Documento Detalhado: Como Funciona o Projeto de Scraping e Processamento de Imóveis

1. Objetivo do Projeto

O objetivo principal deste projeto é automatizar todo o processo de coleta, consolidação, enriquecimento geográfico (geocodificação) e visualização de anúncios imobiliários. O foco está na região de Goiânia/GO, buscando dados de diversas fontes online para permitir análises integradas e a visualização desses dados em mapas interativos.

2. Estrutura do Projeto e Componentes

O projeto é modular, dividido em scripts independentes, cada um responsável por uma etapa específica como scraping, processamento ou visualização. Essa separação facilita a manutenção, atualização e teste de cada parte individualmente.

O script `FacilitaImoveis.py` atua como um scraper focado no site Facilita Imóveis. Ele acessa categorias específicas de imóveis, como venda e aluguel para casas e apartamentos. Para contornar mecanismos anti-bot, utiliza a biblioteca Selenium em conjunto com `undetected_chromedriver`. O script abre as URLs desejadas, aguarda o carregamento completo da página e extrai informações estruturadas, como preço, endereço, tipo, número de quartos, banheiros, vagas, área e o link direto para o anúncio, utilizando a biblioteca BeautifulSoup. As informações extraídas são salvas individualmente em arquivos JSON.

Similarmente, o script `Invest.py` realiza a raspagem de dados do site investt.com.br. Além de usar Selenium e `undetected_chromedriver`, ele incorpora um mecanismo para clicar iterativamente no botão "Ver mais", garantindo que mais imóveis sejam carregados e coletados. Para reduzir a probabilidade de bloqueios, este script também implementa um rotacionamento de user-agent. A extração foca nos dados detalhados apresentados nos cards de imóveis.

O `Olxscaping.py` é dedicado à raspagem de anúncios imobiliários na OLX, especificamente para a região da grande Goiânia. Este script navega pelas páginas de resultados de forma sequencial, cuja configuração pode ser ajustada. Ele coleta os links e os dados principais exibidos nos cards da lista de anúncios, e em seguida, extrai detalhes mais aprofundados de cada anúncio individual. Os dados são então separados e salvos em arquivos específicos, organizados por tipo de imóvel (casas/apartamentos) e finalidade (compra/aluguel).

Para o site zapimoveis.com.br, o projeto utiliza o script `ZapImoveis.py`. Ele é configurado para navegar por diferentes páginas e categorias de imóveis. Uma inicialização robusta do driver Selenium é implementada para mitigar problemas comuns de compatibilidade e detecção. O script captura informações detalhadas de cada imóvel, incluindo endereço completo, preço, número de quartos, vagas de garagem, entre outros dados relevantes.

Por fim, o script `VivaReal.py` realiza o scraping no site vivareal.com.br, buscando categorias de imóveis semelhantes às dos outros scrapers. Este script utiliza filtros disponíveis no site para refinar a busca por região e finalidade do imóvel. O processo envolve a navegação página por página, coletando dados como endereço completo, preço e área do imóvel. Para evitar bloqueios, são aplicados delays entre as requisições.

O `Processamento.py` funciona como o script central de consolidação e unificação. Ele carrega os arquivos JSON gerados por todos os scrapers. Sua função é normalizar os campos, como preço, área e número de quartos, aplicando funções auxiliares para converter formatos (ex: remover R$, normalizar ponto decimal) e extrair valores de strings. Os endereços são limpos para prepará-los para a geocodificação, removendo frases genéricas que possam prejudicar a precisão. A geocodificação é realizada utilizando Selenium para simular uma busca no Nominatim/OpenStreetMap, extraindo latitude e longitude. Para otimizar as requisições e evitar repetições, um cache compartilhado é utilizado. O script respeita os limites de requisição do serviço e insere pausas entre as chamadas. Uma etapa crucial é a detecção e remoção de duplicados na base de dados final. Isso é feito através de heurísticas extensas que comparam imóveis com base na proximidade geográfica (latitude/longitude), similaridade de preço e tamanho/área, e correspondência parcial de endereços. O uso de multiprocessing acelera o processo. Os dados consolidados e limpos são salvos em arquivos JSON padronizados, organizados por categoria, no diretório `resultado`.

Finalmente, o script `Mapa.py` é responsável pela visualização dos dados. Ele consome os arquivos JSON consolidados do diretório de resultados. Para cada imóvel, verifica e obtém as coordenadas geográficas (reutilizando o cache de geocoding). Utilizando a biblioteca Folium, o script gera mapas interativos, posicionando marcadores para cada imóvel. Os marcadores são agrupados por bairro, e um Marker Cluster é aplicado para otimizar a visualização em áreas com muitos imóveis. Detalhes como preço, área, número de quartos e o link do anúncio são adicionados aos popups dos marcadores. Uma legenda colorida dinâmica, baseada em faixas de preço configuráveis, é adicionada ao mapa. O script permite a seleção de múltiplos estilos

de mapa (como Claro/Escuro) e salva as visualizações geradas em arquivos HTML independentes para fácil acesso via navegador.

3. Fluxo Completo do Projeto

O fluxo de execução do projeto se inicia com a **Etapa A: Raspar dados brutos**. Nesta fase, cada um dos scripts de scraping é executado. Cada scraper começa por definir as URLs das categorias de imóveis que serão visitadas. Em seguida, um navegador Chrome "não detectável" é aberto, geralmente em modo headless, para evitar que os sites identifiquem o acesso como automatizado. O scraping é realizado de forma sequencial, respeitando a paginação do site ou acionando botões como "ver mais" para carregar todos os anúncios disponíveis. As informações principais de cada anúncio — como preço, endereço, tipo de imóvel (casa, apartamento), número de quartos, banheiros, vagas de garagem, área e o link direto para o anúncio original — são extraídas. Ao final da raspagem de cada fonte, os dados coletados são salvos em arquivos JSON específicos para cada categoria e fonte, organizados em diretórios dedicados (por exemplo, `olx_data`, `zapimoveis_data`).

Após a coleta, segue-se a **Etapa B: Processamento e unificação**. O script `Processamento.py` é o responsável por esta etapa. Ele inicia carregando todos os arquivos JSON gerados pelos diferentes scrapers. Os dados são então extraídos e normalizados utilizando funções auxiliares para padronizar formatos de preço, área e número de quartos. Os endereços são limpos, removendo termos genéricos que poderiam interferir na precisão da geocodificação. A geocodificação é realizada utilizando Selenium para interagir com o serviço Nominatim/OpenStreetMap, buscando as coordenadas de latitude e longitude para cada endereço. Um cache compartilhado é empregado para evitar a repetição de buscas por endereços já processados, otimizando o tempo e reduzindo o número de requisições online. O script também respeita os limites de requisição do serviço de geocodificação, inserindo pausas entre as chamadas. Uma parte fundamental deste processo é a detecção e remoção de anúncios duplicados na base de dados final. Isso é feito aplicando heurísticas que comparam imóveis com base na proximidade geográfica, similaridade de preço e área, e correspondência parcial de endereços. Essas heurísticas ajudam a fundir registros da mesma propriedade, priorizando aquele com informações mais completas. O uso de multiprocessing acelera a execução desta etapa. Finalmente, os dados consolidados, geocodificados e livres de duplicados são salvos em arquivos JSON padronizados por categoria no diretório `resultado`.

A etapa final é a **Etapa C: Geração de mapas**. O script `Mapa.py` consome os arquivos JSON finalizados localizados no diretório `resultado`. Para cada imóvel listado, o script

verifica e obtém suas coordenadas geográficas, reutilizando o cache de geocoding já populado na etapa anterior. Utilizando a biblioteca Folium, são gerados mapas interativos. Cada imóvel é representado por um marcador no mapa, e esses marcadores são organizados por bairros. Para bairros com muitos imóveis, um Marker Cluster é aplicado, agrupando os marcadores para uma visualização mais limpa e otimizada. Ao clicar em um marcador, um popup exibe detalhes relevantes do imóvel, como preço, área, número de quartos e o link direto para o anúncio. Uma legenda colorida dinâmica é adicionada ao mapa, categorizando os imóveis por faixas de preço, cuja configuração é flexível. O script também permite a seleção de diferentes estilos de mapa (como claro e escuro) e salva as visualizações resultantes em arquivos HTML independentes, que podem ser facilmente visualizados em qualquer navegador web.

4. Aspectos Técnicos e Detalhes Importantes

4.1 Selenium e undetected_chromedriver

A utilização de Selenium em conjunto com `undetected_chromedriver` é crucial para este projeto, pois permite contornar os bloqueios anti-bot frequentemente encontrados em sites de grande porte. Estratégias como a rotação de User-Agents e a desativação de certas features automáticas do navegador são empregadas para simular um comportamento mais humano e dificultar a detecção. A inclusão de delays e esperas explícitas no código garante que as páginas web estejam completamente carregadas antes que a extração de dados seja tentada, prevenindo erros causados por elementos ainda não renderizados.

4.2 Normalização e limpeza dos dados

O processo de normalização e limpeza dos dados é essencial para garantir a consistência e a usabilidade do dataset final. Preços são tratados para remover símbolos monetários (R$), pontuação de milhar e garantir o uso correto do ponto decimal. As áreas dos imóveis são extraídas de strings que podem conter unidades de medida (ex: "248,96 m²"). Informações sobre o número de quartos, vagas de garagem e banheiros são extraídas de strings que podem ter formatos variados, mas parcialmente padronizados. Endereços são limpos proativamente, removendo frases genéricas como "Casa para Comprar em" que, se mantidas, poderiam levar a geocodificações incorretas ou imprecisas.

4.3 Geocodificação

A geocodificação é realizada simulando uma busca no Google Maps via Selenium para obter as coordenadas de latitude e longitude a partir dos endereços limpos. Para

evitar a repetição de buscas pelo mesmo endereço e acelerar o processo, um cache é mantido em um arquivo JSON. Quando a geocodificação falha para um determinado endereço, um fallback é utilizado, atribuindo as coordenadas do centro de Goiânia. É mantido um controle rigoroso sobre o número de requisições enviadas ao serviço OpenStreetMap/Nominatim para respeitar seus limites de uso e termos de serviço.

4.4 Duplicidade

A detecção e tratamento de duplicidades é um eixo central para garantir a qualidade do dataset final. Heurísticas avançadas são aplicadas para identificar registros que se referem à mesma propriedade. Essas heurísticas consideram a proximidade geográfica (baseada nas coordenadas de latitude e longitude), a similaridade de preços e tamanho/área dos imóveis, e a correspondência parcial entre os textos dos endereços. O objetivo é fundir múltiplos registros da mesma propriedade em um único, selecionando a entrada que contém as informações mais completas e precisas. Este processo minimiza o ruído e a redundância no banco de dados final, tornando-o mais confiável para análises.