Cvičení 8: XML-RPC

Dnes vytvoříte webovou službu, ve které budete aplikovat filtry na vaše fotky.

Server bude reprezentovat knihovnu pro práci s obrázky a bude poskytovat metody, které vstupní obrazy nějak upraví.

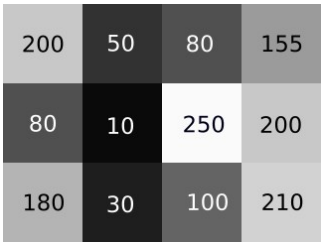
Klient načte obrázek, nahraje na server, volá funkce, které obrázek zpracují, poté obrázek ze serveru stáhne a zobrazí na výstup.

V klientovi je pro práci s obrazy použita knihovna Pillow:
https://pillow.readthedocs.io/en/stable/installation.html
Server už pracuje pouze se seznamem dat, takže tam knihovna není vyžadována.

Co je hotovo?

Klient již obsahuje funkce `load_image` a `save_image`. První zmíněná funkce vyžaduje parametr cestu k obrázku, vrací slovník s klíči `size` a `data` – rozměry obrázku a hodnoty pixelů (reprezentováno jako 1D pole, předpokládejme obrázek ve stupních šedi, hodnoty v rozmezí 0-255).
Např. výstup pro obrázek níže bude následující:

```
{'size': (4, 3), 'data': [200, 50, 80, 155, 80, 10, 250, 200, 180, 30, 100, 210]}
```



Do funkce `save_image` se předá název výstupního obrázku, slovník ve stejném formátu jako výše a obrázek se uloží.

Server vytvoří `XMLRPCServer` a zaregistruje `introspection_functions`.

Co je vaším úkolem?

Vytvořte třídu `ImageProcessing`, která bude mít instanční proměnné pro uložení rozměrů obrázku (buď jako jednu proměnnou nebo zvlášť pro šířku a výšku) a hodnoty pixelů obrázku – takže data ze slovníku, který vznikne při načtení obrázku u klienta. V kostruktoru ale žádná data nepředávejte, instanční proměnné nastavte na `None`.

POZOR: každá funkce na serveru, která bude volána z klienta, musí dát klientovi response, tzn. něco vrátit.

Implementujte následující metody:
`upload_image(img)` – nahraje obrázek na server, tj. uložíte data z `img` do instančních proměnných.
(0.5b)

`download_image()` - stáhne obrázek ze serveru, tj. vrátíte data klientovi. **Toto je jediný způsob, jak půjde data ze serveru získat, následující 2 funkce budou jen provádět operace s obrázkem, ale vracet jej nebudou.**
(0.5b)

`invert_image()` - pokud je nějaký obrázek na serveru, provede obrácení hodnot pixelů, tzn. pokud je hodnota pixelu `p`, bude změněna na `255-p`. Pokud obrázek na serveru není (proměnná je None), vyvolejte výjimku s textem “Image not uploaded”.
(1b)

`edge_detect()` - pokud je nějaký obrázek na serveru, detekuje hrany v obraze. K tomu využijte diference mezi hodnotami pixelu. Pro výpočet diference v osách *x* a *y* použijte následující vzorce:

$$f_x(p) = f(p) - f(p-1)$$
$$f_y(p) = f(p) - f(p-width)$$

kde `width` je šířka obrázku. Při výpočtu dejte pozor, zda nelezete mimo obraz a také, zda v ose *x* od sebe neodečítáte hodnoty sousedních pixelů na různých řádcích (např. 80-155 a 180-200 v případě obrázku výše) – to se dá vyřešit pomocí `p % width`. Výslednou hodnotu, kterou uložíte do obrazu, vypočtěte jako:

$$e(p) = max\{|f_x(p)|, |f_y(p)|\}$$

Pro pixely, kde výpočet nelze aplikovat, tj. první řádek a první sloupec obrazu, nastavte hodnotu 0. Pokud obrázek na serveru není (proměnná je None), vyvolejte výjimku s textem “Image not uploaded”.
(1.5b)

Mějme obraz s názvem ***[NAME].png***.

V klientovi vytvořte spojení na server, uploadněte obraz, proveďte inverzi barev, stáhněte obraz a uložte jako ***[NAME]_inverted.png***. Poté uploadněte původní obraz, proveďte detekci hran, stáhněte a uložte jako ***[NAME]_edges.png***. Pro ***airplane.png*** tedy ***airplane_inverted.png*** a ***airplane_edges.png***. Výsledky by měly vypadat jako na obrázku níže.
(1.5b)

Zdroj obrazu: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/