Listas de Prospecção .vcf com Javascript no Console do Browser

Acesse no GitHub

OSÉIAS MAGALHÃES

oseiasmagalhaes.com.br

Este guia detalhado demonstra como automatizar a criação de listas de prospecção em um algoritmo simples e funcional para o formato .vcf, utilizando Javascript, Google Maps, diretamente no navegador Chrome.

O objetivo final deste projeto é utilizar o WhatsApp ou outros apps de conversa para acessar os contatos criados, agilizando o envio de mensagens. Não disponibilizarei dados reais neste documento.

O processo é dividido em seis etapas:

1 - Extensão .vcf

O formato .vcf (Virtual Card File) é um padrão aberto para armazenar informações de contato, como nome, telefone, endereço e site. É ideal para criar listas de prospecção, pois permite importar facilmente os contatos para diversos aplicativos, como agendas e softwares de CRM.

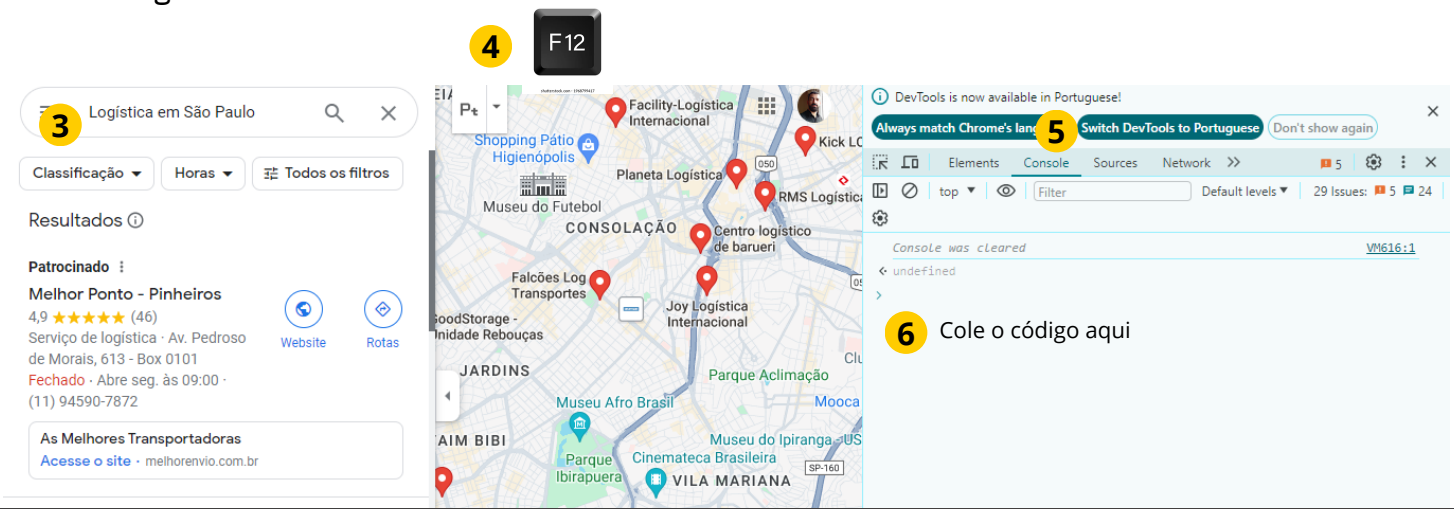
2. Extração de dados usando Javascript:

A extração de dados do Google Maps para este projeto contempla apenas nome e telefone de empresas no Google Maps. Usando um código javascript para automatizar um processo que seria manual, ao final da extração este código gera um arquivo .vcf e faz o download.

3. Siga os seguintes passos:

1. Copie esse código
2. Acesse o site GMaps https://maps.google.com/
3. Faça sua busca pressione F12
4. Clique em Console
5. Cole-o no Console do Navegador *
6. Mantenha a página aberta para que os dados possam ser baixados

*Caso o console não permita a colagem do código digite “permitir colar”, tecle enter, em seguida cole o código.



4. Análise do Código

const scrollInterval = setInterval(rolarScroll, 3000)

Esse código JavaScript inicializa com uma chamada rolarScroll que rola automaticamente o scroll de um contêiner específico. Vou explicar o que cada parte faz:

1. **function rolarScroll() { ... }**: Define uma função chamada rolarScroll que será chamada repetidamente em intervalos regulares para rolar o scroll do contêiner.
2. **const container = document.querySelector('#QA0Szd...')**: Seleciona o contêiner específico no qual você deseja rolar o scroll. O seletor CSS usado (#QA0Szd > div > div > div.w6VYqd > div.bJzME.tTVLSc > div > div.e07Vkf.kA9KIf > div > div > div.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd > div.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd) identifica o contêiner com base em sua estrutura HTML.
3. **if (container && container.scrollHeight > container.clientHeight + container.scrollTop) { ... } else { ... }**: Verifica se o contêiner existe e se o scroll ainda não atingiu o final. Isso é feito comparando a altura total do contêiner (scrollHeight) com a altura visível do contêiner mais a posição atual do scroll (clientHeight + scrollTop).
4. **container.scrollTop += 800**: Se o scroll ainda não atingiu o final, a posição do scroll é incrementada em 800 pixels para rolar a página para baixo.
5. **clearInterval(scrollInterval)**: Se o scroll atingiu o final, o intervalo de rolagem é parado usando clearInterval(scrollInterval).
6. **extrairDadosContato()**: Quando o scroll chegar ao final, chama a função extrairDadosContato().
7. **const scrollInterval = setInterval(rolarScroll, 3000)**: Define um intervalo de tempo (3000 milissegundos ou 3 segundos, neste caso) para chamar a função rolarScroll repetidamente. Isso faz com que o scroll seja rolado automaticamente a cada 3 segundos.

extrairDadosContato()

- **Seleção de elementos:** O código começa selecionando todos os elementos HTML que têm a classe **.UaQhfb**, que é onde estão concentrados os dados de contato no Google Maps.
- **Extração de dados de contato:** Em seguida, ele extrai os dados de contato de cada elemento encontrado. Para cada elemento, ele busca os elementos com as classes **.qBF1Pd** e **.UsdlK**, que geralmente contêm o nome e o telefone do contato, respectivamente. Esses dados são armazenados em um array de objetos chamado dadosContato, onde cada objeto contém um nome e um telefone.
- **Escrita de arquivo:** Depois de extrair os dados de contato, o código itera sobre o array dadosContato e constrói uma string no formato de cartão de visita eletrônico (vCard) para cada contato. Essa string é concatenada à variável conteudoArquivo, que é inicializada fora da função. Cada linha do vCard é adicionada à conteudoArquivo, separando as informações com quebras de linha (\n).

- **Download do arquivo:** Após criar o conteúdo do arquivo, a função **criarArquivoTxtEDownload** é chamada, passando o conteúdo do arquivo e o nome do arquivo como parâmetros. Esta função, é responsável por criar um arquivo de texto com o conteúdo especificado na variável **conteudoArquivo** e iniciar o download automático desse arquivo.

Para facilitar a busca por nicho em agendas, base de contatos, concatena-se a busca ao nome da empresa extraído, por exemplo, “Nome da empresa” + “Logística em São Paulo”.

Conclusão:

Automatizar a criação de listas de prospecção com javascript e .vcf é uma estratégia valiosa para profissionais, empresas e pessoas em busca de oportunidades. Esse processo economiza tempo, reduz erros e melhora a organização dos contatos. Lembre-se de sempre utilizar as informações coletadas de maneira ética e respeitosa.

É importante realizar a concatenação do nome da empresa com o segmento pois isso poderá facilitar a prospecção por nichos específicos.

Benefícios para diferentes públicos:

Profissionais:

- Crie listas de prospecção para novos clientes.
- Encontre potenciais parceiros e colaboradores.
- Amplie sua rede de contatos profissionais.

Pequenas e médias empresas:

- Aumente a base de clientes e leads.
- Genere novas oportunidades de negócios.
- Otimize o processo de vendas e marketing.

Grandes empresas:

- Encontre fornecedores e parceiros estratégicos.
- Realize pesquisas de mercado e coleta de dados.
- Crie campanhas de marketing direcionadas.

Pessoas em busca de oportunidades de emprego:

- Encontre empresas do seu segmento de atuação.
- Facilite o envio de currículos direcionados.
- Aumente suas chances de conseguir uma entrevista.

Observações importantes:

- É fundamental respeitar as leis anti-spam e de proteção de dados ao utilizar listas de prospecção.
- Envie mensagens dentro do limite diário permitido pela plataforma responsável, para que você não seja banido.
- Utilize os contatos de forma ética e profissional.

Mantenha suas listas atualizadas para garantir a sua eficácia.

Requisitos para este projeto

Os requisitos mínimos para rodar o projeto de criação de listas de prospecção com Python e .vcf dependem do seu sistema operacional e das suas necessidades específicas. No entanto, em geral, você precisará de um computador desktop que tenha um navegador Chrome ou Microsoft Edge, os quais foram testados.

Observações

A atualização por meio da plataforma Google Maps ao seu site implicará na descontinuidade deste código, dado que utiliza-se de referência a elementos, que ao serem modificados deixam de funcionar corretamente.

De toda forma este código continua válido com objeto de estudo para a automatização de tarefas.

```
//Copie esse código
//Acesse o GMaps https://maps.google.com/
//Faça sua busca pressione F12
//Clique em Console
//Cole-o no Console do Navegador
//Mantenha a página aberta para que os dados possam ser baixados

function criarArquivoTxtEDownload(conteudo, nomeArquivo) {
    // Cria um Blob com o conteúdo do arquivo de texto
    const blob = new Blob([conteudo], { type: 'text/plain' });

    // Cria um URL para o Blob
    const url = window.URL.createObjectURL(blob);

    // Cria um link de download
    const linkDownload = document.createElement('a');
    linkDownload.href = url;
    linkDownload.download = nomeArquivo;

    // Adiciona o link à página (é necessário para alguns navegadores)
    document.body.appendChild(linkDownload);

    // Dispara o clique no link automaticamente
    linkDownload.click();

    // Remove o link da página
    document.body.removeChild(linkDownload);

    // Limpa o URL do Blob
    window.URL.revokeObjectURL(url);
}

function extrairDadosContato() {
    const contatos = document.querySelectorAll('.UaQhfb');

    const dadosContato = Array.from(contatos).map(contato => {
        const nomes = Array.from(contato.querySelectorAll('.qBF1Pd')).map(nome
=> nome.textContent.trim());
        const telefones =
Array.from(contato.querySelectorAll('.UsdlK')).map(telefone =>
telefone.textContent.trim());

        return {
            nomes: nomes,
            telefones: telefones
        };
    });

const busca = document.querySelectorAll('.searchboxinput')[0].value;
var conteudoArquivo = "";
const nomeArquivo = busca + '.vcf';

// Escrever arquivo
dadosContato.forEach(contato => {
```

```
    conteudoArquivo += 'BEGIN:VCARD\n' +
                       'VERSION:3.0\n' +
                       'FN:' + contato.nomes[0] + ' ' + busca + '\n' +
                       'TEL:' + contato.telefones[0] + '\n' +
                       'END:VCARD\n\n';
});

criarArquivoTxtEDownload(conteudoArquivo, nomeArquivo);
    //console.log(conteudoArquivo);

}

// Função para rolar automaticamente o scroll
function rolarScroll() {
    // Selecione o contêiner que você deseja rolar
    const container =  document.querySelector('#QA0Szd > div > div > div.w6VYqd
> div.bJzME.tTVLSc > div > div.e07Vkf.kA9KIf > div > div >
div.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd >
div.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd');

    // Verifique se o contêiner existe e se o scroll ainda não atingiu o final
    if (container && container.scrollHeight > container.clientHeight +
container.scrollTop) {
        container.scrollTop += 800; // Ajuste o valor conforme necessário para
controlar a quantidade de rolagem
    } else {
        clearInterval(scrollInterval); // Pare o intervalo quando o scroll
chegar ao final

        //chamada para extração dos dados
        extrairDadosContato();
    }
}

// Defina um intervalo para chamar a função de rolagem automaticamente a cada
1000 milissegundos (1 segundo)
const scrollInterval = setInterval(rolarScroll, 3000);
```