**Automação de Coleta de Dados: escopo do projeto**

**Tabela

Descrição gerada automaticamente**

**Fase 1: Planejamento**

**Objetivo:**

Definir os requisitos detalhados do projeto e criar um cronograma detalhado das atividades.

**Sprint 1: Reunião Inicial e Levantamento de Requisitos**

**Duração:** 1 dia

**Tarefas:**

1. **Agendar reunião inicial com o cliente:** Definir data e hora.
2. **Preparar pauta da reunião:**
   * Objetivos do projeto
   * Funcionalidades desejadas
   * APIs a serem utilizadas
   * Escopo do projeto
   * Requisitos não funcionais (performance, segurança, etc.)
3. **Conduzir a reunião inicial com o cliente:**
   * Documentar todos os requisitos e expectativas.
4. **Analisar requisitos levantados:** Identificar possíveis riscos e dependências.

**Comando:** Agende a reunião inicial e prepare a pauta da reunião.

### **Fase 2: Desenvolvimento Configuração do Ambiente de Desenvolvimento**

**Tarefas da Sprint**

1. **Instalação do Python e pip**
   * Verificar se o Python já está instalado
   * Instalar o Python (se necessário)
   * Verificar se o pip está instalado
2. **Instalação e Configuração do Visual Studio Code**
   * Download e instalação do Visual Studio Code
   * Configuração básica do Visual Studio Code
   * Instalação de extensões recomendadas para Python
3. **Configuração do Ambiente Virtual**
   * Criação de um ambiente virtual
   * Ativação do ambiente virtual
4. **Instalação de Bibliotecas Necessárias**
   * Instalação das bibliotecas necessárias via pip
5. **Configuração do Controle de Versão com Git e GitHub**
   * Instalação do Git
   * Configuração do Git
   * Criação de um repositório no GitHub
   * Conexão do repositório local com o repositório remoto no GitHub
6. **Configuração do PostgreSQL (se necessário)**
   * Instalação do PostgreSQL
   * Configuração básica do PostgreSQL
   * Criação de um banco de dados para o projeto

Host: localhost

Database: automacao\_dados

Port: 5432

Username: postgres

Senha: 1234

### Sprint: Integração com APIs Selecionadas

#### Tarefas da Sprint

1. **Configuração das Credenciais das APIs**
   * Obter e configurar as credenciais para cada API
   * Armazenar as credenciais de forma segura (ex.: arquivo .env)
2. **Integração com a API DataForSEO**
   * Configurar a autenticação
   * Implementar a coleta de dados de tendências e buscas históricas
   * Testar e validar a integração
3. **Integração com a API RapidAPI**
   * Configurar a autenticação
   * Implementar a coleta de dados de tendências e buscas históricas
   * Testar e validar a integração
4. **Integração com a API Zenserp**
   * Configurar a autenticação
   * Implementar a coleta de dados de tendências e buscas históricas
   * Testar e validar a integração
5. **Integração com a API OpenAI**
   * Configurar a autenticação
   * Implementar a geração de conteúdo ou análises adicionais
   * Testar e validar a integração
6. **Integração com a API Shopify**
   * Configurar a autenticação
   * Implementar a coleta de dados de compras efetivadas
   * Testar e validar a integração

### Tarefa 2: Integração com a API DataForSEO

#### Passos para Integração com a API DataForSEO

#### 1. Configurar a Autenticação

1. **Obter Credenciais:**
   * Se você ainda não fez isso, registre-se em [DataForSEO](https://dataforseo.com/) e obtenha suas credenciais de API (API key).
2. **Armazenar Credenciais no Arquivo .env:**
   * No arquivo .env, adicione sua chave de API:

env

Copiar código

DATAFORSEO\_API\_KEY=your\_dataforseo\_api\_key

1. **Carregar Credenciais no Script Python:**
   * No seu script Python, carregue a chave de API:

python

Copiar código

import os

from dotenv import load\_dotenv

load\_dotenv()

dataforseo\_api\_key = os.getenv('DATAFORSEO\_API\_KEY')

#### 2. Implementar a Coleta de Dados de Tendências e Buscas Históricas

1. **Instalar a Biblioteca Requests (se ainda não tiver instalado):**
   * No terminal, digite:

sh

Copiar código

pip install requests

1. **Criar Função para Coletar Dados:**
   * Implemente uma função para fazer requisições à API DataForSEO e coletar dados de tendências e buscas históricas:

python

Copiar código

import requests

import json

def get\_trends\_data(api\_key, keyword, location):

url = "https://api.dataforseo.com/v3/keywords\_data/google\_trends/explore/live"

headers = {

'Content-Type': 'application/json',

'Authorization': f'Basic {api\_key}'

}

payload = {

"keywords": [keyword],

"location\_code": location,

"language\_code": "en"

}

response = requests.post(url, headers=headers, data=json.dumps(payload))

if response.status\_code == 200:

return response.json()

else:

print(f"Error: {response.status\_code}")

return None

# Exemplo de uso:

trends\_data = get\_trends\_data(dataforseo\_api\_key, "example keyword", "2840") # Código de localização para Holanda

print(trends\_data)

1. **Teste e Valide a Integração:**
   * Execute o script e verifique se os dados de tendências estão sendo retornados corretamente.

### Comandos para Integração com a API DataForSEO

1. **Armazene a Chave de API no Arquivo .env:**
   * Adicione a linha no seu arquivo .env:

env

Copiar código

DATAFORSEO\_API\_KEY=your\_dataforseo\_api\_key

1. **Carregue as Credenciais no Script Python:**
   * Adicione o código para carregar as credenciais:

python

Copiar código

import os

from dotenv import load\_dotenv

load\_dotenv()

dataforseo\_api\_key = os.getenv('DATAFORSEO\_API\_KEY')

1. **Instale a Biblioteca Requests:**
   * No terminal, digite:

sh

Copiar código

pip install requests

1. **Crie e Teste a Função para Coletar Dados:**
   * Adicione o seguinte código ao seu script Python:

python

Copiar código

import requests

import json

def get\_trends\_data(api\_key, keyword, location):

url = "https://api.dataforseo.com/v3/keywords\_data/google\_trends/explore/live"

headers = {

'Content-Type': 'application/json',

'Authorization': f'Basic {api\_key}'

}

payload = {

"keywords": [keyword],

"location\_code": location,

"language\_code": "en"

}

response = requests.post(url, headers=headers, data=json.dumps(payload))

if response.status\_code == 200:

return response.json()

else:

print(f"Error: {response.status\_code}")

return None

# Exemplo de uso:

trends\_data = get\_trends\_data(dataforseo\_api\_key, "example keyword", "2840") # Código de localização para Holanda

print(trends\_data)

**Agora, siga os passos acima para integrar a API DataForSEO ao seu projeto. Após concluir, informe para prosseguirmos para a tarefa de integração com a API RapidAPI.**

### Visão Geral do Projeto e Próximas Fases

### Fase 2: Desenvolvimento

#### Sprint: Integração com APIs Selecionadas

1. **Integração com a API DataForSEO**
2. **Integração com a API RapidAPI**
3. **Integração com a API Zenserp**
4. **Integração com a API OpenAI**
5. **Integração com a API Shopify**

### Fase 3: Desenvolvimento do Script de Coleta de Dados

#### Sprint: Desenvolvimento do Script

1. **Desenvolvimento do Script de Coleta de Dados**
2. **Implementação da Lógica de Limpeza e Transformação de Dados**
3. **Implementação da Lógica de Armazenamento de Dados (Excel/PostgreSQL)**
4. **Implementação da Lógica de Notificações e Alertas**

### Fase 4: Testes

#### Sprint: Testes e Validação

1. **Testes Unitários para Funções de Coleta de Dados**
2. **Testes Unitários para Funções de Limpeza e Transformação de Dados**
3. **Testes Unitários para Funções de Armazenamento de Dados**
4. **Testes de Integração com APIs**
5. **Validação dos Dados Coletados**
6. **Ajustes e Correções Baseados nos Testes**

### Fase 5: Implantação

#### Sprint: Implantação e Treinamento

1. **Deploy da Solução em Ambiente de Produção**
2. **Configuração do Cronograma de Execução Automática**
3. **Treinamento do Cliente para Uso da Solução**
4. **Documentação do Projeto**

### Fase 6: Suporte e Manutenção

#### Sprint: Suporte e Atualizações

1. **Suporte Técnico e Correção de Erros**
2. **Atualizações e Melhorias Incrementais**
3. **Monitoramento e Ajustes do Sistema**
4. **Relatórios de Performance e Utilização**
5. DATABASE\_URL=postgresql://postgres:1234@localhost:5432/automacao\_dados

DATAFORSEO\_API\_KEY=b61ed2f42c2a873e

1. RAPIDAPI\_KEY=aab3aeccddmsh4c34c7724936494p18f17ajsn82419ee940a5
2. ZENSERP\_API\_KEY=your\_zenserp\_api\_key
3. #OPENAI\_API\_KEY=sk-proj-RjMeqAWXqbORvRO320PIT3BlbkFJWkOTKtR54eNgBPtbmNqn
4. SHOPIFY\_API\_KEY=your\_shopify\_api\_key
5. #Authorization: Bearer sk-proj-RjMeqAWXqbORvRO320PIT3BlbkFJWkOTKtR54eNgBPtbmNqn

WEB SCRAPLING AMAZON:

import requests

from bs4 import BeautifulSoup

import pandas as pd

import time

import re

def get\_amazon\_bestsellers(url, retries=5, backoff\_factor=0.3):

    headers = {

        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3"

    }

    for i in range(retries):

        response = requests.get(url, headers=headers)

        if response.status\_code == 200:

            break

        elif response.status\_code == 503:

            print(f"Failed to retrieve the page. Status code: 503. Retrying {i+1}/{retries}...")

            time.sleep(backoff\_factor \* (2 \*\* i))  # Exponential backoff

        else:

            print(f"Failed to retrieve the page. Status code: {response.status\_code}")

            return []

    if response.status\_code != 200:

        print(f"Failed to retrieve the page after {retries} retries.")

        return []

    soup = BeautifulSoup(response.content, "html.parser")

    products = []

    # Verificar a estrutura HTML da página para encontrar o seletor correto

    items = soup.select("li.a-carousel-card")  # Ajuste o seletor para os itens de produto

    print(f"Found {len(items)} items.")

    for item in items:

        # Ajuste os seletores conforme necessário

        title = item.select\_one("div.p13n-sc-truncate-desktop-type2")

        price = item.select\_one("span.\_cDEzb\_p13n-sc-price\_3mJ9Z")

        rating = item.select\_one("span.a-icon-alt")

        image = item.select\_one("img.a-dynamic-image")

        link = item.select\_one("a.a-link-normal")

        # Verificação e logs dos elementos encontrados

        if title:

            title = title.get\_text(strip=True)

        else:

            title = "No title"

        print(f"Title: {title}")

        if price:

            price = price.get\_text(strip=True)

            # Separar o símbolo da moeda e o valor

            currency\_symbol = re.findall(r'[^\d.,]+', price)[0]

            value = re.findall(r'[\d.,]+', price)[0]

        else:

            currency\_symbol = "Not Available"

            value = "Not Available"

        print(f"Currency Symbol: {currency\_symbol}, Value: {value}")

        if rating:

            rating = rating.get\_text(strip=True)

        else:

            rating = "No rating"

        print(f"Rating: {rating}")

        if image:

            image\_link = image.get('src')

        else:

            image\_link = "No image link"

        print(f"Image link: {image\_link}")

        if link:

            product\_link = "https://www.amazon.nl" + link.get('href')

        else:

            product\_link = "No product link"

        print(f"Product link: {product\_link}")

        products.append({

            "title": title,

            "currency\_symbol": currency\_symbol,

            "price\_value": value,

            "rating": rating,

            "image\_link": image\_link,

            "product\_link": product\_link

        })

    return products

def save\_to\_excel(products, filename):

    df = pd.DataFrame(products)

    df.to\_excel(filename, index=False)

    print(f"Products saved to {filename}")

if \_\_name\_\_ == "\_\_main\_\_":

    url = "https://www.amazon.nl/gp/new-releases"

    products = get\_amazon\_bestsellers(url)

    save\_to\_excel(products, "amazon\_new\_releases.xlsx")

**SHOPIFY**

## **Perguntas frequentes sobre categorias de compras on-line**

### Qual categoria de compras on-line teve o maior crescimento?

De acordo com as últimas projeções, a categoria de alimentos deverá apresentar o maior crescimento, com um aumento projetado de 24,1%.

### Qual categoria de compras on-line tem o crescimento mais consistente?

A categoria de alimentos tem apresentado crescimento consistente nos últimos quatro anos, com um aumento de 51,6% em 2020, 39,7% em 2021, 13,3% em 2022, 16,0% em 2023 e uma projeção de 24,1% em 2024.

### Qual categoria de compras on-line está crescendo menos?

A projeção é que os produtos de mídia apresentem o menor crescimento, com um aumento de apenas 6,1%.

### Quais são as categorias de compras on-line mais populares nos Estados Unidos?

* Moda (US$ 162,9 bilhões)
* Alimentos (US$ 125,6 bilhões)
* Eletrônicos (US$ 120,1 bilhões)
* Bebidas (US$ 111,6 bilhões)
* Artigos de bricolage e hardware (US$ 108,6 bilhões)
* Móveis (US$ 74,5 bilhões)
* Mídia (US$ 37,7 bilhões)
* Beleza e cuidados pessoais (US$ 28,4 bilhões)
* Brinquedos e hobbies (US$ 21,2 bilhões)
* Bens de luxo (US$ 14,9 bilhões)

Esses números representam as receitas de comércio eletrônico projetadas para essas categorias de compras.

### Quais são as categorias de compras on-line mais populares no mundo?

* Eletrônicos (US$ 922,5 bilhões)
* Moda (US$ 760 bilhões)
* Alimentos (US$ 460,1 bilhões)
* Bebidas (US$ 248,7 bilhões)
* Artigos de bricolage e hardware (US$ 220,2 bilhões)
* Móveis (US$ 220,1 bilhões)
* Mídia (US$ 193,9 bilhões)
* Beleza e cuidados pessoais (US$ 169,6 bilhões)
* Produtos de tabaco (US$ 116,6 bilhões)
* Brinquedos e hobbies (US$ 89,8 bilhões)

Esses números representam as receitas de comércio eletrônico projetadas para essas categorias de compras.

### Qual é a categoria de varejo on-line mais vendida?

De acordo com as últimas projeções, os eletrônicos assumiram a liderança como a categoria de varejo online mais vendida no mundo todo. A previsão é que os consumidores gastem US$ 922,5 bilhões em produtos eletrônicos.

TOP 150 PRODUTOS

https://www.shopify.com/blog/best-dropshipping-products#1