# SelemX

## 1. Opis projektu

**SelemX** to narzędzie wiersza poleceń do:

* pobierania tweetów z X/Twitter (widok mobilny) przy użyciu Selenium i „Chrome for Testing”,
* łączenia danych z lokalnej bazy **SQLite** (tryb **DB-first**) z brakującymi wpisami dociąganymi z sieci (**top-up**),
* deduplikacji (połączenie **filtru Bloom** i klucza głównego w bazie),
* analizy sentymentu i metryki „polarity” dla języka polskiego,
* tworzenia wizualizacji (rozkład sentymentów, trend polaryzacji, **chmura słów**),
* pracy z **checkpointami** (wznawianie zarówno scrapowania RAW, jak i analizy),
* eksportu wyników do **CSV** i **Parquet**,
* łagodnego radzenia sobie z **rate-limitami** (wykrywanie blokad, cooldown z paskiem postępu).

Projekt jest przygotowany do pracy na Windows, Linux i macOS. Nie wymaga globalnej instalacji Chrome — potrafi automatycznie pobrać i użyć „Chrome for Testing” z dopasowanym ChromeDriverem.

## 2. Wymagania systemowe

* Python 3.10+ (zalecany 3.11–3.13).
* System: Windows 10/11, Ubuntu/Debian, macOS.
* Uprawnienia do tworzenia plików w katalogu projektu (foldery browser/, db/, results/).
* Dostęp do Internetu (pierwsze uruchomienie pobiera Chrome CFT/Driver).
* W przypadku Windows: uruchom terminal/PowerShell z uprawnieniami do zapisu w katalogu projektu.

## 3. Struktura katalogów

Po pierwszym uruchomieniu projekt porządkuje dane w następujących folderach:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

## 4. Instalacja

**4.1. Szybka instalacja (Linux/macOS/Git Bash na Windows)**

1. Nadaj wykonywalność skryptowi:

chmod +x setup.sh

1. Uruchom:

./setup.sh

Skrypt:

* tworzy nowe środowisko venv/,
* aktualizuje pip,
* instaluje pakiety z requirements.txt,
* aktywuje środowisko.

**4.2. Ręczna instalacja (Windows PowerShell)**

1. Utwórz i aktywuj wirtualne środowisko:

python -m venv venv

venv\Scripts\activate

1. Zainstaluj pakiety:

pip install -r requirements.txt

## 5. Konfiguracja (config.py)

Domyślna konfiguracja (ważne elementy):

* **Foldery bazowe**: browser/, db/, results/ — tworzone automatycznie.
* **Chrome/Driver**: jeśli brak, zostaną pobrane do browser/chrome\_for\_testing/....
* **Baza danych**: db/tweets.sqlite.
* **Profil Chrome**: USER\_DATA\_DIR = str((Path.cwd() / "browser" / "profile").resolve())

To **Twoje obecne ustawienie**. Profil jest lokalny dla repozytorium, co ułatwia logowanie do X przy kolejnych uruchomieniach, także w trybie headless.

* **Cooldown (rate-limit)**: RATE\_LIMIT\_COOLDOWN = 450 sekund (można nadpisać flagą CLI).
* **Checkpointy** (progi czasu i liczby twittów) – patrz rozdział 10.
* **Zapis wyników**: domyślnie CSV i Parquet są włączone (można wyłączyć flagami --no-csv/--no-parquet).

## 6. Uruchomienie (manual użytkownika)

1. Aktywuj środowisko (jeśli nie użyto setup.sh):
   * Windows (PowerShell): venv\Scripts\activate
   * Linux/macOS: source venv/bin/activate
2. Uruchom:

python main.py

Program poprosi o:

* + słowo kluczowe (keyword),
  + daty since i until w formacie YYYY-MM-DD,
  + limit liczby tweetów,
  + nazwę kolekcji (domyślnie przyjmuje keyword).

1. Jeśli **nie** używasz --db-only, otworzy się Chrome:
   * zaloguj się do X (jeśli wymagane),
   * kliknij w wstrzyknięty na dole ekranu przycisk **„KONTYNUUJ (kliknij po zalogowaniu)”**,
   * scraper ruszy automatycznie również wtedy, gdy wykryje treści tweetów bez konieczności klikania.
2. Obserwuj paski postępu:
   * pobieranie i rozpakowanie Chrome/Driver,
   * scrapowanie (z podziałem na okna czasowe),
   * ewentualny cooldown (rate-limit),
   * analiza sentymentu i tworzenie wykresów.
3. Wyniki znajdziesz w results/<collection>/<since>\_to\_<until>/... oraz w bazie (db/tweets.sqlite).

## 7. Tryby pracy i „kolekcje”

* **Kolekcja** to etykieta (np. „Trump”), która grupuje tweety tematycznie. W bazie możesz utrzymywać wiele kolekcji.
* Analiza **zawsze dotyczy okna dat** [since..until], niezależnie od tego, ile tweetów dana kolekcja ma łącznie.
* Logika **DB-first + top-up**:
  + najpierw pobieramy tweety z **DB** dla danej kolekcji w oknie [since..until],
  + jeśli brakuje rekordów do osiągnięcia max\_tweets — dociągamy brakujące z X/Twitter,
  + nowe tweety trafiają do DB i są przypięte do kolekcji; przy kolejnym uruchomieniu będą traktowane jako dane lokalne.

## 8. Użycie (CLI)

**8.1. Presety**

Możesz użyć gotowych profili startowych, a następnie nadpisać dowolny parametr flagami:

* --preset daily\_refresh  
  Ostatnie 24h, --use-bloom, --refresh, --cooldown 600, --max-tweets 1000.
* --preset rolling7  
  Ostatnie 7 dni, --use-bloom, --cooldown 600, --max-tweets 500.
* --preset deep\_crawl  
  „Głęboki” zbiór: --max-tweets 5000, --use-bloom, --cooldown 900, --resume,  
  progi checkpointów: --analysis-progress-sec 20, --progress-every 200, --progress-sec 45.
* --preset db\_only  
  Analiza wyłącznie z bazy (domyślnie 7 dni), --max-tweets 1000.
* --preset server\_headless  
  7 dni, --use-bloom, --headless, --cooldown 600, --max-tweets 800.  
  Wymaga gotowego profilu logowania (--user-data-dir ./browser/profile).
* --preset parquet\_only  
  Tylko Parquet (--no-csv), --max-tweets 500.

**8.2. Najważniejsze flagi**

* Ścieżki:
  + --browser-dir PATH — domyślnie ./browser.
  + --db-dir PATH — domyślnie ./db.
  + --results-dir PATH — domyślnie ./results.
  + --user-data-dir PATH — domyślnie **./browser/profile** (Twoje aktualne ustawienie).
  + --headless — Chrome w tle (zwykle wymaga wcześniejszego zalogowania w profilu).
* Deduplikacja / limity / checkpointy:
  + --use-bloom — włącza HybridDeduper (Bloom + DB).
  + --cooldown SECONDS — czas odczekania przy rate-limit (domyślnie 300).
  + --progress-every N — RAW checkpoint co N nowych tweetów (domyślnie 100).
  + --progress-sec N — RAW checkpoint co N sekund (domyślnie 60).
  + --analysis-progress-sec N — ANALYSIS checkpoint co N sekund (domyślnie 30).
  + --checkpoint-keep K — ile checkpointów z timestampem trzymać (domyślnie 5).
* Wznawianie / odświeżanie:
  + --resume — wznów **RAW + ANALYSIS** z najnowszych checkpointów.
  + --resume-raw lub --resume-analysis — wybiórczo.
  + --refresh — wymuś dociągnięcie z Twittera nawet, gdy DB ma komplet w danym oknie.
  + --db-only — użyj tylko danych z DB (bez scrapingu).
* Zapis wyników:
  + --no-parquet — nie zapisuj Parquet.
  + --no-csv — nie zapisuj CSV.
* Parametry merytoryczne:
  + --keyword "fraza"
  + --since YYYY-MM-DD
  + --until YYYY-MM-DD
  + --max-tweets N
  + --collection "NazwaKolekcji" (domyślnie = keyword)

**8.3. Przykłady uruchomień**

* Ostatni tydzień, Bloom, 500 tweetów:

python main.py --preset rolling7 --keyword "LGBT" --use-bloom --max-tweets 500

* Głęboki zbiór z wznawianiem i dłuższym cooldownem:  
    
  python main.py --preset deep\_crawl --keyword "Tusk" --since 2025-07-01 --until 2025-08-30
* Analiza tylko z bazy:  
    
  python main.py --preset db\_only --keyword "inflacja" --since 2025-08-01 --until 2025-08-30 --db-only
* Serwer/headless z gotowym profilem:  
    
  python main.py --preset server\_headless --keyword "Tesla" --user-data-dir "./browser/profile" –headless\
* Tylko Parquet (bez CSV):  
    
  python main.py --preset parquet\_only --keyword "PKB"

## 9. Przepływ działania (skrót)

1. Parsowanie presetów i flag.
2. Ustalenie ścieżek i konfiguracji (w tym USER\_DATA\_DIR = ./browser/profile).
3. Tryb **DB-only** albo uruchomienie Selenium + Chrome CFT:
   * otwarcie mobile.twitter.com/search,
   * przycisk **„KONTYNUUJ”** (po zalogowaniu) lub start po wykryciu treści.
4. Scrapowanie:
   * podział zakresu czasu na miesiące, następnie na krótkie „slice’y” (dni),
   * deduplikacja (Bloom + DB + pamięć bieżącej sesji),
   * checkpointy RAW i bieżący zapis do DB.
5. Analiza sentymentu i polaryzacji:
   * checkpointy ANALYSIS,
   * wykresy PNG, eksport CSV/Parquet.

## 10. Checkpointy

W SelemX mamy **dwa niezależne etapy**, dla których trzymamy **osobne checkpointy** i którymi można **wznawiać** selektywnie:

1. **RAW** (scraping) — pobieranie **surowych tweetów** z X/Twitter i zapisywanie ich do bazy SQLite (oraz przypinanie do kolekcji).
2. **ANALYSIS** — analiza sentymentu/polaryzacji na **podzbiorze** bazy w oknie dat [since..until], wraz z generowaniem wykresów i eksportów.

Każdy z etapów ma **własne pliki checkpointów** i **własną logikę resume**. Dzięki temu można:

* wznowić **tylko zbieranie** (RAW), **tylko analizę** (ANALYSIS) lub **oba** naraz,
* kontrolować zachowanie przez flagi --resume, --resume-raw, --resume-analysis.

**Miejsca zapisów:**

* **RAW (scraping)**  
  Folder: db/checkpoints/raw/  
  Pliki:
  + raw\_progress\_<collection>\_<since>\_<until>\_latest.csv
  + kopie z timestampem (rotacja: CHECKPOINT\_KEEP, domyślnie 5)  
    Wyzwalanie: co --progress-every tweetów **lub** co --progress-sec sekund.  
    Zawartość: id, raw\_text, date, url.
* **ANALYSIS**  
  Folder: db/checkpoints/analysis/  
  Pliki:
  + analysis\_progress\_latest.csv
  + kopie z timestampem (rotacja: CHECKPOINT\_KEEP)  
    Wyzwalanie: co --analysis-progress-sec sekund.
* **Wznawianie**  
  Flagi: --resume, --resume-raw, --resume-analysis.  
  Wznowienie pobiera dane z plików \*\_latest.csv i kontynuuje od miejsca przerwania.

## 11. Deduplikacja (HybridDeduper)

* **Filtr Bloom** (w pamięci, opcjonalnie serializowany do db/tweet\_ids\_bloom.pickle) — szybkie odsiewanie znanych identyfikatorów.
* **DB (klucz PRIMARY KEY tweets.id)** — gwarancja braku duplikatów w bazie.
* **Fallback ID** — gdy brak status/<id> w URL, używany jest stabilny hash: txt\_<sha1(clean\_text)> (clean\_text = tekst po oczyszczeniu z URL-i, wzmiankowań itp.).
* Deduplikacja działa między uruchomieniami — dzięki serializacji filtru Bloom i kluczowi w DB.

## 12. Baza danych (SQLite)

Plik bazy: db/tweets.sqlite.

**12.1. Tabele**

**Tabela: tweets**

* id (TEXT, PRIMARY KEY): ID z /status/<id> lub fallback txt\_<sha1(clean\_text)>.
* text (TEXT, NOT NULL): surowy tekst tweeta (zachowany oryginał).
* date (TEXT, ISO 8601, NULL dopuszczalne): np. 2025-08-30T12:34:56+00:00.
* url (TEXT, NULL dopuszczalne): pełny link do tweeta.

**Tabela: collections**

* id (INTEGER, PRIMARY KEY AUTOINCREMENT).
* name (TEXT, UNIQUE, NOT NULL): nazwa kolekcji (np. „Zandberg”).

**Tabela: collection\_tweets**

* collection\_id (INTEGER, NOT NULL) → FK do collections(id).
* tweet\_id (TEXT, NOT NULL) → FK do tweets(id).
* PRIMARY KEY: (collection\_id, tweet\_id) (tweet przypięty do kolekcji co najwyżej raz).

**12.2. Indeksy (tworzone w store.py)**

* CREATE INDEX IF NOT EXISTS idx\_tweets\_date ON tweets(date);
* CREATE INDEX IF NOT EXISTS idx\_ct\_coll ON collection\_tweets(collection\_id);
* CREATE INDEX IF NOT EXISTS idx\_ct\_tweet ON collection\_tweets(tweet\_id);

**12.3. Semantyka i praktyka użycia**

* **Kolekcja jest długowieczna**: z czasem „rośnie” o kolejne tweety dociągane w różnych uruchomieniach.
* **Analiza jest okienkowa**: zawsze wybierasz since..until — raport i eksport obejmą dokładnie ten przedział (podzbiór kolekcji).
* **DB-first**: to, co już mamy w DB (dla tej kolekcji i okna), jest użyte w pierwszej kolejności; później następuje top-up z sieci.

**12.4. Przykładowe zapytania**

* Tweety kolekcji w oknie dat:  
    
  SELECT t.\*  
  FROM tweets t  
  JOIN collection\_tweets ct ON ct.tweet\_id = t.id  
  JOIN collections c ON c.id = ct.collection\_id  
  WHERE c.name = 'NFZ'  
   AND (t.date IS NULL OR date(t.date) BETWEEN date('2025-08-23') AND date('2025-08-30'));
* Liczba tweetów przypiętych do danej kolekcji:  
    
  SELECT COUNT(\*)  
  FROM collection\_tweets ct  
  JOIN collections c ON c.id = ct.collection\_id  
  WHERE c.name = 'UKRAINA';

## 13. Wyniki i wizualizacje

W folderze results/<collection>/<since>\_to\_<until>/ znajdziesz:

* Pliki danych:
  + \*.csv (jeśli nie wyłączono --no-csv),
  + \*.parquet (jeśli nie wyłączono --no-parquet).
* Wykresy:
  + sentiment\_distribution.png — histogram/rozkład klas sentymentu,
  + polarity\_trend.png — wykres trendu średniej polaryzacji w czasie,
  + wordcloud.png — chmura słów z tekstów z okna.

Wyniki analizy **nie** są zapisywane w SQLite — dzięki temu baza pozostaje lekka; raportowe wyniki są wersjonowane po oknie dat w results/.

## 14. Obsługa rate-limitów i blokad

* Wykrywane są typowe komunikaty X/Twitter, m.in.:  
  „Something went wrong. Try reloading.”, „Retry”, „Reload”, „Too many requests / Rate limit”, polskie odpowiedniki.
* Strategia:
  1. szybkie, krótkie próby (kliknięcie „Retry”, odświeżenie strony),
  2. **cooldown** na czas --cooldown (domyślnie 300 s) z **paskiem postępu**,
  3. ponowna próba.
* Osobny pasek postępu działa także przy oczekiwaniu na kliknięcie przycisku **„KONTYNUUJ”** po zalogowaniu.

## 15. Najlepsze praktyki

* Jedna **kolekcja = jeden temat** (np. Zandberg, Inflacja). Dzięki temu top-up z czasem buduje spójny korpus.
* Przy długich runach używaj --resume oraz sensownych progów --progress-every i --progress-sec.
* Jeżeli często trafiasz na rate-limit: zwiększ --cooldown (np. 600 lub 900), zmniejsz --max-tweets, rozważ inny IP.
* Tryb headless zwykle wymaga **gotowego zalogowanego profilu** (--user-data-dir ./browser/profile).

## 16. Rozwiązywanie problemów

1. **Windows: „Sandbox cannot access executable… (0x5)”**
   * Uruchom PowerShell jako Administrator lub przenieś repozytorium do folderu użytkownika (poza „Program Files”).
   * Sprawdź, czy Windows Defender/AV nie blokuje browser/.../chrome.exe.
2. **QUOTA\_EXCEEDED / DEPRECATED\_ENDPOINT / PHONE\_REGISTRATION\_ERROR**
   * Sieć, quota usług Google/X; SelemX sam uruchamia cooldown.
   * Jeśli problem nawraca: zwiększ --cooldown, zmniejsz --max-tweets, użyj innego IP/VPN.
3. **Read timed out (Selenium/DevTools)**
   * Niestabilne łącze lub zawieszony driver. Wydłuż --cooldown, przerwij i uruchom z --resume.
4. **„Brak tweetów do analizy” po długim scrapie**
   * Upewnij się, że:  
     a) --collection zgadza się z kolekcją, do której zapisywałeś,  
     b) okno [since..until] nie jest puste (np. literówka lub odwrotna kolejność dat),  
     c) przy --db-only w bazie faktycznie istnieją tweety w tym oknie.
   * Dla pewności uruchom bez --db-only albo z --refresh.
5. **Logowanie / widok X w trybie headless**
   * Najpierw zaloguj się w **nie-headless**, zamknij, potem używaj --headless z tym samym --user-data-dir.

## 17. Bezpieczeństwo i prywatność

* Profil przeglądarki jest lokalny (./browser/profile), izolowany od Twojego profilu systemowego.
* Baza tweets.sqlite przechowuje jedynie **surowe treści tweetów** + minimalne metadane (data, URL, mapowania do kolekcji).
* Wyniki analizy (sentyment, polarity) trafiają do plików w results/ (łatwo je wersjonować, przenosić i czyścić).

## 18. Szybka ściąga (cheatsheet)

* Instalacja (Linux/macOS/Git Bash):  
  ./setup.sh
* Uruchomienie interaktywne:  
  python main.py
* Ostatni tydzień, Bloom, 500 szt.:  
  python main.py --preset rolling7 --keyword "Zandberg" --use-bloom --max-tweets 500
* Głęboko + wznawianie:  
  python main.py --preset deep\_crawl --keyword "Tusk" --since 2025-07-01 --until 2025-08-30 --resume
* Tylko DB:  
  python main.py --preset db\_only --keyword "inflacja" --db-only
* Headless serwer:  
  python main.py --preset server\_headless --keyword "WIG20" --user-data-dir "./browser/profile" --headless

## 19. Załącznik — skrót opcji konfiguracyjnych (config.py)

* USER\_DATA\_DIR — **domyślnie ./browser/profile**.
* RATE\_LIMIT\_COOLDOWN — czas cooldownu w sekundach (domyślnie 300).
* RAW\_PROGRESS\_EVERY\_N\_TWEETS / RAW\_PROGRESS\_EVERY\_SEC — progi checkpointów RAW.
* AN\_PROGRESS\_MIN\_INTERVAL\_SEC — minimalny odstęp między checkpointami analizy.
* CHECKPOINT\_KEEP — retencja archiwalnych checkpointów (z timestampem).
* SAVE\_CSV / SAVE\_PARQUET — włącz/wyłącz zapis wyjściowy.
* USE\_BLOOM / BLOOM\_SERIAL — kontrola filtru Bloom i ścieżka serializacji.
* CFT\_JSON, CFT\_OUTDIR, CFT\_CHANNEL — źródło i miejsce instalacji „Chrome for Testing”.

## 20. Informacje końcowe

* SelemX jest stworzony z myślą o odporności na zmiany UI X (widok mobilny) i ograniczenia platformy (rate-limit).
* W razie zmian po stronie X można regulować parametry: --cooldown, --max-tweets, --use-bloom, --resume, a także korzystać z profilu ./browser/profile.