Dokumentasi Proyek Scraping Dapodik

by Michael Vincent Sebastian Handojo

1. Metodologi dan Overview Proyek

Proyek ini bertujuan untuk mengekstrak data detail sekolah (Profil, PTK, Siswa, Sarpras) dari API dan *web portal* Dapodik Kemendikbud. Data mentah dari API (JSON) dikombinasikan dengan data terperinci dari halaman HTML (melalui *scraping* **BeautifulSoup**) untuk menghasilkan *dataset* yang kaya (52 kolom per sekolah).

1.1 Metodologi Utama: Arsitektur Modular

Sistem ini mengadopsi prinsip **Arsitektur Modular** (Separation of Concerns):

1. **Abstraksi Tugas Inti:** Semua fungsi teknis (HTTP Request, Parsing, I/O CSV, Error Handling) diisolasi ke dalam satu modul utilitas (**dapodik_utils.py**).
2. **Orkestrasi Logika:** Skrip utama (**kabBekasi.py, dkk.**) hanya bertindak sebagai *controller* yang mengimpor dan menjalankan fungsi-fungsi inti, fokus pada logika *looping* dan *filtering* wilayah.

1.2 Tools, Dependencies, dan Environment

Proyek ini menggunakan bahasa Python dan mengandalkan beberapa *library*:

Kategori	Tools/Library	Peran Teknis dalam Proyek
HTTP Request	requests	Menangani semua permintaan HTTP ke Dapodik API (JSON) dan URL profil sekolah (HTML).
HTML Parsing	beautifulsoup4 (bs4)	Digunakan di parse_html() untuk menavigasi, mencari, dan mengekstrak data dari struktur DOM halaman web Dapodik yang kompleks.
I/O Data	csv (Standard Lib)	Memastikan data diekspor ke format CSV yang terstruktur.
System Utility	os, time (Standard Lib)	os digunakan untuk manajemen *path* (os.path.join) dan yang terpenting, untuk **data**

		integrity (os.fsync). time digunakan untuk mekanisme *rate limiting* dan *backoff* (delay).
Configuration	urllib3	Digunakan untuk menonaktifkan InsecureRequestWarning, umumnya terkait dengan sertifikat SSL server API lama atau konfigurasi *request* spesifik.
Data Format	json	Digunakan untuk memproses respons dari Dapodik API yang berbentuk JSON.

2. Analisis Kode Layer Utilitas (dapodik_utils.py)

File ini adalah jantung fungsional sistem, menyediakan **4 fungsi kritis** yang menjamin operasional dan ketahanan:

2.1 Fungsi Request Berketahanan (request_api() & request_html())

Fungsi ini mengimplementasikan mekanisme **Infinite Retry** dengan **Exponential Backoff**:

Komponen	Implementasi Teknis	Tujuan
Infinite Loop	while True:	Memastikan skrip tidak pernah mati karena kegagalan jaringan (Timeout, Connection Error) atau error server 5xx (Internal Server Error).
Exception Handling	Menangkap requests.exceptions.RequestException dan Exception umum.	Melindungi dari kegagalan koneksi (*network-level*) dan error Python lainnya.

Backoff Strategy	time.sleep(backoff)	Setelah kegagalan, skrip berhenti sejenak. Nilai backoff biasanya dikalikan pada *loop* berikutnya (meski diimplementasi di skrip utama, ini adalah prinsipnya). Jeda ini mencegah IP di-*ban* dan memberi waktu server untuk pulih.
URL Parameterization	request_api() menerima level_wilayah, kode_wilayah, semester_id sebagai parameter, menjadikannya fleksibel untuk mengakses setiap *level* hierarki wilayah Dapodik.	

2.2 Fungsi Parsing HTML (parse_html(url))

Fungsi ini bertanggung jawab untuk mengubah data *unstructured* (HTML) menjadi data *structured* (Dictionary Python):

1. **Ekstraksi Data Terstruktur:** Menggunakan BeautifulSoup untuk menargetkan elemen berdasarkan atribut CSS/ID.
    - **Data Profil:** Konten di dalam elemen .panel-body dan .panel-default diurai secara iteratif.
    - **Data Kunci-Nilai:** Data diekstrak dengan menargetkan tag (kunci) dan mengambil *sibling* teksnya (nilai), sebuah pola umum untuk tabel statistik HTML.

2. **Output:** Mengembalikan *dictionary* berlapis (nested dict) yang berisi semua data rinci sekolah (e.g., Identitas, Data Rinci, Data PTK/Siswa), siap untuk di-*flat* menjadi baris CSV.

2.3 Fungsi Integritas Data I/O (append_to_csv())

Fungsi ini adalah yang paling penting untuk **transaksionalitas** *scraping* Anda:

Fitur	Implementasi Teknis	Dampak pada Ketahanan
Write Mode	Menggunakan mode='a' (append) dan newline=''	Memastikan baris baru ditambahkan ke CSV yang sudah ada, dan mencegah baris kosong yang berlebihan pada sistem operasi tertentu.
Data Integrity	**csvfile.flush()** diikuti oleh **os.fsync(csvfile.fileno())**	Ini adalah *best practice* kritikal. flush() membersihkan *buffer* Python, dan os.fsync() secara eksplisit memaksa sistem operasi untuk menulis data dari *buffer* OS ke **media penyimpanan fisik (disk)**. Ini menjamin data sekolah yang baru diambil tidak akan hilang jika terjadi *crash* sistem (*power outage*, *kernel panic*, dll.).

2.4 Fungsi Resume Otomatis (load_processed_ids())

Fungsi ini mendukung fitur **Resume Otomatis**:

1. Membaca file CSV target sebelum *looping* dimulai.
2. Mengekstrak semua sekolah_id_enkrip yang sudah ada dan menyimpannya dalam struktur **Set Python**.
3. Penggunaan **Set** memberikan kinerja pencarian O(1) (*constant time*), yang sangat efisien untuk memeriksa apakah ID sekolah sudah ada, bahkan pada *dataset* besar.

3. Analisis Kode Layer Kontroler Wilayah

Skrip (kabBekasi.py, kotaBekasi.py, kotaDepok.py) menerapkan Logika Iterasi 4-Tier:

3.1 Struktur Iterasi (4-Tier)

Skrip melakukan *query* Dapodik secara hierarkis, dari Level 0 hingga Level 4:

1. **Level 0:** Ambil semua Provinsi (Indonesia).
2. **Level 1:** Ambil semua Kota/Kabupaten di Provinsi target (Jawa Barat).
3. **Level 2: Filtering Wilayah** (lihat poin B). Ambil semua Kecamatan di Kota/Kabupaten target.
4. **Level 3:** Ambil semua Daftar Sekolah di Kecamatan target.

3.2 Logika Filter Wilayah yang Spesifik

Filter pada Level 2 sangat spesifik untuk memastikan pemisahan data yang benar:

Skrip	**Kata Kunci Filter**	**Logika Pemisahan**
kotaDepok.py	KOTA DEPOK	String *matching* eksplisit pada nama Kota/Kabupaten.
kotaBekasi.py	KOTA BEKASI	String *matching* eksplisit pada nama Kota/Kabupaten.
kabBekasi.py	KAB & BEKASI **AND NOT** KOTA	Menggunakan logika **Negative Lookahead** (tidak mengandung KOTA) untuk secara tegas mengecualikan Kota Bekasi dan hanya mengambil Kabupaten Bekasi, mengatasi inkonsistensi penulisan nama wilayah dari API.

3.3 Implementasi Rate Limiting

Untuk mencegah *IP ban* atau *timeout* akibat *spamming* server, skrip menerapkan jeda waktu secara strategis:

1. **Makro Delay (Level Kecamatan):** time.sleep(8) diterapkan **setelah** seluruh data di satu Kecamatan selesai diproses.

2. **Micro Delay (Level Sekolah):** time.sleep(3) diterapkan **setelah** data detail (parse_html) satu sekolah berhasil diambil.

4. Kendala dan Solusi (Robustness Summary)

Kendala Teknis	Mekanisme Solusi yang Diimplementasikan	Lokasi Kode
Server Tidak Stabil / Timeout	**Infinite Retry & Backoff:** Menggunakan while True untuk terus mencoba permintaan HTTP yang gagal dengan jeda waktu yang bertambah (backoff).	request_api(), request_html()
Potensi IP Ban	**Strategis Time Delay:** Jeda 8 detik antar Kecamatan dan 3 detik antar Sekolah untuk menghindari *rate limiting* yang agresif.	Skrip Kontroler (e.g., kabBekasi.py)
Data Loss / Duplikasi Saat Crash	**I/O Sync (os.fsync)**: Memaksa penulisan ke *disk fisik* setelah setiap baris CSV ditulis. Data sekolah terakhir tidak akan hilang.	append_to_csv() (di dapodik_utils.py)
Memproses Ulang Sekolah (Duplikasi)	**Set-Based Resume:** Memuat semua ID yang sudah ada ke Set Python (O(1) *lookup*) sebelum *looping*, memastikan sekolah yang sudah sukses di-skip.	load_processed_ids() (di dapodik_utils.py)
Inkonsistensi Nama Wilayah API	**String Filtering Fleksibel:** Menggunakan in operator dengan kombinasi AND NOT (logika *negative lookahead*) untuk membedakan KABUPATEN BEKASI dari KOTA BEKASI.	Skrip Kontroler (e.g., kabBekasi.py)