Zespół Szkół NR.10
Im. Stanisława Staszica

# Sprawozdanie z projektu ProxyScrape

Autor: Adam Płochocki
Warszawa, 13.05.2025 r.

# Spis treści

## Opis projektu

Program Proxy Rotator to aplikacja stworzona do zarządzania adresami ip serwerów proxy oraz filtrowania ich według różnych kryteriów. Umożliwia użytkownikom:

- Wyświetlanie dostępnych serwerów proxy.
- Filtrowanie ich według kryteriów takich jak adres czy kraj.
- Dodawanie nowych serwerów proxy oraz usuwanie niepotrzebnych.
- Sprawdzanie, czy dany serwer proxy działa prawidłowo.
- Obsługę logowania i rejestracji użytkowników.
- Głównym celem tej aplikacji jest usprawnienie procesu zarządzania serwerami proxy, co jest szczególnie przydatne w przypadku zastosowań, które wymagają anonimowości lub omijania ograniczeń geograficznych.

## Jak działa ten projekt?

Projekt wykorzystuje architekturę składającą się z kilku odrębnych komponentów:

- Frontend (interfejs użytkownika): Aplikacja jest zbudowana w JavaFX, gdzie interfejs użytkownika pozwala na wykonywanie różnorodnych operacji związanych z zarządzaniem proxy.
- Backend: Logika aplikacji została zaimplementowana jako zestaw klas odpowiedzialnych za obsługę bazy danych, operacje na danych użytkownika, filtrację serwerów proxy oraz ich kontrolę.
- Baza danych: Aplikacja korzysta z zewnętrznej bazy danych, do której połączenie zarządzane jest przez klasę DatabaseManager.

Kroki działania aplikacji:

1. Użytkownik loguje się do aplikacji za pomocą swojego adresu e-mail i hasła. Jeśli nie posiada konta, może je zarejestrować.
2. Po zalogowaniu użytkownik przechodzi do głównego widoku, gdzie może przeglądać dostępne serwery proxy, filtrować je, dodawać nowe lub usuwać istniejące.

3. Operacje na danych są realizowane przez połączenia z bazą danych, zapewniając synchronizację i trwałość danych.
4. Użytkownik może sprawdzić, czy serwery proxy są aktywne, dzięki automatycznej weryfikacji działania adresów.

## Publiczne metody projektu i ich opis

1. Klasa: **DatabaseManager** - Klasa zarządza połączeniem z bazą danych oraz umożliwia wykonywanie operacji na danych.

- **public static void connect()**

  **Opis**: Nawiązuje połączenie z bazą danych.
  **Parametry**: Brak.
  **Zwracana wartość**: Brak.
  **Zastosowanie**: Wywoływana podczas uruchamiania aplikacji.

- **public static void executeUpdate(String query, Object... params)**

  **Opis**: Wykonuje zapytania SQL typu INSERT, UPDATE i DELETE.
  **Parametry**:
  - query - ciąg zapytania SQL.
  - params - parametry zastępowane w zapytaniu.

  **Zwracana wartość**: Brak.

- **public static ResultSet executeQuery(String query, Object... params)**

  **Opis**: Wykonuje zapytania SQL typu SELECT i zwraca wyniki w postaci ResultSet.
  **Parametry**:
  - query - zapytanie SQL.
  - params - parametry dla zapytania.

  **Zwracana wartość**: Obiekt ResultSet zawierający wyniki.

- **public static void executeBatch(String query, List<Object[]> batches)**

  **Opis**: Umożliwia wykonanie wielu zapytań SQL w jednym kroku.
  **Parametry**:
  - query - szablon zapytania SQL.
  - batches - lista zestawów parametrów dla zapytań.

  **Zwracana wartość**: Brak.

2. Klasa: **UserPrefs** - Klasa umożliwia zarządzanie ustawieniami użytkownika przy użyciu mechanizmu przechowywania preferencji.

- **public static void setUserId(int userId)**

  **Opis**: Zapisuje identyfikator użytkownika w preferencjach.
  **Parametry**:
  - userId - identyfikator użytkownika.

  **Zwracana wartość**: Brak.

- **public static int getUserId()**

  **Opis**: Pobiera zapisany identyfikator użytkownika.
  **Parametry**: Brak.
  **Zwracana wartość**: Identyfikator użytkownika lub -1, jeśli brak danych.

- **public static void removeUserId()**

  **Opis**: Usuwa zapisany identyfikator użytkownika z preferencji.
  **Parametry**: Brak.
  **Zwracana wartość**: Brak.

- **public static void setLastProxyFolder(String path)**

  **Opis**: Zapisuje ścieżkę do ostatnio używanego folderu z serwerami proxy.
  **Parametry**:
  - path - ścieżka.

  **Zwracana wartość**: Brak.

- **public static String getLastProxyFolder()**

  **Opis**: Pobiera zapisany folder z serwerami proxy.
  **Parametry**: Brak.
  **Zwracana wartość**: Ścieżka do folderu (String).

3. Klasa: **UserService** - Odpowiada za obsługę logiki użytkownika, w tym logowanie i rejestrację.

- **public static boolean login(String email, String password)**

  **Opis**: Loguje użytkownika, sprawdzając dane użytkownika w bazie danych.
  **Parametry**:
  - email - login użytkownika.
  - password - hasło użytkownika.

**Zwracana wartość**: true w przypadku pomyślnego logowania, false w przeciwnym razie.

- **public static boolean register(String email, String password, String name, String last_name)**

  **Opis**: Rejestruje nowego użytkownika w systemie.
  **Parametry**:
  - email - e-mail użytkownika.
  - password - hasło użytkownika.
  - name - imię.
  - last_name - nazwisko.

  **Zwracana wartość**: true, jeśli rejestracja powiodła się, false w przeciwnym razie.

4. Klasa: **ProxyManager** - Zarządza serwerami proxy oraz ich obsługą (np. dodawanie, usuwanie).

- **public static void addProxies(Set<String> proxies)**

  **Opis**: Dodaje nowe serwery proxy.
  **Parametry**:
  - proxies - zbiór adresów serwerów.

  **Zwracana wartość**: Brak.

- **public static List<ProxyElement> getProxies()**

  **Opis**: Pobiera listę wszystkich serwerów proxy.
  **Parametry**: Brak.
  **Zwracana wartość**: Lista obiektów ProxyElement.

- **public static void deleteProxies(List<ProxyElement> proxies)**

  **Opis**: Usuwa określone serwery proxy.
  **Parametry**:
  - proxies - lista serwerów do usunięcia.

  **Zwracana wartość**: Brak.

- **public static String getProxySet()**

  **Opis**: Pobiera aktualnie ustawiony serwer proxy.
  **Parametry**: Brak.
  **Zwracana wartość**: String z adresem proxy lub null.

5. Klasa: **ProxyFilter** - Zawiera mechanizmy filtrowania listy serwerów proxy.

- **public static List<ProxyElement> filterByAddress(List<ProxyElement> proxies, String address)**

  **Opis**: Filtruje serwery proxy według adresu.

**Parametry**:
- proxies - lista proxy.
- address - adres do filtrowania.

**Zwracana wartość**: Przefiltrowana lista ProxyElement.

- **public static List<ProxyElement> filterByCountryName(List<ProxyElement> proxies, String countryName)**

  **Opis**: Filtruje serwery proxy według nazwy kraju.
  **Parametry**:
  - proxies - lista proxy.
  - countryName - nazwa kraju do filtrowania.

  **Zwracana wartość**: Przefiltrowana lista ProxyElement.

- **public static List<ProxyElement> filterByBoth(List<ProxyElement> proxies, String address, String countryName)**

  **Opis**: Filtruje serwery proxy według adresu i kraju.
  **Parametry**:
  - proxies - lista proxy.
  - address - adres.
  - countryName - nazwa kraju.
  - Zwracana wartość: Przefiltrowana lista ProxyElement.

## Możliwości dalszego rozwoju

1. **Zaawansowane filtrowanie** – dodanie kryteriów, takich jak prędkość czy anonimowość proxy.
2. **Integracja z API** – automatyczne pobieranie i aktualizacja danych proxy.
3. **Statystyki i analiza** – prezentacja statystyk działania i jakości proxy.
4. **Optymalizacja i bezpieczeństwo** – np. dwustopniowe uwierzytelnianie i lepsza obsługa dużych list proxy.
5. **Personalizacja UI** – ulepszony interfejs i wersja mobilna.
6. **Automatyczna weryfikacja** – testowanie prędkości i usuwanie nieaktywnych proxy.
7. **Powiadomienia** – system informowania o błędach lub nowych proxy.
8. **Wsparcie dla współpracy** – konta z różnymi poziomami dostępu i współdzielenie list proxy.

Rozszerzenia te zwiększą funkcjonalność aplikacji, poprawiając jej użyteczność i wydajność.

## Podsumowanie

Projekt Proxy Rotator zrealizowano jako aplikację ułatwiającą zarządzanie dużymi ilościami serwerów proxy. Poprzez funkcje takie jak filtrowanie, dodawanie oraz weryfikacja proxy, aplikacja oszczędza czas i poprawia organizację. Przygotowana architektura z łatwością może zostać rozwinięta w przyszłości, np. o dodatkowe mechanizmy analityczne.