos diversas formas para buildar uma aplicação em Electron, iremos utilizar o Electron Forge, a mais rápida e fácil. O processo de build é uma compilação ou transpiração do nosso código para ser executado em outro contexto, otimizando e automatizando processos, evitando nós mesmos fazer isso. O build é a versão estável e otimizada da aplicação em desenvolvimento, ela que devemos distribuir para o usuário final, onde ele só irá se preocupar em baixar e abrir.

Para começar o processo de build, precisamos instalar uma dependência e rodar um comando para gerar script e configurações automaticamente:

```
$ npm install --save-dev @electron-forge/cli
$ npx electron-forge import
```

Já estamos aptos para fazer o build da nossa aplicação, podemos ver que o git foi iniciado e o nosso package.json foi modificado. Para fazer o build devemos:

```
$ npm run make
```

O Electron Forge cria um diretório out no nosso projeto, contém os arquivos do build, com eles você pode mandar o instalador e executar em qualquer lugar para acessar a nossa aplicação.

3 Conclusão

Vemos várias abordagens de como o Electron pode ser feito implemetando e como é a integração com suas API juntamente com tecnologias de aplicação web, então essa ferramenta é amplamente utilizada no mercado. Várias empresas hoje tem como aplicações desktop o Electron, como discord (aplicativo de comunicação), o VS Code da Microsoft (um ambiente de desenvolvimento), Hyper (terminal do windows customizável), slack (aplicativo de mensagem, mas bem famoso no ramo empresarial). Vimos então que esse framework é muito bem utilizado pela comunidade, também suportado sendo ele umas das principais a ferramentas para se criar aplicações desktop sem muitos esforços reaproveitando o conceito como já falamos de tecnologias web, onde detém um grande ecossistema, tem uma maturidade e está bem famosa no mercado e muitas pessoas desenvolve as utilizando.

o Electron uma tem uma fraqueza no quesito performance, por causa de ele não ser um aplicativo puramente nativo e precisa passar em o nível de compilação. A aplicação nativa então é sempre mais viável a construir aplicação, mas justamente a

vantagem do Electron é não ser preciso aprender uma nova linguagem para construir aplicações desse tipo, se tiver conhecimentos de desenvolvimento web.

REFERENCIA

Kinney, Steve. Electron in Action. 2 de outubro de 2018

Electron: Crie aplicativos desktop multiplataforma com JavaScript, HTML e CSS. Electronjs.org, 2022. Disponível em: < https://www.electronjs.org/ >. Acesso em: 01/01/2021.