

LAPORAN DATA SCRAPING

Disusun untuk Memenuhi Matakuliah Data Scraping
Dibimbing oleh Pelsri Ramadar N.S., M.Kom



Kelompok 1

Farrelyno Luhur Priyambodo P. 1123102111

Habibul Halim Cahyono 1123102122

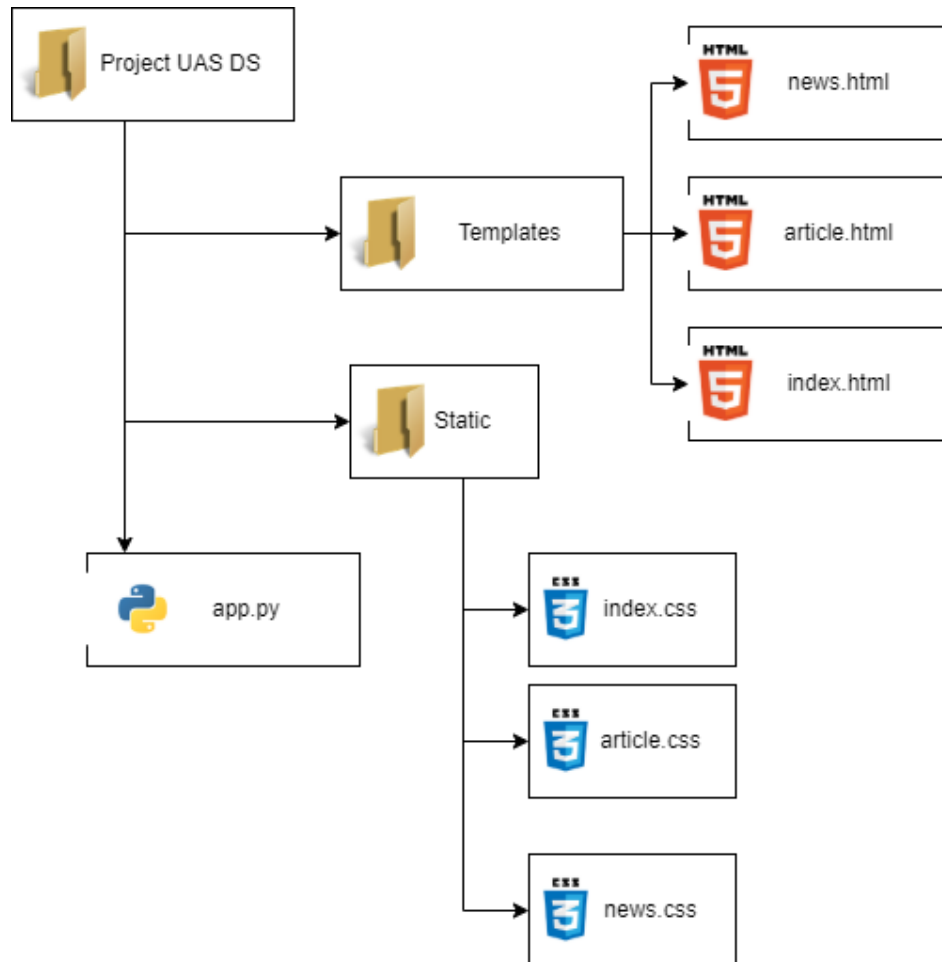
Favian Nurruddin 1123102138

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
SEKOLAH TINGGI ILMU KOMPUTER PGRI BANYUWANGI
2025**

UAS DATA SCRAPING

1. STRUKTUR PROJECT

a) Struktur Folder



b) Code

app.py :

```
from flask import Flask, render_template, request
import requests
import re
from bs4 import BeautifulSoup

app = Flask(__name__)
```

```

@app.route("/")
def home():
    kincir = kincir_articles()
    jagat = jagat_articles()
    game8 = game8_articles()
    eurogamer = eurogamer_articles()

    return render_template("index.html", kincir=kincir, jagat=jagat, game8=game8, eurogamer=eurogamer)

def kincir_articles():
    html_doc = requests.get('https://kincir.com/category/game/')
    soup = BeautifulSoup(html_doc.text, "html.parser")
    popular_area = soup.find(attrs={'class': 'd-lg-grid category__latest-listSingle'})

    data = popular_area.find_all(attrs={'class': 'card__post-landscape'})
    kincir = []
    for item in data:
        img_tag = item.find('img')
        img = img_tag.get('data-lazy-src') or img_tag.get('src')
        url = item.find('a')['href']
        title = item.find('a')['aria-label']
        time = item.find('span').text

        kincir.append({
            'img': img,
            'link': url,
            'title': title,
            'time': time
        })

    return kincir

def jagat_articles():
    html_doc = requests.get('https://jagatplay.com/')
    soup = BeautifulSoup(html_doc.text, "html.parser")
    popular_area = soup.find(attrs={'class': 'ct__main'})

    data = popular_area.find_all(attrs={'class': 'art'})
    jagat = []
    for item in data:
        img = item.find('img')['src']
        url = item.find('a')['href']
        title = item.find('h2').text
        time = item.find('div', {'class': 'art__date'}).text

```

```

        jagat.append({
            'img': img,
            'link': url,
            'title': title,
            'time': time
        })
    return jagat

def game8_articles():
    html_doc = requests.get('https://game8.co/articles/reviews')
    soup = BeautifulSoup(html_doc.text, "html.parser")
    popular_area = soup.find(attrs={'class': 'p-articleListItem__reviews'})

    data = popular_area.find_all(attrs={'class': 'p-articleListItem'})
    game8 = []
    for item in data:
        img = item.find('img')['data-src']
        url = item.find('a')['href']
        title = item.find('div', {'class': 'p-articleListItem__title'}).text
        time = item.find('time').text

        if url.startswith('/'):
            url = f'https://game8.co{url}'

        game8.append({
            'img': img,
            'link': url,
            'title': title,
            'time': time
        })

    return game8

def eurogamer_articles():
    html_doc = requests.get('https://www.eurogamer.net/news')
    soup = BeautifulSoup(html_doc.text, "html.parser")
    popular_area = soup.find(attrs={'class': 'archive__items'})

    data = popular_area.find_all(attrs={'class': 'archive__item'})
    eurogamer = []
    for item in data:
        img = item.find('img')['src']
        url = item.find('a')['href']
        title = item.find('h2').text
        time = item.find('time').text

```

```

        eurogamer.append({
            'img': img,
            'link': url,
            'title': title,
            'time': time
        })
    return eurogamer

@app.route("/news/<source>")
def news(source):
    articles = {
        "kincir": kincir_articles,
        "jagat": jagat_articles,
        "game8": game8_articles,
        "eurogamer": eurogamer_articles
    }
    if source in articles:
        article_data = articles[source]()
        return render_template("news.html", source=source, articles=article_data)

def fetch_article(url, source):
    url = request.args.get('url')
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    article = {}

    if source == "kincir":
        data = soup.find(attrs={'class': 'article'})
        if data:
            title = data.find('h1').text
            time = data.find('div', class_='article__info-date info').text
            thumb = data.find('div', class_='inner').find('img')['src']
            konten_div = data.find('div', class_='content')
            if konten_div:
                konten = konten_div.find_all(['p', 'h2', 'h3', 'img'])
                konten_list = []
                for tag in konten:
                    if tag.name == 'img':
                        img_src = tag.get('src', "") or tag.get('data-lazy-src', "")
                        if "cloudfront.net" in img_src:
                            konten_list.append(str(tag))
                    else:
                        konten_list.append(str(tag))
            else:
                konten_list = []

```

```

        article = {
            'title': title,
            'time': time,
            'thumb': thumb,
            'konten': konten_list
        }

elif source == "jagat":
    data = soup.find(attrs={'class': 'jgpost__box'})
    if data:
        title = data.find('h1').text
        time = data.find('div', class_='jgauthor__posted').text

        thumb_div = data.find('div', class_='jgpost__feat-img')
        if thumb_div and 'style' in thumb_div.attrs:
            style = thumb_div['style']
            match = re.search(r"url\(['\"]?(.*?)['\"]?)", style)
            thumb = match.group(1) if match else None
        else:
            thumb = None

        figure = data.find('picture').find('img')['src']
        konten_div = data.find('div', class_='jgpost__content')
        if konten_div:
            konten = konten_div.find_all(['p', 'h5', 'h3', 'figure'])
            konten_list = [str(tag) for tag in konten]
        else:
            konten_list = []

        article = {
            'title': title,
            'time': time,
            'thumb': thumb,
            'figure': figure,
            'konten': konten_list
        }

elif source == "game8":
    data = soup.find(attrs={'class': 'l-2col__main__article'})
    if data:
        title = data.find('h1').text
        time = data.find('time').text
        img_tag = data.find('p').find('img')
        if img_tag:
            thumb = img_tag['src']

```

```

else:
    thumb_div = data.find('div', class_='p-article__gameScore__head')
    if thumb_div and 'style' in thumb_div.attrs:
        style_content = thumb_div['style']
        match = re.search(r'url\(["\"]?(.*?)["\"]?)', style_content)
        if match:
            thumb = match.group(1)
    else:
        thumb = None

konten_div = data.find('div', class_='p-articleBody article-style-wrapper')
if konten_div:
    konten = konten_div.find_all(['p', 'h2', 'h3', 'iframe'])
    konten_list = []
    for tag in konten:
        if tag.name == 'iframe' and tag.get('data-src'):
            tag['src'] = tag['data-src']
            konten_list.append(str(tag))
else:
    konten_list = []

article = {
    'title': title,
    'time': time,
    'thumb': thumb,
    'konten': konten_list
}

elif source == "eurogamer":
    data = soup.find(attrs={'class': 'page_content'})
    if data:
        title = data.find('h1').text
        time = data.find('time').text
        thumb = data.find('img')['src']
        konten_div = data.find('div', class_='article_body')
        if konten_div:
            konten = konten_div.find_all('p')
            konten_list = [str(tag) for tag in konten]
        else:
            konten_list = []

    article = {
        'title': title,
        'time': time,
        'thumb': thumb,

```

```

        'konten': konten_list
    }

    return article

@app.route("/article/<source>")
def detail_article(source):
    url = request.args.get('url')
    article = fetch_article(url, source)
    return render_template("article.html", articles={source: article})

if __name__ == "__main__":
    app.run(debug=True)

```

Penjelasan kode app.py

- `from flask import Flask, render_template, request` : Mengimpor pustaka Flask yang digunakan untuk membuat aplikasi web, `render_template` untuk merender template HTML, dan `request` untuk menangani permintaan HTTP.
- `import requests` : Mengimpor pustaka `requests` untuk melakukan permintaan HTTP.
- `import re` : Mengimpor modul `re` untuk menggunakan ekspresi reguler.
- `from bs4 import BeautifulSoup` : Mengimpor `BeautifulSoup` dari pustaka `bs4` untuk mem-parsing HTML.
- `app = Flask(__name__)` : Membuat instansi aplikasi Flask.
- `@app.route("/")` : Mendefinisikan route untuk halaman utama (/) dan mendefinisikan fungsi `home`.
- `def home():` : Fungsi `home` untuk mengatur tampilan halaman utama.
- `kincir = kincir_articles()` : Memanggil fungsi untuk mengambil artikel dari Kincir.
- `jagat = jagat_articles()` : Memanggil fungsi untuk mengambil artikel dari JagatPlay.
- `game8 = game8_articles()` : Memanggil fungsi untuk mengambil artikel dari Game8.
- `eurogamer = eurogamer_articles()` : Memanggil fungsi untuk mengambil artikel dari Eurogamer.
- `return render_template("index.html", kincir=kincir, jagat=jagat, game8=game8, eurogamer=eurogamer)` : Mengirimkan data artikel ke template `index.html` untuk dirender.

- `def kincir_articles():` : Mendefinisikan fungsi untuk mengambil artikel dari situs Kincir.
- `html_doc = requests.get('https://kincir.com/category/game/')` : Mengambil HTML dari situs Kincir.
- `soup = BeautifulSoup(html_doc.text, "html.parser")` : Mem-parsing HTML menggunakan BeautifulSoup.
- `popular_area = soup.find(attrs={'class': 'd-lg-grid category__latest-listSingle'})` : Mencari elemen HTML yang berisi artikel populer.
- `data = popular_area.find_all(attrs={'class': 'card__post-landscape'})` : Mengambil semua elemen HTML yang berisi artikel.
- `kincir = []` : Membuat list kosong untuk menyimpan data artikel.
- `for item in data:` : Melakukan iterasi untuk setiap artikel.
- `img_tag = item.find('img')` : Mengambil tag gambar dari artikel.
- `img = img_tag.get('data-lazy-src') or img_tag.get('src')` : Mengambil URL gambar dari tag gambar.
- `url = item.find('a')['href']` : Mengambil URL artikel.
- `title = item.find('a')['aria-label']` : Mengambil judul artikel.
- `time = item.find('span').text` : Mengambil waktu artikel.
- `kincir.append({'img': img, 'link': url, 'title': title, 'time': time})` : Menambahkan data artikel ke dalam list.
- `return kincir` : Mengembalikan list artikel dari situs Kincir.
- `def jagat_articles():` : Mendefinisikan fungsi untuk mengambil artikel dari situs JagatPlay.
- `html_doc = requests.get('https://jagatplay.com/')` : Mengambil HTML dari situs JagatPlay.
- `soup = BeautifulSoup(html_doc.text, "html.parser")` : Mem-parsing HTML menggunakan BeautifulSoup.
- `popular_area = soup.find(attrs={'class': 'ct__main'})` : Mencari elemen HTML yang berisi artikel populer.

- `data = popular_area.find_all(attrs={'class': 'art'})` : Mengambil semua elemen HTML yang berisi artikel.
- `jagat = []` : Membuat list kosong untuk menyimpan data artikel.
- `for item in data:` : Melakukan iterasi untuk setiap artikel.
- `img = item.find('img')['src']` : Mengambil URL gambar dari artikel.
- `url = item.find('a')['href']` : Mengambil URL artikel.
- `title = item.find('h2').text` : Mengambil judul artikel.
- `time = item.find('div', {'class': 'art__date'}).text` : Mengambil waktu artikel.
- `jagat.append({'img': img, 'link': url, 'title': title, 'time': time})` : Menambahkan data artikel ke dalam list.
- `return jagat` : Mengembalikan list artikel dari situs JagatPlay.
- `def game8_articles():` : Mendefinisikan fungsi untuk mengambil artikel dari situs Game8.
- `html_doc = requests.get('https://game8.co/articles/reviews')` : Mengambil HTML dari situs Game8.
- `soup = BeautifulSoup(html_doc.text, "html.parser")` : Mem-parsing HTML menggunakan BeautifulSoup.
- `popular_area = soup.find(attrs={'class': 'p-articleListItem__reviews'})` : Mencari elemen HTML yang berisi artikel populer.
- `data = popular_area.find_all(attrs={'class': 'p-articleListItem'})` : Mengambil semua elemen HTML yang berisi artikel.
- `game8 = []` : Membuat list kosong untuk menyimpan data artikel.
- `for item in data:` : Melakukan iterasi untuk setiap artikel.
- `img = item.find('img')['data-src']` : Mengambil URL gambar dari artikel.
- `url = item.find('a')['href']` : Mengambil URL artikel.
- `title = item.find('div', {'class': 'p-articleListItem__title'}).text` : Mengambil judul artikel.
- `time = item.find('time').text` : Mengambil waktu artikel.

- `if url.startswith('/') :` : Memeriksa apakah URL dimulai dengan "/" untuk melengkapi URL.
- `url = f'https://game8.co{url}' :` Melengkapi URL artikel.
- `game8.append({'img': img, 'link': url, 'title': title, 'time': time}) :` Menambahkan data artikel ke dalam list.
- `return game8 :` Mengembalikan list artikel dari situs Game8.
- `def eurogamer_articles(): :` Mendefinisikan fungsi untuk mengambil artikel dari situs Eurogamer.
- `html_doc = requests.get('https://www.eurogamer.net/news') :` Mengambil HTML dari situs Eurogamer.
- `soup = BeautifulSoup(html_doc.text, "html.parser") :` Mem-parsing HTML menggunakan BeautifulSoup.
- `popular_area = soup.find(attrs={'class': 'archive__items'}) :` Mencari elemen HTML yang berisi artikel populer.
- `data = popular_area.find_all(attrs={'class': 'archive__item'}) :` Mengambil semua elemen HTML yang berisi artikel.
- `eurogamer = [] :` Membuat list kosong untuk menyimpan data artikel.
- `for item in data: :` Melakukan iterasi untuk setiap artikel.
- `img = item.find('img')['src'] :` Mengambil URL gambar dari artikel.
- `url = item.find('a')['href'] :` Mengambil URL artikel.
- `title = item.find('h2').text :` Mengambil judul artikel.
- `time = item.find('time').text :` Mengambil waktu artikel.
- `eurogamer.append({'img': img, 'link': url, 'title': title, 'time': time}) :` Menambahkan data artikel ke dalam list.
- `return eurogamer :` Mengembalikan list artikel dari situs Eurogamer.
- `def fetch_article(url, source): :` Mendefinisikan fungsi untuk mengambil detail artikel berdasarkan URL dan sumber yang diberikan.
- `url = request.args.get('url') :` Mengambil URL dari permintaan HTTP.

- `response = requests.get(url)` : Melakukan permintaan HTTP untuk mengambil konten artikel.
- `soup = BeautifulSoup(response.text, 'html.parser')` : Mem-parsing konten HTML menggunakan BeautifulSoup.
- `article = {}` : Menginisialisasi dictionary kosong untuk menyimpan data artikel.
- `if source == "kincir":` : Memeriksa apakah sumber artikel adalah Kincir.
- `data = soup.find(attrs={'class': 'article'})` : Mencari elemen HTML yang berisi artikel Kincir.
- `title = data.find('h1').text` : Mengambil judul artikel Kincir.
- `time = data.find('div', class_='article__info-date info').text` : Mengambil waktu artikel Kincir.
- `thumb = data.find('div', class_='inner').find('img')['src']` : Mengambil URL gambar thumbnail artikel Kincir.
- `konten_div = data.find('div', class_='content')` : Mencari elemen HTML yang berisi konten artikel Kincir.
- `if konten_div:` : Memeriksa apakah elemen konten ada.
- `konten = konten_div.find_all(['p', 'h2', 'h3', 'img'])` : Mengambil semua elemen konten artikel Kincir.
- `konten_list = []` : Membuat list kosong untuk menyimpan konten artikel.
- `for tag in konten:` : Melakukan iterasi untuk setiap elemen konten.
- `if tag.name == 'img':` : Memeriksa apakah elemen adalah gambar.
- `img_src = tag.get('src', "") or tag.get('data-lazy-src', "")` : Mengambil URL gambar dari elemen gambar.
- `if "cloudfront.net" in img_src:` : Memeriksa apakah URL gambar berasal dari cloudfront.net.
- `konten_list.append(str(tag))` : Menambahkan elemen gambar ke dalam list konten.
- `else:` : Jika elemen bukan gambar.
- `konten_list.append(str(tag))` : Menambahkan elemen teks ke dalam list konten.
- `article = {'title': title, 'time': time, 'thumb': thumb, 'konten': konten_list}` : Membuat dictionary untuk menyimpan data artikel Kincir.

- `elif source == "jagat":` : Memeriksa apakah sumber artikel adalah JagatPlay.
- `data = soup.find(attrs={'class': 'jgpost__box'})` : Mencari elemen HTML yang berisi artikel JagatPlay.
- `title = data.find('h1').text` : Mengambil judul artikel JagatPlay.
- `time = data.find('div', class_='jgauthor__posted').text` : Mengambil waktu artikel JagatPlay.
- `thumb_div = data.find('div', class_='jgpost__feat-img')` : Mencari elemen HTML yang berisi gambar thumbnail.
- `if thumb_div and 'style' in thumb_div.attrs:` : Memeriksa apakah elemen thumbnail ada dan memiliki atribut style.
- `style = thumb_div['style']` : Mengambil nilai atribut style dari elemen thumbnail.
- `match = re.search(r"url\(['\"]?(.*?)['\"]?\)", style)` : Menggunakan ekspresi reguler untuk mencari URL gambar dalam atribut style.
- `thumb = match.group(1) if match else None` : Mengambil URL gambar dari hasil pencarian ekspresi reguler.
- `else:` : Jika elemen thumbnail tidak ada atau tidak memiliki atribut style.
- `thumb = None` : Mengatur thumbnail menjadi None.
- `figure = data.find('picture').find('img')['src']` : Mengambil URL gambar dari elemen picture.
- `konten_div = data.find('div', class_='jgpost__content')` : Mencari elemen HTML yang berisi konten artikel JagatPlay.
- `if konten_div:` : Memeriksa apakah elemen konten ada.
- `konten = konten_div.find_all(['p', 'h5', 'h3', 'figure'])` : Mengambil semua elemen konten artikel JagatPlay.
- `konten_list = [str(tag) for tag in konten]` : Menambahkan setiap elemen konten ke dalam list konten.
- `else:` : Jika elemen konten tidak ada.
- `konten_list = []` : Mengatur list konten menjadi kosong.

- `article = {'title': title, 'time': time, 'thumb': thumb, 'figure': figure, 'konten': konten_list}` : Membuat dictionary untuk menyimpan data artikel JagatPlay.
- `elif source == "game8":` : Memeriksa apakah sumber artikel adalah Game8.
- `data = soup.find(attrs={'class': 'l-2col__main__article'})` : Mencari elemen HTML yang berisi artikel Game8.
- `title = data.find('h1').text` : Mengambil judul artikel Game8.
- `time = data.find('time').text` : Mengambil waktu artikel Game8.
- `img_tag = data.find('p').find('img')` : Mencari elemen gambar dalam paragraf.
- `if img_tag:` : Memeriksa apakah elemen gambar ada.
- `thumb = img_tag['src']` : Mengambil URL gambar dari elemen gambar.
- `else:` : Jika elemen gambar tidak ada.
- `thumb_div = data.find('div', class_='p-article__gameScore__head')` : Mencari elemen HTML yang berisi gambar thumbnail.
- `if thumb_div and 'style' in thumb_div.attrs and 'background-image' in thumb_div['style']:` : Memeriksa apakah elemen thumbnail ada dan memiliki atribut style dengan background-image.
- `style_content = thumb_div['style']` : Mengambil nilai atribut style dari elemen thumbnail.
- `match = re.search(r'url\(["\']?(.*?)["\']?\)', style_content)` : Menggunakan ekspresi reguler untuk mencari URL gambar dalam atribut style.
- `if match:` : Memeriksa apakah pencarian ekspresi reguler berhasil.
- `thumb = match.group(1)` : Mengambil URL gambar dari hasil pencarian ekspresi reguler.
- `else:` : Jika pencarian ekspresi reguler tidak berhasil.
- `thumb = None` : Mengatur thumbnail menjadi None.
- `konten_div = data.find('div', class_='p-articleBody article-style-wrapper')` : Mencari elemen HTML yang berisi konten artikel Game8.
- `if konten_div:` : Memeriksa apakah elemen konten ada.

- `konten = konten_div.find_all(['p', 'h2', 'h3', 'iframe'])` : Mengambil semua elemen konten artikel Game8.
- `konten_list = []` : Membuat list kosong untuk menyimpan konten artikel.
- `for tag in konten:` : Melakukan iterasi untuk setiap elemen konten.
- `if tag.name == 'iframe' and tag.get('data-src'):` : Memeriksa apakah elemen adalah iframe dan memiliki atribut data-src.
- `tag['src'] = tag['data-src']` : Mengatur atribut src dari iframe dengan nilai data-src.
- `konten_list.append(str(tag))` : Menambahkan elemen konten ke dalam list konten.
- `else:` : Jika elemen bukan iframe.
- `konten_list.append(str(tag))` : Menambahkan elemen teks ke dalam list konten.
- `article = {'title': title, 'time': time, 'thumb': thumb, 'konten': konten_list}` : Membuat dictionary untuk menyimpan data artikel Game8.
- `elif source == "eurogamer":` : Memeriksa apakah sumber artikel adalah Eurogamer.
- `data = soup.find(attrs={'class': 'page_content'})` : Mencari elemen HTML yang berisi artikel Eurogamer.
- `title = data.find('h1').text` : Mengambil judul artikel Eurogamer.
- `time = data.find('time').text` : Mengambil waktu artikel Eurogamer.
- `thumb = data.find('img')['src']` : Mengambil URL gambar thumbnail artikel Eurogamer.
- `konten_div = data.find('div', class_='article_body')` : Mencari elemen HTML yang berisi konten artikel Eurogamer.
- `if konten_div:` : Memeriksa apakah elemen konten ada.
- `konten = konten_div.find_all('p')` : Mengambil semua elemen konten artikel Eurogamer.
- `konten_list = [str(tag) for tag in konten]` : Menambahkan setiap elemen konten ke dalam list konten.
- `else:` : Jika elemen konten tidak ada.
- `konten_list = []` : Mengatur list konten menjadi kosong.

- `article = {'title': title, 'time': time, 'thumb': thumb, 'konten': konten_list}` : Membuat dictionary untuk menyimpan data artikel Eurogamer.
- `return article` : Mengembalikan dictionary artikel yang berisi data artikel.
- `@app.route("/article/<source>")` : Mendefinisikan route untuk menampilkan detail artikel dari sumber yang diberikan.
- `def detail_article(source):` : Mendefinisikan fungsi untuk menampilkan detail artikel.
- `url = request.args.get('url')` : Mengambil URL dari permintaan HTTP.
- `article = fetch_article(url, source)` : Memanggil fungsi `fetch_article` untuk mengambil detail artikel berdasarkan URL dan sumber.
- `return render_template("article.html", articles={source: article})` : Mengirimkan data artikel ke template `article.html` untuk dirender.
- `@app.route("/news/<source>")` : Mendefinisikan route untuk menampilkan artikel berdasarkan sumber yang diberikan.
- `def news(source):` : Mendefinisikan fungsi untuk menampilkan artikel berdasarkan sumber.
- `articles = {"kincir": kincir_articles, "jagat": jagat_articles, "game8": game8_articles, "eurogamer": eurogamer_articles}` : Membuat dictionary yang berisi fungsi untuk masing-masing sumber artikel.
- `if source in articles:` : Memeriksa apakah sumber artikel ada dalam dictionary `articles`.
- `article_data = articles[source]()` : Memanggil fungsi yang sesuai dengan sumber artikel dan mengambil data artikel.
- `return render_template("news.html", source=source, articles=article_data)` : Mengirimkan data artikel ke template `news.html` untuk dirender.
- `if __name__ == "__main__":` : Memeriksa apakah skrip ini dijalankan sebagai program utama.
- `app.run(debug=True)` : Menjalankan aplikasi Flask dalam mode debug.

index.html :

```
<header>
  <nav class="navbar navbar-expand-sm navbar-dark fixed-top justify-content-center">
    <a href="/"><h1>GAMENEWS</h1></a>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/news/kincir" target="_blank">Kincir</a></li>
      <li><a href="/news/jagat" target="_blank">JagatPlay</a></li>
      <li><a href="/news/game8" target="_blank">Game8</a></li>
      <li><a href="/news/eurogamer" target="_blank">EuroGamer</a></li>
    </ul>
    <div class="search-bar">
      <input type="text" placeholder="Search">
      <button>Search</button></div>
    </nav>
  </header>
  <div class="hero">
    
    <div class="hero-text">
      "If There's Hole, There's Goal"
    </div>
  </div>

  <div class="content">
    <div class="news-card">
      
      <div class="news-details">
        <h2>Kincir - Your Gaming News</h2>
        <p>Stay updated with the latest game news, reviews, and tips.</p>
        <div class="footer">
          <span>https://kincir.com/category/game/</span>
          <a href="/news/kincir">Lihat Situs</a>
        </div>
      </div>
    </div>
    <div class="news-card">
      
      <div class="news-details">
        <h2>Jagat Play - Gaming at its best!</h2>
        <p>Jagat Play is gaming part of Jagat Review who share, discuss, and review latest AAA and indie games for fun!</p>
        <div class="footer">
          <span>https://jagatplay.com/</span>
          <a href="/news/jagat">Lihat Situs</a>
        </div>
      </div>
    </div>
  </div>
```

```

</div>
<div class="news-card">
  
  <div class="news-details">
    <h2>Game8 - The Top Gaming and App Walkthroughs</h2>
    <p>Game8 delivers the latest in gaming news, walkthrough information, and other useful tools</p>
    <div class="footer">
      <span>https://game8.co/articles/reviews</span>
      <a href="/news/game8">Lihat Situs</a>
    </div>
  </div>
</div>
</div>
<div class="news-card">
  
  <div class="news-details">
    <h2>EuroGamer - Your Go-To Video Game News</h2>
    <p>Your trusted source for video game news, reviews and guides, since 1999.</p>
    <div class="footer">
      <span>https://www.eurogamer.net/news</span>
      <a href="/news/eurogamer">Lihat Situs</a>
    </div>
  </div>
</div>
</div>
</div>
</div>

```

Penjelasan kode index.html

- Pada class “navbar”, bagian ini dapat digunakan dalam menu navigasi atau daftar sumber daya eksternal di halaman web. Ketika pengguna mengklik salah satu tautan, halaman web yang sesuai akan dibuka di tab atau jendela browser baru
- Pada class “content”, bagian ini digunakan dalam halaman web yang menampilkan card berita atau informasi tentang situs web lain. Card berita ini dapat menampilkan gambar, judul, deskripsi, dan tautan ke situs web lain.

article.html :

```

<div class="container mt-5 pt-4">
  <div>
    <h2>{{ article.title }}</h2>
    <p>{{ article.time }}</p>
    
  </div>

```

```

        {% for content in article.konten %}
            {{ content|safe }}
        {% endfor %}
    </div>
</div>
{% endfor %}
</div>

```

Penjelasan kode article.html

- `<h2>{{ article.title }}</h2>`: Menampilkan judul artikel yang diambil dari data dinamis `article.title`.
- `<p>{{ article.time }}</p>`: Menampilkan waktu publikasi artikel yang diambil dari data dinamis `article.time`.
- ``: Menampilkan gambar thumbnail artikel dari URL yang diambil dari data dinamis `article.thumb`.
- `<div>{% for content in article.konten %}{{ content|safe }}{% endfor %}</div>`: Bagian ini menggunakan loop for untuk iterasi melalui setiap elemen dalam `article.konten`. `{{ content|safe }}` menampilkan konten artikel dengan filter `safe` untuk memastikan HTML diinterpretasikan dengan benar.

news.html :

```

<section class="container">
    <h1 class="title">{{ source | upper }} NEWS</h1>
    <div class="container flex-container">
        <div class="row flex-item">
            {% for article in articles %}
                <div class="article-row">
                    
                    <h5>{{ article['title']|safe }}</h5>
                    <p>{{ article['time']|safe }}</p>
                    <a href="{{ url_for('detail_article', source=source, url=article['link']) }}" class="btn btn-
primary">Baca Selengkapnya</a>
                </div>
            {% endfor %}
        </div>
    </div>
</section>

```

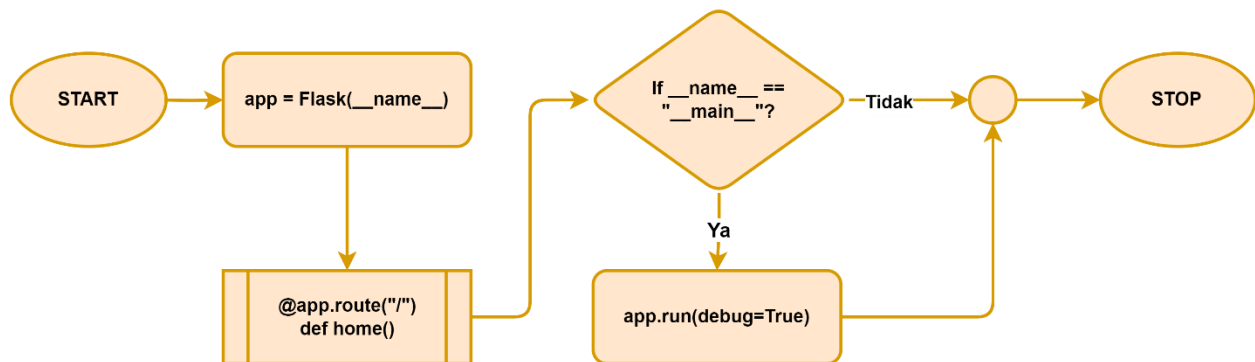
Penjelasan kode news.html

- `<h1 class="title">{{ source | upper }} NEWS</h1>`: Menampilkan judul berita dengan nama sumber (`source`) yang diubah menjadi huruf besar (menggunakan filter `upper`).

- `{% for article in articles %}`: Looping menggunakan template engine (seperti Jinja2) untuk iterasi melalui setiap artikel dalam list articles.
- `<div class="article-row">`: Membuat div dengan kelas article-row untuk setiap artikel.
- ``: Menampilkan gambar artikel, diambil dari URL `article['img']` dengan filter `safe` untuk memastikan HTML aman.
- `<h5>{{ article['title']|safe }}</h5>`: Menampilkan judul artikel yang diambil dari `article['title']` dengan filter `safe`.
- `<p>{{ article['time']|safe }}</p>`: Menampilkan waktu artikel yang diambil dari `article['time']` dengan filter `safe`.
- `Baca Selengkapnya`: Link untuk membaca artikel selengkapnya. Fungsi `url_for` digunakan untuk membangkitkan URL menuju fungsi `detail_article` dengan parameter `source` dan `url` artikel.
- `{% endfor %}`: Menutup loop for article.

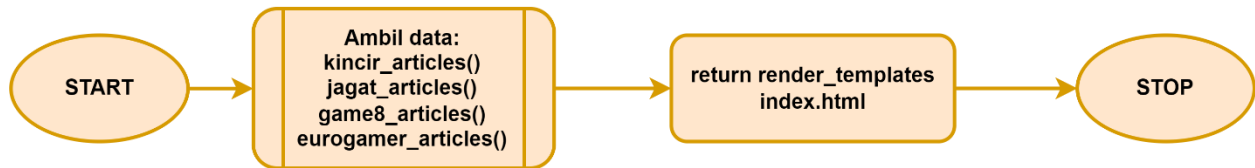
2. FLOWCHART

a) Umum



- Memulai koding
- Menginisiasi system flask
- Membuka rute URL root dan mengaktifkan fungsi home ke halaman index
- Mengecek apakah skrip python dijalankan secara langsung atau diimpor sebagai modul.
- Jika bisa, maka server flask akan berjalan dan mode DEBUG diaktifkan.
- Jika tidak, maka system akan berhenti.
- Mengakhiri koding.

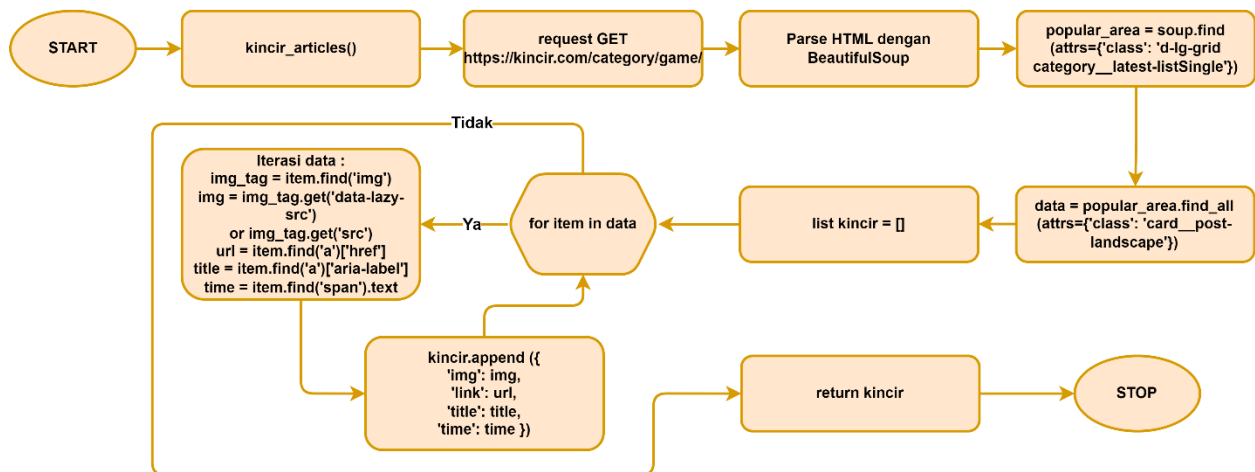
b) Function Home



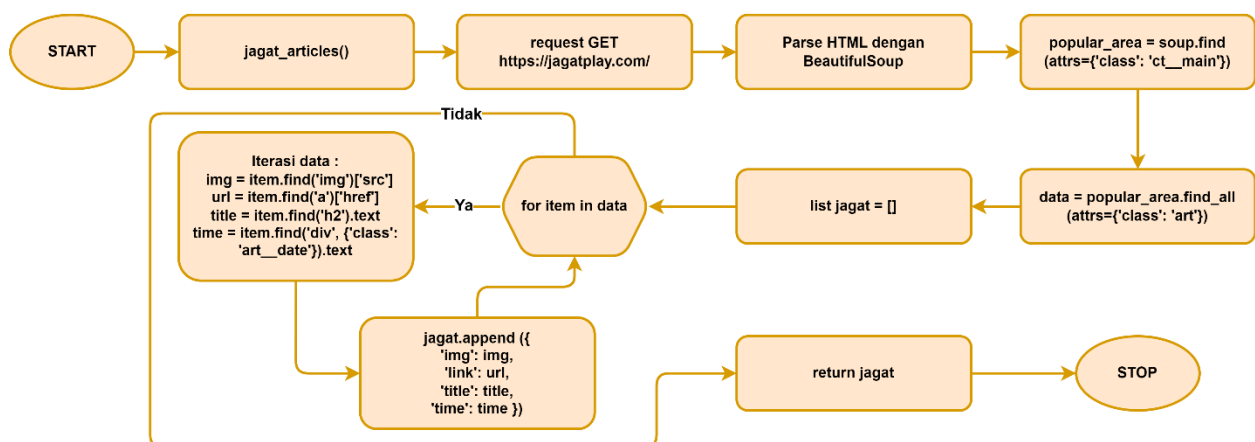
- Memulai koding
- Function akan mengambil data dari empat sumber yang berbeda, yaitu kincir_articles(), jagat_articles(), game8_articles(), dan eurogamer_articles().
- Setelah data diambil dari sumber-sumber tersebut, fungsi akan mengembalikan hasil render template ke index.html
- Mengakhiri function

c) Function kincir_articles, jagat_articles, game8_article, eurogamer_articles

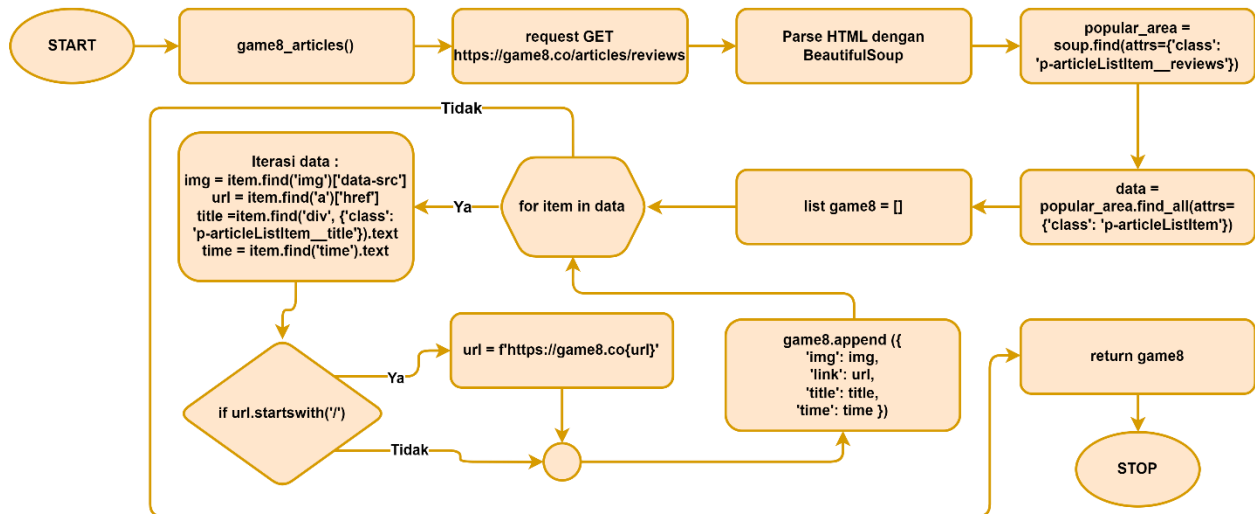
Kincir_articles



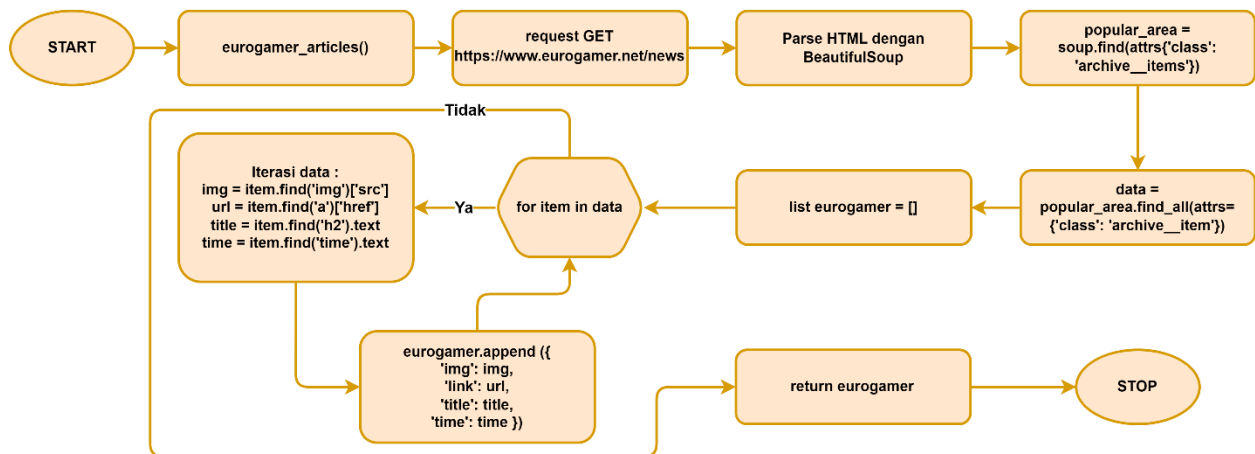
Jagat_articles



Game8_articles



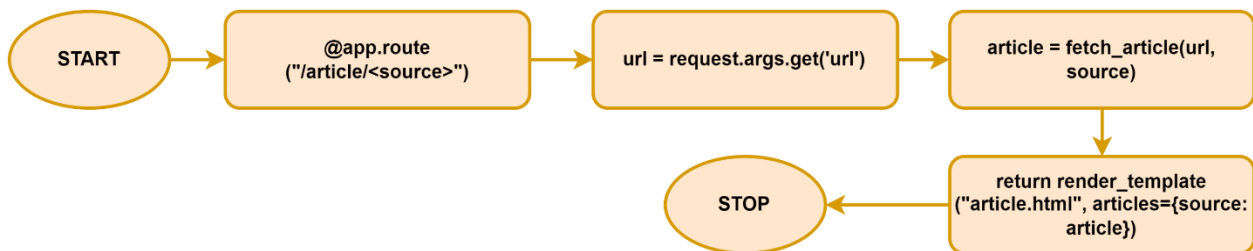
Eurogamer_articles



Dikarenakan proses ke empat fungsi tersebut sama, maka akan dijadikan satu:

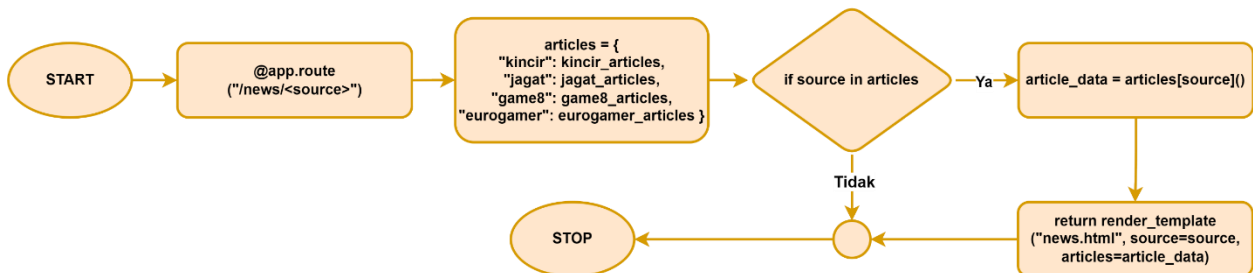
- Memulai proses koding
- Memanggil masing-masing fungsi untuk dijalankan
- Melakukan permintaan GET ke URL masing-masing artikel
- Konten HTML dari permintaan GET diuraikan menggunakan BeautifulSoup.
- Fungsi mencari elemen dengan class tertentu dalam popular_area
- Semua elemen dengan class yang ditentukan yang ditemukan dalam popular_area diambil dan disimpan dalam data.
- Membuat list kosong dari masing-masing artikel untuk diinisialisasi
- Memulai loop untuk mengekstrak item dalam data dan memindahkan ke dalam list, seperti gambar, url, judul artikel, dan tanggal
- Setelah itu mengembalikan list masing-masing artikel web

d) Function News



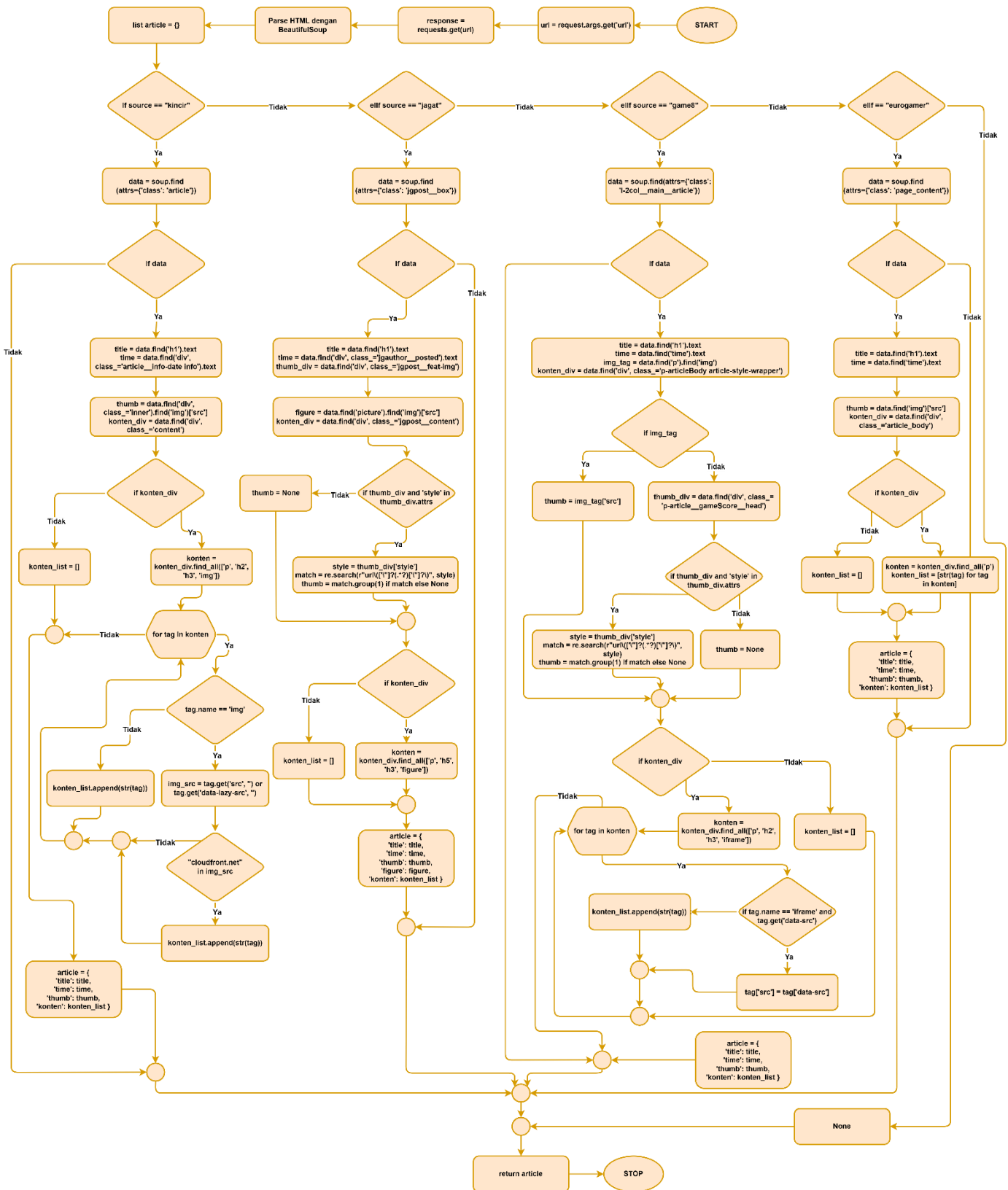
- Memulai proses
- Mengarahkan ke rute `"/news/<source>"`
- Mendefinisikan sebuah kamus (dictionary) dengan artikel-artikel dari sumber kincir, jagat, game8, dan eurogamer
- Mengecek apakah sumber (source) ada di dalam dictionary "articles"
- Jika iya, maka akan memanggil fungsi dari dictionary untuk mendapatkan data source artikel, setelah itu mengembalikan template render ke news.html dengan data dari sumber yang telah diambil tadi.
- Jika tidak, maka langsung menuju ke akhir
- Akhir proses

e) Function Detail_Article



- Terima parameter url dan source dari argument
- Ambil "url" dari request args
- Parsing HTML dengan beautiful soup
- Inisialisasi dictionary yang kosong
- Mengecek dari sumber apa artikel yang diambil
- Mengecek data yang akan diambil
- Jika ada, maka akan memulai proses pengambilan data dan memasukkannya ke detail artikel
- Jika tidak, maka akan ke proses ke-7
- Mengembalikan dictionary article yang telah diisi
- Akhir proses

f) Function Fetch_Artikel



- Memulai proses
- Permintaan HTTP dengan fungsi `requests.get(url)` menggunakan URL yang diambil dari argumen `requestArgs.get("url")`.
- Variabel `list_article` diinisialisasi dengan nilai 0. Proses parsing HTML menggunakan library BeautifulSoup.
- Pemilihan sumber (source), alur bercabang ke beberapa kondisi:
 - `source == "kincir"`
 - `source == "jagat"`
 - `source == "gamedaim"`
 - `source == "eurogamer"`

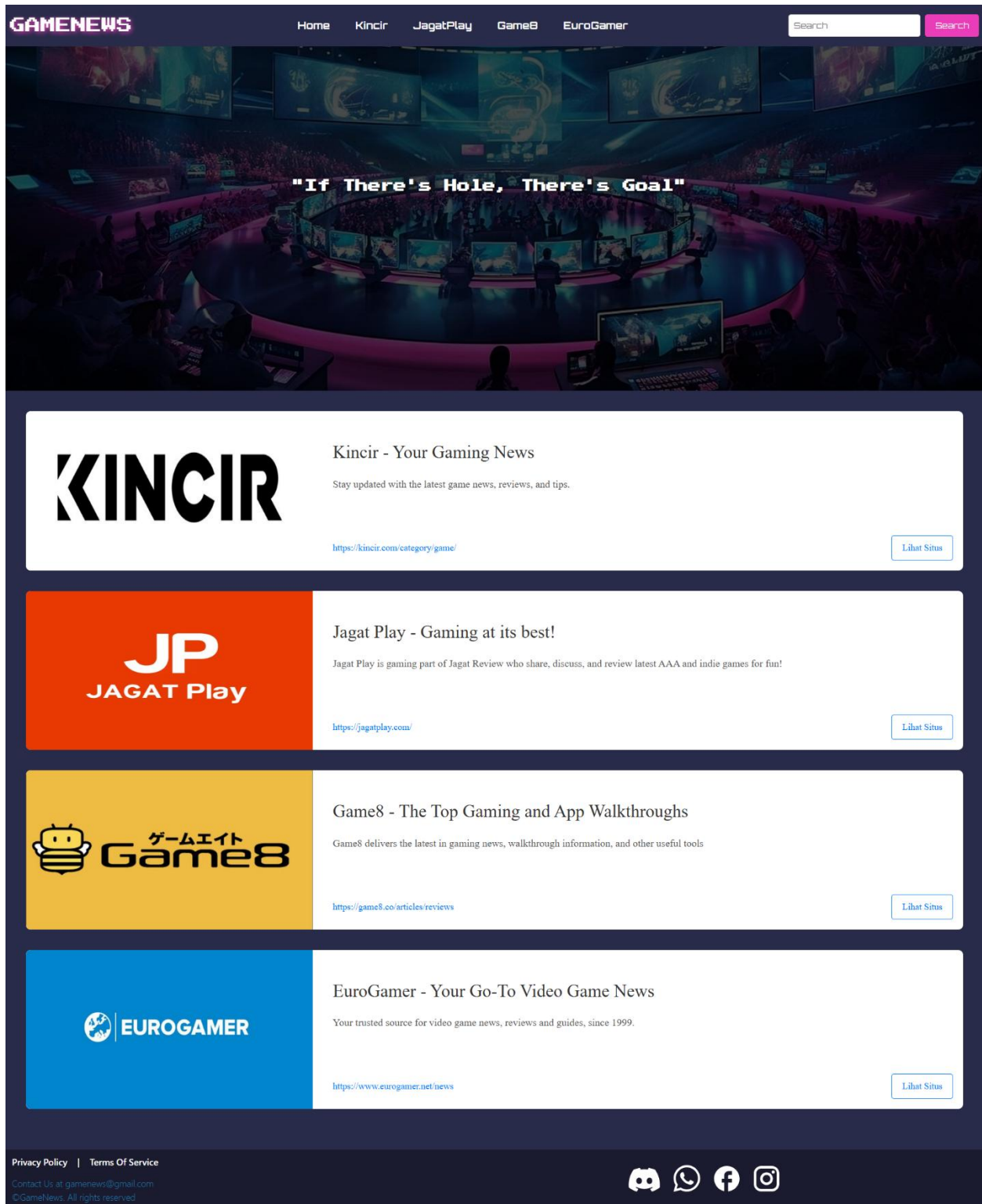
Jika tidak ada kondisi yang terpenuhi, proses berhenti dengan mengembalikan None.

- Proses parsing berdasarkan source, elemen data yang relevan diambil menggunakan class yang sesuai. Berikut detailnya :
 - Source "kincir"
 - 1) Data diambil dari elemen dengan atribut `class="article"`.
 - 2) Jika data ditemukan, elemen title, time, dan konten_div diambil menggunakan metode `.find()`.
 - 3) Jika konten_div ditemukan, elemen-elemen tag dalam konten_div diproses :
 - ❖ Tag img diambil menggunakan `.get("src")` atau `get('data-lazy-src', '')`.
 - ❖ Semua tag dimasukkan ke dalam list `konten_list`.
 - 4) Data artikel disusun dalam bentuk dictionary berisi title, time, thumb, dan konten.
 - Source "jagat"
 - 1) Data diambil dari elemen dengan atribut `class="jgpost__box"`.
 - 2) Jika data ditemukan :
 - ❖ Elemen title, time, thumb, figure, dan konten_div diproses.
 - ❖ thumb diekstraksi dari style menggunakan regex untuk mengambil URL gambar.
 - ❖ Data artikel disusun dalam bentuk dictionary berisi title, time, thumb, figure, dan konten.
 - Source "game8"
 - 1) Data diambil dari elemen dengan atribut `class="td-ss-main-content"`.
 - 2) Jika data ditemukan
 - ❖ Elemen title, time, `img_tag`, dan konten_div diproses.
 - ❖ thumb diperoleh dari atribut `src` di dalam `img_tag`.

- ❖ Data artikel disusun dalam bentuk dictionary berisi title, time, thumb, dan konten.
- Source “eurogamer”
 - 1) Data diambil dari elemen dengan atribut class=" page_content".
 - 2) Jika data ditemukan:
 - ❖ Elemen title, time, thumb, dan konten_div diproses.
 - ❖ Konten_div diperoleh dari elemen dengan atribut “p”.
 - ❖ Data artikel disusun dalam bentuk dictionary berisi title, time, thumb, dan konten.
- Setelah proses parsing selesai, hasil berupa dictionary article dikembalikan ke pemanggil.
- Proses berakhir pada node stop.

3. HASIL TAMPILAN

Halaman Utama




Halaman Kumpulan Berita (1 contoh)

GAMENEWS

HomeKincirJagatPlayGameBEuroGamer

Search


JAGAT NEWS



Sony Batalan Pengembangan 2 Game Live Service Yang Namanya Masih Misteri

January 17, 2025


[Baca Selengkapnya](#)



Nintendo Switch 2 Diperlihatkan Secara Resmi Melalui Video Reveal

January 17, 2025


[Baca Selengkapnya](#)



Banyaknya Info Leak Nintendo Switch 2 Akan Berdampak Pada Kesuksesan Rilis Console

January 16, 2025


[Baca Selengkapnya](#)



Nintendo Dan Sega Angkat Bicara Soal Legalitas Emulator

January 16, 2025


[Baca Selengkapnya](#)



Ubisoft Akhirnya Perbaiki Assassin's Creed Valhalla & Origins di Windows 11

January 16, 2025


[Baca Selengkapnya](#)



Doom Kini Bisa Dimainkan Dalam Format PDF

January 16, 2025


[Baca Selengkapnya](#)



Modder Baldur's Gate 3 Membuat Campaign Berbasis World of Warcraft

January 15, 2025


[Baca Selengkapnya](#)



Pemain Marvel Rivals Gunakan Invisible Woman Untuk Deteksi Bot

January 15, 2025


[Baca Selengkapnya](#)



Marvel Rivals Rencanakan Akan Rilis Satu Hero Baru Tiap Setengah Season

January 15, 2025


[Baca Selengkapnya](#)



Nintendo Umumkan Ketersediaan Alarms di Pasaran Pada Maret 2025

January 15, 2025


[Baca Selengkapnya](#)



Sony Patenkan Teknologi Untuk Kurangi Lag Input Ketika Bermain

January 15, 2025


[Baca Selengkapnya](#)



Pemain Final Fantasy XIV Terancam Privasinya Akibat Mod Berbahaya

January 14, 2025


[Baca Selengkapnya](#)



Pemain Escape from Tarkov Dihantui Ancaman Cheat Misterius

January 14, 2025


[Baca Selengkapnya](#)



Duet Night Abyss Membuka Pendaftaran Untuk Masa Close Beta Pertama

January 14, 2025

[Baca Selengkapnya](#)



Square Enix Buat Aturan Baru Untuk Lindungi Karyawan Dari Serangan Fans

January 13, 2025


[Baca Selengkapnya](#)

Halaman Detail Setiap Artikel (1 contoh)

GAMENEWS[Home](#)[Kincir](#)[JagatPlay](#)[Game8](#)[EuroGamer](#)

Duet Night Abyss Membuka Pendaftaran Untuk Masa Close Beta Pertama


News January 14, 2025



Duet Night Abyss kembali membuka kesempatan untuk gamer yang penasaran dengan game fast paced action bertema anime ini, melalui Close Beta pertamanya.

Ketika Duet Night Abyss pertama diumumkan melalui trailernya pada 2023, gameplay yang mirip Warframe tetapi bertema anime langsung menawan hati gamer di seantero dunia. Setelah dua tahun berselang, akhirnya tim developernya membuka kesempatan untuk gamer guna mencobanya melalui Close Beta pertamanya.

Bila Anda tertarik untuk mencoba memainkannya, Hero Games telah membuka kesempatan untuk mendaftarkan diri melalui [website resmi close betanya](#). Berlangsung hingga 10 Februari 2025 nanti, Anda bisa mencoba peruntungan melalui beberapa cara yang ditawarkan oleh tim developer game ini.



Pertama adalah dengan menggunakan email Anda untuk membuat akun di Hero Games, dan masuk ke dalam 'Test Recruitment' yang merupakan jalur standar untuk menjadi salah satu tester di close beta. Pada jalur ini, Anda akan diberikan daftar pertanyaan untuk melihat kecocokan Anda sebagai peserta test nantinya.

Beberapa pertanyaan mengenai kebiasaan bermain Anda, seperti platform apa yang akan digunakan untuk memainkan Duet Night Abyss, game yang dimainkan selama 6 bulan belakangan, hingga berapa lama Anda memainkannya menjadi contoh pertanyaan yang akan ditemui di sana.

Selain melalui jalur tersebut, Anda juga bisa mencoba peruntungan dengan cara lain, yaitu menjalankan beberapa tugas harian guna mendapatkan token. Token tersebut nantinya bisa digunakan untuk menarik 'gacha' dengan kesempatan untuk langsung terpilih sebagai peserta tes.

Tugas yang akan ditemui di sana lebih mencakup mengikuti dan share sosial media milik Duet Night Abyss, mengajak teman untuk ikut serta mendaftar, dan melakukan pre-register. Saat ini, Close Beta bisa diikuti oleh gamer mobile di Android dan iOS serta platform PC, tidak seperti Technical Test yang sempat dilaksanakan untuk PC saja.

Hadirnya Duet Night Abyss tentunya akan memanaskan persaingan di ranah game action RPG penuh grinding, yang saat ini masih didominasi oleh Warframe dan [The First Descendant](#). Semoga saja Hero Games akan memutuskan untuk melebarkan sayapnya ke gamer console, tidak terbatas pada mobile dan PC saja seperti saat ini.