

[illegible]

Martin Skalický
2022

Zadání

Popis

Cílem projektu je vytvoření aplikace umožňující vylepšené vyhledávání ve fotografiích uložených na serveru Flickr. Flickr poskytuje aplikační rozhraní umožňující získat prakticky libovolná data o fotografiích, která jsou přístupna i z oficiálního webového rozhraní. Cílem projektu je tedy aplikace, která umožní vyhledávání na serveru Flickr založené na klíčových slovech, stejně jako je tomu na serveru Flickr nyní, ale navíc bude možné zadat výsledky přeradit podle různých metadat.

Rozšíření, které má aplikace nabízet, oproti stávajícímu rozhraní Flickru spočívá v možnosti vložení “doplňujícího” dotazu, podle kterého bude výsledek setříděn. Na výstup vydá fotografie odpovídající danému klíčovému slovu, ale setříděné podle vzdálenosti k zadaným metadatům. Implementovat reranking k úplně každému atributu není potřeba, ale alespoň jeden pro každý datový typ vyskytující se v seznamu datových typů atributů:

- řetězec
- GPS
- integer
- datum

Uživatel bude mít možnost reranking libovolně kombinovat, tj. např. vyhledávat podle vzdálenosti k danému místu a zároveň blízkosti k nějakému datu.

Vzhledem k povaze aplikace bude dbán důraz na kvalitní, uživatelsky přívětivé provedení. Příkladem může být např. možnost zadat souřadnice jinak, než pomocí GPS (např. adresa, nebo kliknutí do mapy), výběr datumu z kalendáře, atp.

Vstup

Klíčové slovo či slova, sada hodnot metadat pro setřídění s váhami a počet výsledků (omezení velikosti výstupu).

Výstup

Fotky, jejichž popis odpovídá klíčovému slovu. Fotky budou setříděné podle vzdálenosti ke zvoleným metadatům.

Požadované vlastnosti

Aplikace by měla obsahovat tyto části:

- Komunikace s Flickr API.
- Identifikace záznamů odpovídajících klíčovému slovu.
- Získání metadat k (identifikovaným) záznamům.
- Výpočty různých typů vzdáleností.
- Třídění podle daných atributů.
- Webový interface

Implementace

Zvolené technologie

Aplikaci jsem se rozhodl napsat jako čistý frontend bez serverové části, protože veškerá data budou získávána z Flickr API a není problém v prohlížeči uložit všechna potřebná data do paměti a to i tak, aby tam zůstala po znovu načtení stránky. Abych mohl vytvořit příjemně interaktivní aplikaci, tak jsem zvolil React knihovnu (také kvůli dobrým zkušenostem s danou knihovnou) pro tvorbu SPA (Single page application). Pro State management jsem vybral Redux, který umožní lépe strukturovat kód a má vestavěnou podporu pro caching http požadavků s nástrojem Redux Toolkit. Aby uživatelské rozhraní mělo příjemný vzhled, tak využiji knihovnu s komponentami s materiálem designem MUI.

Řazení dle metadat

Následující sekce detailně popisuje jednotlivá metadata podle kterých lze podrobněji řadit obrázky. Uživatel může pro jednotlivá metadata vždy zvolit referenční hodnotu (ať již přímo zadáním hodnoty či nepřímo např. výběrem bodu na mapě) a příslušnou váhu. Váha umožňuje nastavit, která metadata jsou pro uživatele důležitější a budou tak mít na řazení větší vliv. Aplikace přerazuje obrázky ihned při jakékoli změně.

Samotný výpočet ohodnocení obrázku probíhá následovně: Pro jednotlivá metadata jsou vypočteny příslušné vzdálenosti (jaké je upřesněno u jednotlivých metadat dále) a následně je provedena normalizace pro jednotlivá metadata - nalezne se maximální hodnota a tou se vzdálenosti podělí. Takto se získá pro každý obrázek x ohodnocení v , kde x označuje počet metadat, ohodnocení v je v rozmezí 0 - 1, kde 0 označuje nejlepší shodu s referenční hodnotou. Následně ještě provádím inverzi $1 - v$, aby 1 znamenala nejlepší shodu, protože jako lidé jsme většinou zvyklí, že vyšší ohodnocení znamená lepší a usnadňuje to interpretaci.

Finální ohodnocení obrázku je vypočteno jako suma všech ohodnocení v , které je vždy vynásobenou příslušnou váhou:

$$\sum_{n=1}^x w_n * v_n = ranking$$

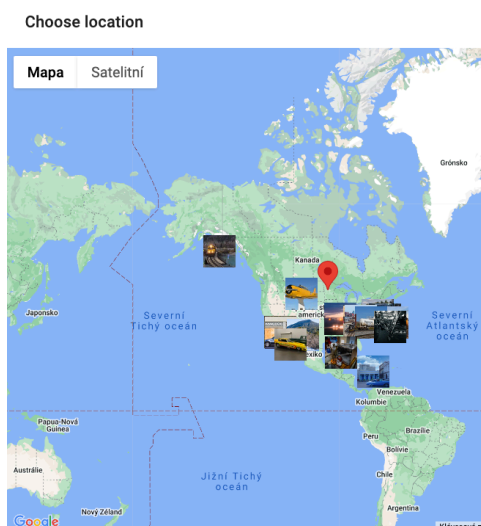
Výsledkem je jedno číslo jako ohodnocení pro každý obrázek, na základě kterého je proveden seřazení sestupně.

Zadání váhy pro jednotlivá metadata je umožněno grafickým posuvníkem, který lépe vizualizuje zvolenou váhu než pouhé číslo. Zvolit je možné váhu od 0 až do 10 se skokem 0,5. Interně pro výpočty je váha normalizována do intervalu $<0, 1>$.



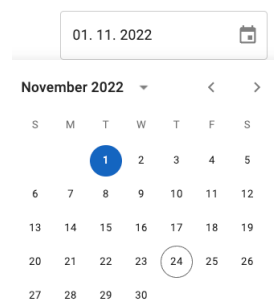
GPS koordináty

Pro nastavení se uživateli zobrazí dialogové okno s Google mapou, na které jsou miniatury všech obrázků umístěných na jejich místo pořízení. Uživatel může podržením levého tlačítka myši vybrat určitý bod, který se nastaví jako referenční. Následně se vypočítá vzdálenost jednotlivých fotek od referenčního bodu. Pro samotný výpočet jsem využil „Haversine formula“. Její výpočet není dokonale přesný a může dojít k maximální chybě 0.5%, protože výpočet předpokládá, že země je pravidelná koule. Mnohem přesnější je např. „Vincenty's formulae“, která počítá se zakřivením země, ale je to iterativní metoda a její výpočet je mnohem složitější. Pro totu danou aplikaci jsem přesvědčen, že „Haversine formula“ je plně dostačující. Aby všechny fotky měly GPS koordináty, tak využívám na Flickr API query parameter díky kterému jsou získány pouze fotky obsahující GPS souřadnice.



Datum pořízení fotografie

Referenční hodnotu může uživatel zadat přímo do textového pole nebo vybrat v interaktivním kalendáři. Vzdálenost se následně vypočítá jako rozdíl referenční hodnoty a datumu pořízení fotografie ve vteřinách. U fotek je čas pořízení vždy uveden.



Šířka fotografie

Přímé zadání hodnoty v pixelech mi nedávalo velký smysl z uživatelského hlediska. Proto jsem zvolil formu tlačítka, které

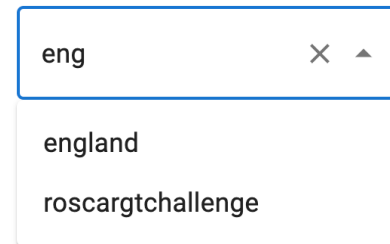
MAX TARGET ▼

MIN TARGET ▲

řadí fotky od nejvyššího rozlišení po nejmenším a opačně. Implementace volí referenční hodnotu v případě nejvyššího rozlišení jako maximální šířku ze všech fotek a pro nejmenšího rozlišení hodnotu 0. Vzdálenost je vypočítána jako absolutní rozdíl referenční hodnoty a šířky fotky. Velmi zřídka, ale při testování se stalo, že šířka fotografie nebyla uvedena a v těchto případech používám vzdálenost 10 000, která je více než případně jakákoli maximální vzdálenost (fotky s vyšším rozlišením než 8k jsem z Flickr API při testování nedostal).

Štítky

Některé fotografie nemají žádný štítek, jiné jich mají desítky. Pro řazení pomocí štítků uživatel zadá referenční textovou hodnotu do vstupního pole, které při psaní interaktivně našeptává nejpodobnější tagy ze všech obrázků. Následně je vypočtena Levenštejnova (editační) vzdálenost jednotlivých štítků vůči referenční hodnotě a nejmenší se vezme jako výsledná pro vzdálenost od referenční hodnoty. Pro Levenštejnovu vzdálenost jsem napsal algoritmus, který je založen na dynamickém programování - jednoduchá varianta vyplňuje tabulku o velikosti délka prvního řetězce n krát délka druhého řetězce m , ale protože se vždy pracuje pouze s aktuálním a předchozím sloupcem v jedné iteraci, tak pro implementaci jsem využil pouze dvojici vektorů o délce nejdelšího řetězce, aby se snížilo využití paměti. Samotná složitost algoritmu je poměrně vysoká $n*m$, naštěstí se zde počítají vzdálenosti pouze pro krátké řetězce, takže to není velký problém. V případě že obrázek nemá uvedený žádný tag, je penalizován vzdáleností 100.



Počet zobrazení

Každá fotografie má u sebe uveden počet shlédnutí. Uživatel má možnost pomocí posuvníku nastavit referenční hodnotu, ke které se mají cílové fotografie blížit se svým počtem shlédnutí. Posuvník je má nejmenší hodnotu rovnou nejmenšímu počtu shlédnutí u fotografií a maximální hodnotu jako nejvyšší počet shlédnutí uvedených u fotografií. Vzdálenost je vypočtena jako absolutní rozdíl referenční hodnoty a počtu shlédnutí u jednotlivých fotografií.



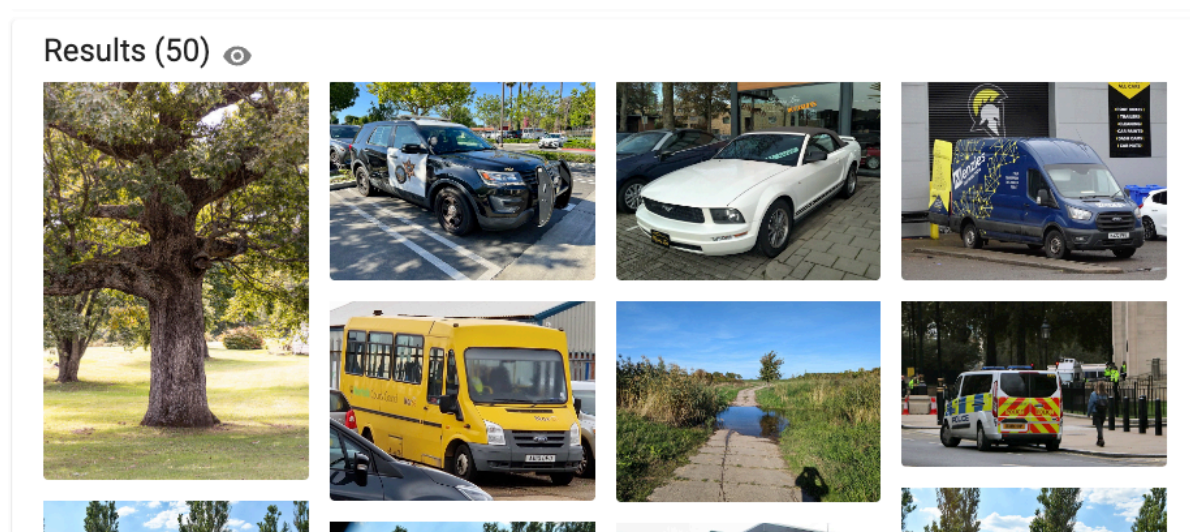
Funkcionalita

Ve vrchní části aplikace je vyhledávací pole, do kterého uživatel zadá vyhledávací výraz, který je předán pro full text search na Flickr API a může zvolit počet obrázků, které chce získat. Limitem je zde Flickr API, které umožňuje

Browse all of photos around the world



najednou vrátit maximálně 500 výsledků. Mohl bych udělat požadavků více najednou, ale nemyslím si, že by to jakýmkoli způsobem pomohlo, protože již v tomto případě při maximálním počtu ke konci jsou obrázky, které s vyhledávacím výrazem nemají moc společného. Po kliknutí na ikonu lupy nebo stisknutí klávesy enter je odeslán požadavek na Flickr. Získané výsledky jsou uloženy do paměti, kde zůstávají i po znovu načtení stránky a obrázky jsou zobrazeny ve spodní části obrazovky do Gridu. Původní záměr byl zobrazit na polovinu obrazovku původní fotografie a na druhou přerazené, aby uživatel mohl vidět vizuálně rozdíl, ale vzhledem k interaktivním povaze aplikace kdy se výsledky přerazují ihned při změně hodnot jsem o tohoto záměru ustoupil.



V závorce u nadpisu je počet získaným výsledků. Vedle nadpisu ikona oka umožňuje zapnutí a vypnutí zobrazení vypočítaných ohodnocení zaokrouhlených na dvě desetinná místa pro jednotlivá metadata a i konečně ohodnocení, které je vypočteno kombinací jednotlivých ohodnocení a příslušných vah. Zobrazené jsou obrázky jsou v nižším rozlišení pro rychlejší načítání. Na obrázky lze kliknout pro zobrazení na celé obrazovce v nejvyšším dostupném rozlišení.

Upravovat řazení pomocí metadat lze ve střední sekci, kde lze zadávat jednotlivé referenční hodnoty a jejich váhy. Při jakékoli změně jsou obrázky ihned přerazeny a uživatel ihned vizuálně vidí jak se obrázky přerazují. Tlačítkem „Reset filters“ lze vynulovat váhy a smazat zadané referenční hodnoty. Zadané hodnoty jsou také ukládány do paměti a zůstávají i po opětovném načtení stránky.

Reranking

GPS location



Date of capture



04. 11. 1924

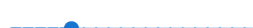


Width (px)



MAX TARGET

Tag



japan



Views

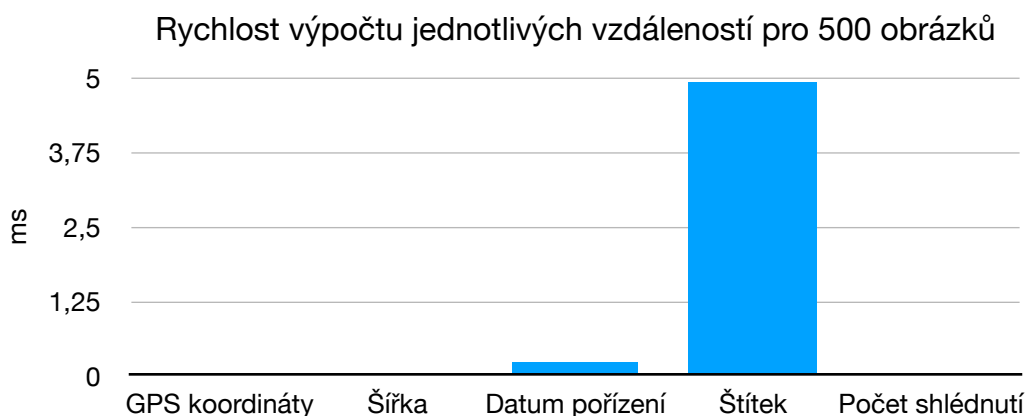


RESET DISABLE

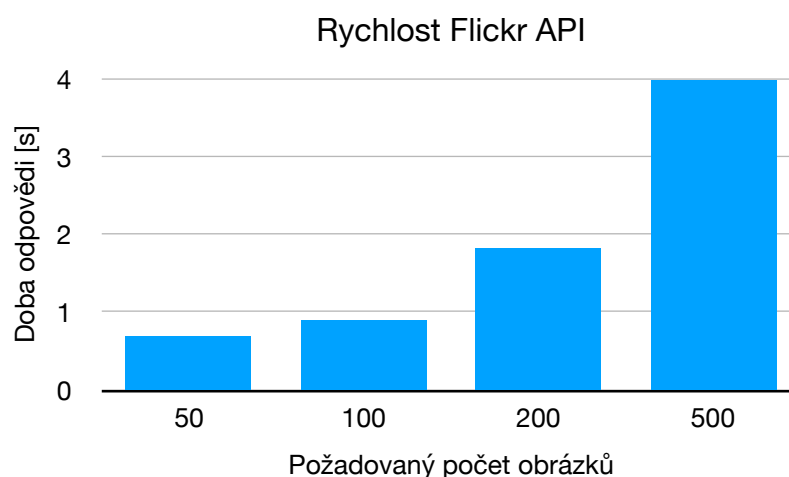
Experimentální část

V zadání bylo doporučeno zaměřit se na případnou paralelizaci, ale vzhledem k rychlosti výpočtů jednotlivých algoritmů pro vzdálenosti, které lze zpracovávat v reálném čase s jedním vláknem zde není potřeba. Obrázky jsou vkládány jako nativní *img* tag, takže zde řeší paralelismus sám prohlížeč. I pokud by byla případně potřeba parcelizace požadavků na Flickr API, tak díky asynchronní povaze jazyka JavaScript, je to otázka přidání tří řádků kódu. Uvedená měření v této sekci byla provedena na zařízení MacBook Air 13“ s arm čipem M1 a webovým prohlížečem Opera ve verzi 92.0.

Změřil jsem rychlosti výpočtů jednotlivých vzdáleností pro 500 obrázků a vynesl do grafu. Hodnoty v grafu jsou průměrem pěti měření. Výpočet konečného ohodnocení z jednotlivých trvá v průměru 0.07 ms.



Z grafu je patrné, že výpočet se pohybuje maximálně v jednotkách milisekund. Počet shlédnutí a šířka je dle očekávání nejrychlejší, protože vzdálenosti je vypočtena jako rozdíl dvou čísel. Následuje výpočet vzdálenosti, který využívá matematický poměrně jednoduchý vzorec pro výpočet. Nejdelší doba výpočtu je strávena na editační vzdálenosti. Jedním z důvodů je složitost samotného algoritmu, která roste mnohem rychleji než lineárně s délkou porovnávaných textů. A druhým důvodem je, že výpočet se pro každou fotku neprovádí jednou jako v případě všech ostatních metadat, ale pro každý obrázek se počítá editační vzdálenost referenční hodnoty a všech uvedených štítků, kterých jsou až desítek u obrázků.



Dále jsem provedl měření rychlosti odpovědi z Flickr API a je zde patrné, že doba čekání na odpověď je řádově delší než samotná doba výpočtu pro re-ranking - sekundy vs. mili sekundy. Toto byl hlavní důvod proč jsem od začátku zavedly využití persistentní paměti pro odpovědi z API. Při zobrazení všech 500 obrázků na stránce se občas objevuje drobné zpomalení (opravdu minimální) - pro vyšší výkon by zde bylo možné zavést tzv. Virtualizovaný grid, který by nezobrazoval všechny fotky najednou ale pouze tu část, kterou si aktuálně uživatel prohlíží.


Shrnutí a závěr

Cílem této práce bylo vytvoření uživatelsky přívětivé aplikace s webovým rozhraním, která má umožnit uživateli najít pomocí fulltext vyhledávání obrázky přes Flickr API s možností přeřazení obrázků na základě jejich metadat. Výsledná aplikace je napsána s pomocí knihovny React s důsledným využitím mezipaměti, aby se omezily požadavky na Flickr API a také zlepšila odezva aplikace. Bylo implementováno napojení na fulltextové vyhledávání Flickr API s možností omezení výsledků. Dále interaktivní přeřazování obrázků pomocí metadat, která jsou uvedena u jednotlivých obrázků. Implementovaný algoritmus přeřazování umožňuje pro následující metadata nastavit referenční hodnotu od které se počítá vzdálenost pro ohodnocení a váhu, podle které uživatel určuje priority pro jednotlivá metadata: GPS koordináty pořízení, šířka fotky, datum pořízení, textových štítků a počet shlédnutí. Výsledné fotografie jsou zobrazeny do Gridu, který se přizpůsobuje velikostí jednotlivých obrázků a dle toho počítá rozložení. Veškeré interakce s rozhraním jako nastavování referenčních hodnot či vah je ihned propagováno a fotky jsou okamžitě příslušně přeřazeny. Na fotky je možné kliknout a zobrazí se přes celou obrazovku v dialogovém okně v plné velikosti.

Dále je umožněno resetovat všechny nastavené hodnoty nebo je pouze vypnout, aby uživatel měl možnost přepínat mezi přeřazenými fotografiemi a původním pořadím. Pro debugging je možno zobrazit overlay přes jednotlivé fotky na kterém se zobrazí vypočtené normalizované ohodnocení pro jednotlivá metadata a také celkové ohodnocení, které vzniklo kombinací jednotlivých a příslušných vah.

Výsledná aplikace je dostupná online na adrese <https://vmm.iotdomu.cz>. Myslím si, že výsledná aplikace působí velmi příjemně z uživatelského pohledu a má logické rozložení. Okamžitě reaguje na změny a přerazuje obrázky, takže uživatel nemusí na nic čekat. Jedna věc se kterou nejsem úplně spokojený je, že jsem nenalezl cestu jak automatizovaně změřit úspěšnost přerazování, což by bylo na samostatnou práci.

Browse all of photos around the world


Type to search... 

50 100 200 500


Reranking

GPS location


Date of capture

04. 11. 1924 

Width (px)

MAX TARGET 


Tag

japan 


Views

RESET DISABLE

Results (50)



location: 1
width: 0
dateTaken: 0
tag: 1
views: 1
final: 1.42



location: 1
width: 0
dateTaken: 0
tag: 1

Aplikace je responzivní a podporuje i zobrazení na telefonu