**Настройка регулярного сканирования правовых порталов на обновления**

Сканирование правовых порталов на обновления требует автоматизации с использованием веб-скрапинга или работы с предоставляемыми API (если доступно). Рассмотрим два подхода: **веб-скрапинг** и **использование RSS/JSON-данных**, если портал их поддерживает.

**Шаги для настройки сканирования**

**1. Определение источников**

Примеры правовых порталов в РФ:

* **Официальный интернет-портал правовой информации**: [pravo.gov.ru](http://pravo.gov.ru)
* **Портал Госдумы РФ**: [sozd.duma.gov.ru](https://sozd.duma.gov.ru)
* **ФСТЭК России**: [fstec.ru](http://fstec.ru)
* **Роскомнадзор**: [rkn.gov.ru](https://rkn.gov.ru)

**2. Сбор данных о структуре порталов**

* Изучите страницы, где публикуются обновления:
  + Новости о внесении новых законов.
  + Список новых нормативных актов.
  + Обновления стандартов и методических рекомендаций.
* Используйте **инструменты для анализа страниц**, такие как:
  + Браузерное расширение **XPath Helper**.
  + Инспектор разработчика в Chrome/Firefox (F12 → "Elements" → "Copy XPath").

**3. Написание скрипта для веб-скрапинга**

Используем **Python** и библиотеку **BeautifulSoup** для извлечения данных с портала.

**Пример кода:**

python

Копировать код

import requests

from bs4 import BeautifulSoup

from datetime import datetime

def fetch\_updates():

url = "https://sozd.duma.gov.ru/legislation"

response = requests.get(url)

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

# Найдем элементы с нужными данными (пример для таблицы с обновлениями)

updates = []

rows = soup.find\_all("tr", class\_="document-row")

for row in rows:

doc\_title = row.find("td", class\_="document-title").text.strip()

doc\_date = row.find("td", class\_="document-date").text.strip()

doc\_link = row.find("a", href=True)["href"]

updates.append({

"title": doc\_title,

"date": doc\_date,

"link": f"https://sozd.duma.gov.ru{doc\_link}"

})

return updates

# Пример использования

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

updates = fetch\_updates()

for update in updates:

print(f"Документ: {update['title']} | Дата: {update['date']} | Ссылка: {update['link']}")

**4. Автоматизация с помощью планировщика**

* **На Windows:**
  + Используйте "Планировщик заданий" для запуска скрипта.
  + Создайте задачу, которая будет запускать ваш Python-скрипт по расписанию.
* **На Linux:**
  + Настройте cron для регулярного выполнения:

bash

Копировать код

crontab -e

# Добавить строку для выполнения скрипта ежедневно в 3 утра

0 3 \* \* \* python3 /path/to/your\_script.py

**5. Уведомления об обновлениях**

* **Email-уведомления:** Используйте библиотеку smtplib для отправки уведомлений.

Пример:

python

Копировать код

import smtplib

from email.mime.text import MIMEText

def send\_email(updates):

sender\_email = "your\_email@example.com"

recipient\_email = "recipient@example.com"

subject = "Новые обновления на правовом портале"

body = "\n".join([f"{update['date']}: {update['title']} ({update['link']})" for update in updates])

message = MIMEText(body, "plain")

message["Subject"] = subject

message["From"] = sender\_email

message["To"] = recipient\_email

with smtplib.SMTP("smtp.example.com", 587) as server:

server.starttls()

server.login(sender\_email, "your\_password")

server.sendmail(sender\_email, recipient\_email, message.as\_string())

* **Телеграм-уведомления:** Используйте Telegram API с ботом.

python

Копировать код

import requests

def send\_telegram\_message(updates):

bot\_token = "YOUR\_BOT\_TOKEN"

chat\_id = "YOUR\_CHAT\_ID"

message = "\n".join([f"{update['date']}: {update['title']} ({update['link']})" for update in updates])

url = f"https://api.telegram.org/bot{bot\_token}/sendMessage"

requests.post(url, data={"chat\_id": chat\_id, "text": message})

**6. Обработка изменений**

* Сравнивайте новые данные с уже загруженными в вашу систему.
* Обновляйте базу только для новых или изменённых документов.

**Дополнительно**

* Если портал предоставляет **RSS**, используйте библиотеку feedparser для обработки:

python

Копировать код

import feedparser

def fetch\_rss\_updates():

feed = feedparser.parse("https://example.com/rss")

updates = []

for entry in feed.entries:

updates.append({

"title": entry.title,

"date": entry.published,

"link": entry.link

})

return updates

**Результат**

* Ежедневное или еженедельное сканирование порталов на изменения.
* Автоматическая загрузка новых документов в базу данных.
* Оповещения об изменениях через Email или Telegram.

Если нужен рабочий код для конкретного портала, сообщите!

4o

**Вы сказали:**

давай по шагам что делаем

**ChatGPT сказал:**

ChatGPT

**Пошаговый план настройки автоматического сканирования правовых порталов**

**Шаг 1: Определение целей и источников**

1. **Цели**:
   * Собирать новые законы, нормативные акты, ГОСТы, указы и изменения в реальном времени.
   * Уведомлять об изменениях и добавлять документы в базу знаний.
2. **Источники данных**:
   * Официальный портал правовой информации: [pravo.gov.ru](http://pravo.gov.ru).
   * Госдума РФ: [sozd.duma.gov.ru](https://sozd.duma.gov.ru).
   * Сайты регуляторов (ФСТЭК, Роскомнадзор, ФСБ).

**Шаг 2: Анализ структуры порталов**

1. **Открыть сайт и изучить его структуру**:
   * Какие разделы публикуют обновления?
   * Есть ли RSS или JSON API для получения данных?
2. **Инструменты анализа**:
   * Включить "Инспектор разработчика" в браузере (F12 → "Elements").
   * Найти CSS-селекторы или XPath для элементов (названия, даты, ссылки).
3. **Пример для таблицы с документами**:
   * Найдите классы HTML-элементов, где находятся:
     + Название документа.
     + Дата публикации.
     + Ссылка на документ.

**Шаг 3: Написание скрипта для сбора данных**

1. **Установить зависимости**:
   * Убедитесь, что Python установлен.
   * Установите библиотеки:

bash

Копировать код

pip install beautifulsoup4 requests

1. **Написать скрипт**:
   * Пример для сбора данных с портала:

python

Копировать код

import requests

from bs4 import BeautifulSoup

def fetch\_documents():

url = "https://sozd.duma.gov.ru/legislation"

response = requests.get(url)

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

updates = []

rows = soup.find\_all("tr", class\_="document-row")

for row in rows:

doc\_title = row.find("td", class\_="document-title").text.strip()

doc\_date = row.find("td", class\_="document-date").text.strip()

doc\_link = row.find("a", href=True)["href"]

updates.append({

"title": doc\_title,

"date": doc\_date,

"link": f"https://sozd.duma.gov.ru{doc\_link}"

})

return updates

# Запуск скрипта

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

documents = fetch\_documents()

for doc in documents:

print(f"{doc['date']}: {doc['title']} ({doc['link']})")

**Шаг 4: Автоматизация выполнения**

1. **Настроить регулярное выполнение скрипта**:
   * **На Windows**:
     + Откройте "Планировщик заданий".
     + Создайте задачу для запуска скрипта:

cmd

Копировать код

python C:\path\to\script.py

* + **На Linux**:
    - Используйте cron:

bash

Копировать код

crontab -e

# Выполнять скрипт ежедневно в 3 утра

0 3 \* \* \* python3 /path/to/script.py

**Шаг 5: Уведомления об изменениях**

1. **Email-уведомления**:
   * Используйте smtplib для отправки писем:

python

Копировать код

import smtplib

from email.mime.text import MIMEText

def send\_email(updates):

sender = "your\_email@example.com"

recipient = "recipient@example.com"

body = "\n".join([f"{u['date']}: {u['title']} ({u['link']})" for u in updates])

msg = MIMEText(body, "plain")

msg["Subject"] = "Обновления правовых документов"

msg["From"] = sender

msg["To"] = recipient

with smtplib.SMTP("smtp.example.com", 587) as server:

server.starttls()

server.login("your\_email@example.com", "password")

server.sendmail(sender, recipient, msg.as\_string())

1. **Telegram-уведомления**:
   * Создайте бота в Telegram (используйте BotFather).
   * Отправляйте обновления:

python

Копировать код

import requests

def send\_telegram\_message(updates):

bot\_token = "YOUR\_BOT\_TOKEN"

chat\_id = "YOUR\_CHAT\_ID"

message = "\n".join([f"{u['date']}: {u['title']} ({u['link']})" for u in updates])

url = f"https://api.telegram.org/bot{bot\_token}/sendMessage"

requests.post(url, data={"chat\_id": chat\_id, "text": message})

**Шаг 6: Хранение данных**

1. **Сохраняйте обновления в файл**:
   * Формат JSON:

python

Копировать код

import json

def save\_to\_file(data, filename="updates.json"):

with open(filename, "w", encoding="utf-8") as file:

json.dump(data, file, ensure\_ascii=False, indent=4)

1. **Добавляйте только новые документы**:
   * Сравнивайте с ранее сохранёнными данными.
   * Обновляйте файл только при нахождении новых записей.

**Шаг 7: Тестирование**

1. **Проверьте корректность скрипта**:
   * Убедитесь, что данные собираются без ошибок.
   * Проверьте формат сообщений и уведомлений.
2. **Исправьте ошибки**:
   * Если структура портала изменилась, обновите селекторы или XPath.

**Шаг 8: Запуск системы**

1. **Запустите скрипт на регулярной основе**.
2. **Подключите уведомления** для автоматического информирования о новых документах.

Если хотите, я могу доработать скрипт под конкретный портал или помочь с настройкой уведомлений. Сообщите, что требуется