Flags Argument Tree

STUDIO PROJEKTOWE 1

Oskar Pawica

Maciej Kutyła

Wstęp

Celem projektu było stworzenie drzewa wywodu, które na podstawie preferencji użytkownika i analizy obrazu znajduje wybraną wcześniej flagę państwa. Mamy do wyboru 200 państw, a program zadaje maksymalnie 9 pytań “tak/nie”. W każdym kroku zbiór państw spełniających nasze wymogi zmniejsza się około dwukrotnie. Pytania są tak skonstruowane, że nigdy nie zachodzi sytuacja, w której na końcu zostaje więcej niż jedna flaga. Po każdym pytaniu flagi, które nie spełniają naszych kryteriów, stają się półprzezroczyste. Cały czas również widzimy, ile flag spełnia nasze kryteria, a po najechaniu na daną flagę kursorem, wyświetla nam się nazwa danego państwa.

Analiza obrazu

Analiza obrazu została napisana w języku **python**. Skorzystaliśmy z dwóch bibliotek, które znacznie ułatwiały cały proces - **Python Imaging Library** oraz **OpenCV.** Na samym początku zdefiniowaliśmy zakresy podstawowych kolorów w HSV (było to wymagane przez biblioteki, których używaliśmy). W pytaniach o wystąpienie konkretnego koloru, bądź zliczaniu ile kolorów znajduje się na fladze sprawdzamy kolor wszystkich pikseli. Później porównujemy je do wspomnianych już wyżej zdefiniowanych kolorów. Jeśli wartość z piksela jest w danym zakresie, możemy stwierdzić, że dana barwa występuje na fladze. Ta sama zasada działania jest również wykorzystana w pytaniu o procentowe wystąpienie danego koloru na fladze - liczymy po prostu liczbę pikseli o danym kolorze. Do wykrywania kształtu na fladze (w naszym przypadku jest to trójkąt) użyta została biblioteka OpenCV. Obraz jest konwertowany do skali szarości i na podstawie kontrastu możemy jesteśmy w stanie określić czy jest to dany kształt.

API

Całość projektu została zrealizowana w postaci aplikacji webowej. Serwer back-endowy, odpowiedzialny za analizę obrazu postawiono na platformie chmurowej **Heroku**.

Aby móc wywoływać odpowiednie metody analizy obrazu, zaimplementowany został kontroler RestAPI. Wykorzystano do tego celu framework aplikacji webowych **Flask**. Funkcje obsługujące zapytania ze strony front-endu przyjmują obiekty JSON zawierające aktualną listę aktywnych krajów oraz odpowiedź użytkownika na ostatnio zadane pytanie i przekazują je dalej do analizy – celem najefektywniejszego dobrania następnego pytania. Po przeprowadzonej analizie zwracana jest odpowiedź serwera w postaci obiektu JSON zawierającego odfiltrowane kraje i nowe pytanie.

Interfejs graficzny programu

Front-end webowej aplikacji został wykonany w języku JavaScript z użyciem biblioteki **React**.

Interfejs na początku pracy programu i po każdorazowym resecie pytań każdemu państwu przypisuje logiczny status aktywnego, natomiast z każdym pytaniem, odfiltrowane pytania stają się nieaktywne. Jest to sprawdzane przy procesie renderingu – odpowiednim nieaktywnym państwom zostają przypisane właściwości półprzezroczystości.

Każdorazowo po otrzymaniu odpowiedzi od serwera, w której zawarte są państwa spełniające dotychczasowe kryteria użytkownika, renderowane są flagi wraz z odpowiednimi właściwościami.

Bezproblemowe ułożenie flag i responsywność aplikacji zapewnił moduł **React Bootstrap** – czyli przygotowana dla Reacta implementacja jednego z najpopularniejszych i najchętniej wykorzystywanych front-endowych frameworków. Z użyciem komponentów modułu można było również dodać do aplikacji pewne “ozdobniki” w postaci zwizualizowanych stanów ładowania czy interaktywnych, prezentujących nazwy państw podpisów dla flag, co wpływa na pozytywny UX programu.

Zapytania wysyłane do serwera, który dokonywał analizy obrazu, zostały dokonane z użyciem asynchronicznie działającego frameworku **Axios**.

Front-end aplikacji został umieszczony na uczelnianym serwerze **student.**

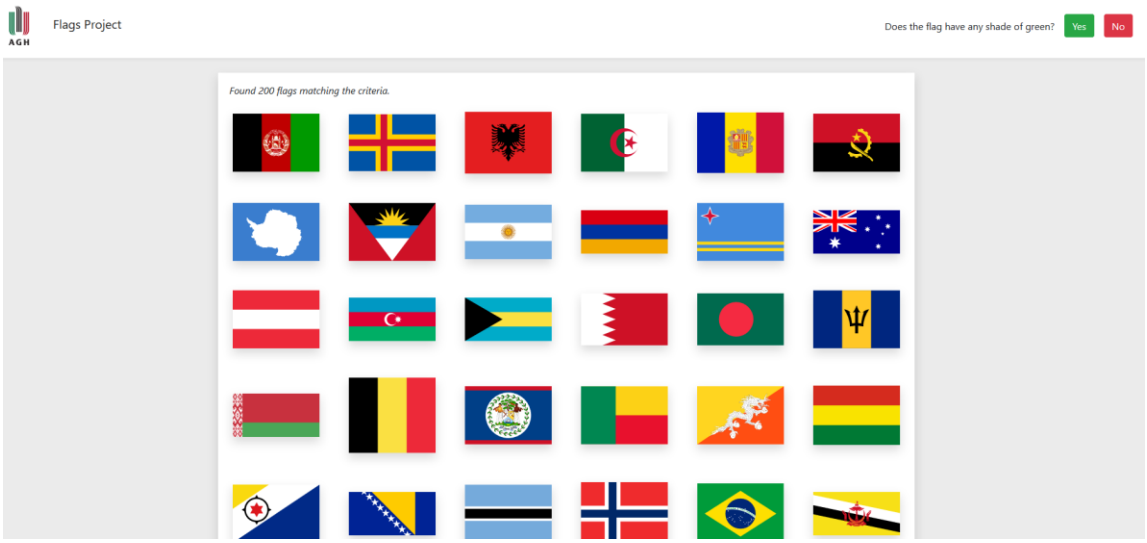
Działanie

Kolejne kroki działania programu można opisać następująco:

1. Start programu i wysłanie do użytkownika pełnej puli 200 państw.
2. Wyświetlenie wszystkich państw wraz z pytaniem.
3. Użytkownik odpowiada na pytanie. Do serwera przesyłane są informacje o pozostałych aktywnych państwach i odpowiedzi użytkownika.
4. Na podstawie odpowiedzi użytkownika przeprowadzana jest filtracja państw (proces filtracji i analizy obrazu został opisany w rozdziale *Analiza obrazu*).
5. Dla aktywnych państw dobierane jest najefektywniej filtrujące kolejne pytanie (takie pytanie, które dokonuje podziału jak najbliższego podziałowi 50/50).
6. Do użytkownika wysyłana jest odpowiedź z odfiltrowanymi flagami i kolejnym pytaniem.
7. Kroki 2-6 są powtarzane, aż zostanie tylko jedna flaga. Po naciśnięciu przycisku **Reset** program rozpoczyna się od początku.

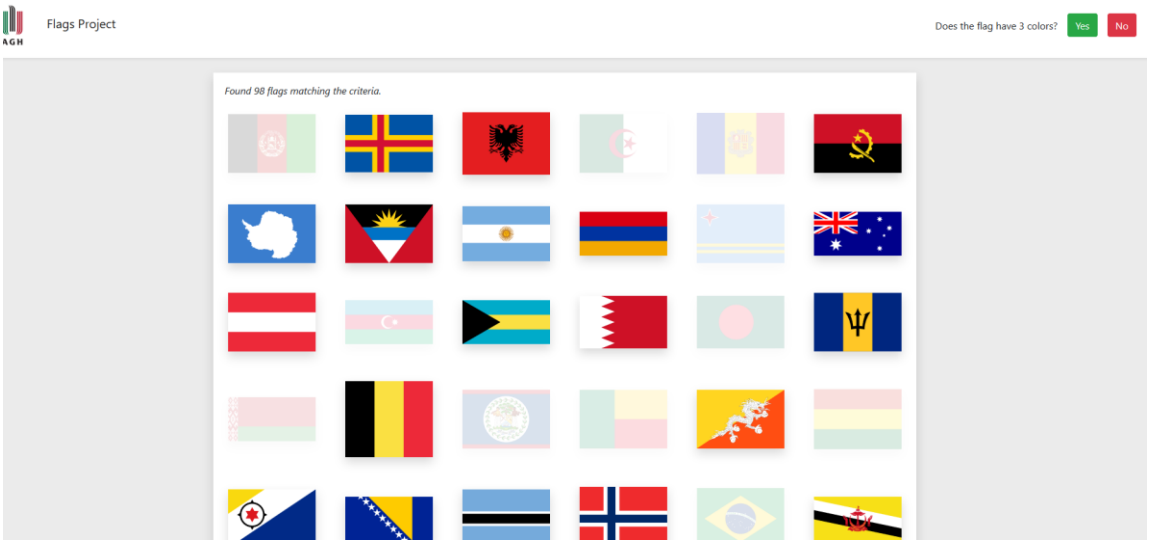
Instrukcja obsługi

Wchodzimy na stronę naszego projektu (link na dole dokumentu) i ukazuje nam się okno startowe. Wciskamy **start** i po chwili ukazuje się nam główna strona.

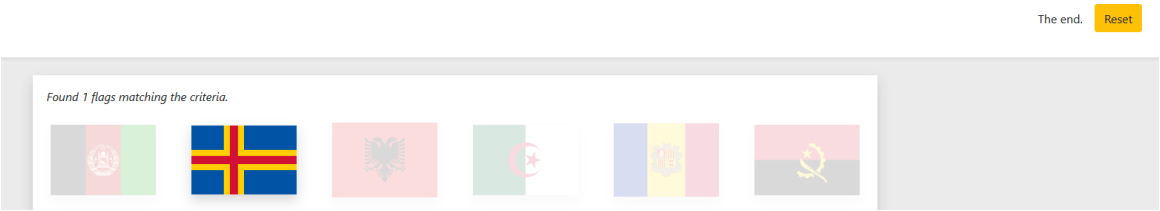


W prawym górnym rogu zadawane są pytania, na które możemy odpowiedzieć tak bądź nie. Nad flagami znajduje się licznik, który pokazuje ile flag spełnia

zadane przez nas kryteria. Możemy również najechać kursorem na daną flagę, by podejrzeć nazwę państwa jakie przedstawia.



Jak możemy zauważyć na załączonym obrazku, bo odpowiedzi na pierwsze pytanie zmieniła się liczba flag spełniających kryteria, a część flag stała się półprzezroczysta. W prawym górnym rogu mam również kolejne, nowe pytanie.



Po znalezieniu wybranej przez nas flagi (w tym przypadku flagi Islandii) w prawym górnym rogu pojawia się możliwość zresetowania wszystkich kryteriów i znalezienia innej flagi.

Podział pracy

Wszelką pracę związaną z projektem staraliśmy się wykonywać razem, będąc na łączach. Za połączenie z API oraz front-end webowej aplikacji odpowiedzialny był Oskar Pawica. Za analizę obrazu oraz dokumentację był odpowiedzialny Maciej Kutyła. Sam algorytm został napisany wspólnie.

Linki:

http://student.agh.edu.pl/~pawicao/flag_project/ - link do programu

https://github.com/pawicao/flag-project-ui - link do kodu programu (front-end)

https://github.com/pawicao/flag-project-api - link do kodu programu (back-end)