Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Extrakce označení přídavných látek z obalů potravin

Plzeň 2019 Martin Šíp

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 14. května 2019

Martin Šíp

Abstract

The main goal of this work was to prove, whether it was possible to use optical character recognition library Tesseract OCR (or its alternative) in the task of extraction of food additives from food labels. The outcome of this work is, among other things, a mobile application, which is able to automatically extract food additives from photographs of food labels. The thesis describes the techniques of optical character recognition and image data preprocessing. It also contains the detailed description of the final implementation including the overview of used technologies. The achieved results and future possible extensions are discussed at the end of the thesis.

Abstrakt

Cílem této diplomové práce bylo ověřit možnost využití knihovny Tesseract OCR (či jiné knihovny) pro optické rozpoznávání znaků v úloze automatického rozpoznávání přídavných látek z fotografií složení potravin. Výstupem práce je mimo jiné mobilní aplikace, která dokáže automaticky rozpoznávat přídavné látky přímo z fotografie složení potraviny. V textu práce je popsána problematika optického rozpoznávání znaků, včetně technik předzpracování obrazových dat a způsobu ověření kvality extrakce. V textu je rovněž uveden detailní popis výsledné implementace a použitých technologií. V závěru diplomové práce jsou kriticky zhodnoceny dosažené výsledky a zmíněna případná budoucí rozšíření.

Poděkování

Rád bych poděkoval Ing. Michalu Nyklovi, PhD. za odborné rady a cenné připomínky, které mi pomohly tuto diplomovou práci vypracovat.

Obsah

1 Úvod

S růstem životní úrovně společnosti roste i její poptávka po kvalitnějších potravinách. Tento zájem je reflektován vznikem webových stránek a mobilních aplikací, které se zabývají kvalitou potravin a jejich složením. V souvislosti s kvalitou potravin je často skloňována problematika potravinářských přídavných látek, z nichž některé mohou být i zdraví škodlivé či rakovinotvorné.

Cílem této práce je ověřit možnost využití knihovny Tesseract OCR (či jiné knihovny) pro optické rozpoznávání znaků v úloze automatického rozpoznávání přídavných látek z fotografií složení potravin. Výstupem práce je systém, který dokáže automaticky rozpoznávat přídavné látky přímo z fotografie složení potraviny. V rámci práce byl také navržen a implementován mechanismus pro automatické ověřování úspěšnosti extrakce přídavných látek na základě testovací kolekce fotografií složení potravin, s jehož využitím byla testována kvalita implementovaného systému.

Ve 2. kapitole textu práce je popsána problematika potravinářských přídavných látek. Nalezneme v ní definici tohoto pojmu, historický vývoj potravinářských přídavných látek a jejich aktuální stav. V poslední části této kapitoly je uvedeno několik webových stránek a mobilních aplikací, které se přídavnými látkami zabývají.

Ve 3. kapitole jsou uvedeny a vysvětleny základní techniky předzpracování obrazových dat, jejichž cílem je zvýšit úspěšnost následného optického rozpoznávání znaků. Tyto techniky zahrnují binarizaci fotografie složení potraviny, nápravu jejího zešikmení, zakřivení apod.

Cílem 4. kapitoly je seznámit čtenáře s problematikou optického rozpoznávání znaků. Nalezneme zde stručný popis historického vývoje OCR systémů a popis technik, které OCR systémy používají. V této kapitole je rovněž uveden přehled aktuálních softwarových řešení a dostupných OCR knihoven.

Další kapitoly obsahují návrh systému pro automatickou extrakci přídavných látek z fotografií složení. Nalezneme zde popis jednotlivých komponent systému, jejich vzájemné komunikace, struktury a technologií použitých při jejich implementaci. Rovněž je zde uveden seznam možných případů užití systému a definice jeho uživatelů.

Popis implementace systému pro extrakci přídavných látek je předmětem 7. kapitoly. Způsob testování úspěšnosti extrakce přídavných látek z fotografií složení potravin a zhodnocení dosažených výsledků nalezneme v kapitole 8.

2 Přídavné látky v potravinách

Pod pojmem přídavná látka, respektive aditivum, rozumíme látku přidávanou do jiných látek či směsí za účelem změny nebo vylepšení jejich stávajících vlastností. Přídavné látky jsou nedílnou součástí většiny paliv, maziv a dalších průmyslových výrobků. Bezesporu nejčastěji skloňované přídavné látky jsou ty potravinářské [20].

V této kapitole je stručně vysvětlena problematika potravinářských přídavných látek, viz část 2.1. Dále v této kapitole nalezneme popis několika webových stránek a mobilních aplikací, které se touto problematikou zabývají, viz část 2.2.

2.1 Potravinářská aditiva

Potravinářské přídavné látky, lidově též označovány jako tzv. *éčka*, jsou chemické látky přidávané do potravin s cílem vylepšit nebo zachovat jejich chuť, vůni, konzistenci nebo další vlastnosti. Některá aditiva jsou lidstvu známa a používána již staletí, k jejich největšímu rozmachu však došlo až v období průmyslové revoluce. Průvodními jevy průmyslové revoluce byla urbanizace[1] a strmý nárůst obyvatelstva (například počet obyvatel Anglie se od poloviny 18. stol. do roku 1830 až zdvojnásobil). Protože lidnatá města musela být zásobována potravinami, které byly dopravovány i na větší vzdálenosti a často se zkazily během přepravy, začali prodejci experimentovat s používáním přídavných látek. Tyto experimenty měly často tragické konce. Nezřídka byly do jídla přidávány olovnaté soli, obzvláště pak chroman olovnatý[2], který je pro lidský organismus vysoce toxický. První zákony zakazující jejich používání vešly v platnost až na počátku 19. století [12].

V současnosti je používání přídavných látek v České republice přísně kontrolováno a regulováno vyhláškou Ministerstva zdravotnictví č. 4/2008 Sb. [41], ze dne 3. ledna 2008. Na základě této vyhlášky rovněž musí výrobci potravin uvádět všechny obsažené přídavné látky na obalu svých výrobků a použití potravinářského aditiva je povoleno pouze v těchto případech:

- Je prokázána technologická potřeba jeho použití a účelu nelze dosáh-

[1] Soustřeďování hospodářského i kulturního života do velkých měst na úkor venkova.
[2] Chroman olovnatý je sloučenina kyseliny chromové a olova.

nout jinými prostředky.

- Použité množství nepředstavuje riziko pro spotřebitele.
- Aditivum zachovává výživovou hodnotu potraviny. Záměrné snížení výživové hodnoty je povoleno pouze v případě, kdy taková potravina nepředstavuje podstatnou složku běžné stravy nebo pokud se jedná o potravinu určenou pro speciální výživu.
- Jedná se o složku potřebnou pro výrobu potraviny určené ke speciální výživě.
- Aditivum zvyšuje trvanlivost potraviny nebo její organoleptické[3] vlastnosti, aniž by snižovalo její jakost.

V Evropské unii se pro označení přídavných látek používá kód skládající se z písmene E, tří až čtyř číslic a může být zakončen 1 až 3 písmeny [38]. Čísla obsažená v kódech aditiv jsou používána k jejich kategorizaci. Tabulka 2.1 zobrazuje orientační zařazení aditiv do skupin na základě jejich kódového označení.

Kód	**Skupiny aditiv**
E100–E199	barviva
E200–E299	konzervanty
E300–E399	antioxidanty, regulátory kyselosti
E400–E499	emulgátory, zahušťovadla, stabilizátory
E500–E599	protispékavé látky, regulátory kyselosti, plnidla
E600–E699	látky zvýrazňující chuť a vůni
E900–E999	lešticí látky, sladidla, balicí plyny, propelanty
E1000–E1999	další látky

Tabulka 2.1: Skupiny potravinářských aditiv

Každé aditivum nesoucí E kód bylo podrobeno testům na možné negativní účinky na zdraví. Cílem těchto testů je stanovení hodnoty NOAEL[4], ze které se dále vychází při výpočtu přijatelné denní dávky[5] pro lidský organismus [38].

[3]Charakteristiky, které lze hodnotit lidskými smysly - vzhled, chuť, vůně, atd.

[4]Jedná se o zkratku anglického *No Observed Adverse Efect Level*, přičemž hodnota určuje množství aditiva, které nezpůsobí zdravotní újmu při dlouhodobém podávání pokusným zvířatům.

[5]Množství aditiva, které při přijímaní potravou v průběhu života nezpůsobí žádnou zdravotní újmu.

Ačkoliv nejsou známy případy, kdy by potravinářské přídavné látky používané v povolených množstvích způsobily jakékoliv zdravotní problémy v běžné populaci, nežádoucí účinky a zdravotní komplikace byly pozorovány u osob nemocných a oslabených. Například potravinářská barviva E104, E102, E122 a další byla pozitivně testována na rozvoj hyperaktivity u malé části dětí. Jsou známy i výskyty negativních kožních reakcí, zejména kopřivky, dýchacích a dalších obtíží vyvolaných právě konzumací některých potravinářských aditiv. Vědecký výbor pro potraviny uvádí, že na směs potravinářských barviv, konzervačních látek a aromatizujících látek reagovalo v expozičních testech negativně 2% nezletilých jedinců [42].

2.2 Aditiva a informační technologie

S růstem životní úrovně společnosti roste i její poptávka po kvalitnějších potravinách. Podle dotazníkového šetření [37], které bylo provedeno na Jihočeské univerzitě v Českých Budějovicích, se pouze 12% dotázaných vůbec nezajímá o složení uvedené na obalech potravin, které konzumuje. Na otázku, zda se zajímají o zdravotní účinky potravinových aditiv, 15% respondentů odpovědělo, že ano, 52% uvedlo, že se o ně zajímá příležitostně, a 33% dotázaných vliv přídavných látek na zdraví nikterak nezajímá.

Rostoucí zájem populace o přídavné látky a složení potravin, které lidé konzumují, je reflektován i vznikem webových stránek a mobilních aplikací věnovaných této problematice. Například v České republice jsou to weby *Fér potravina*, *dTest* a *Víš co jíš?*, které jsou popsány v částech 2.2.1 až 2.2.3.

2.2.1 Web FÉR potravina

Jednou z nejpopulárnějších českých webových stránek, která se zabývá potravinářskými aditivy a kvalitou potravin obecně, je web FÉR potravina provozovaný stejnojmennou neziskovou organizací. Web můžete navštívit na adrese *www.ferpotravina.cz*.

Součástí webu je rozsáhlá databáze potravinářských výrobků, umožňující jejich vyhledávání podle různých kritérií, např. na základě jména, EAN[6] kódu, výrobce apod. U každého výrobku zde můžeme nalézt informace o jeho výrobci, dodavateli, nutričních hodnotách, složení, obsažených přídavných

[6] *European Article Number* (EAN), od roku 2009 též označovaný jako *International Article Number* nebo *Global Trade Item Number* (GTIN), česky Mezinárodní číslo obchodní položky, je jednotná mezinárodní číselná identifikace zejména spotřebních výrobků a dalších obchodovatelných položek, která je na výrobky často tisknuta v podobě čárového kódu.

látkách a v neposlední řadě zde nalezneme i hodnocení jeho kvality. Dále FÉR potravina disponuje katalogem přídavných látek. Jednotlivá aditiva jsou vyhledatelná dle jejich názvu či kódového označení. U každého aditiva si můžeme přečíst jeho název, charakteristiku, použití, výrobní proces, možné nežádoucí účinky a hodnocení jeho škodlivosti. Toho hodnocení je provedeno skrze bodovou stupnici (viz tabulka 2.2).

Škodlivost	**Popis**
0	Přírodní látka, získaná přírodní cestou
1	Látka vyskytující se v přírodě, získaná synteticky
2	Syntetická přísada, bez známých vedlejších účinků
3	Přísada nevhodná pro děti či alergiky
4	Přísada podezřelá z vyvolávání alergií, hyperaktivity
5	Přísada pravděpodobně způsobující alergie, hyperaktivitu
6	Přísada, která může mít karcinogenní účinky

Tabulka 2.2: Stupnice škodlivosti aditiv vytvořená spolkem FÉR potravina

Nezisková organizace FÉR potravina provozuje i populární mobilní aplikaci, rovněž nesoucí stejné jméno. Aplikace poskytuje stejnou funkcionalitu jako výše zmíněné webové rozhraní a navíc umožňuje vyhledání potraviny na základě fotografie čárového kódu.

Využitý obsah webu

Pro potřeby diplomové práce byl z webu *www.ferpotravina.cz* získán seznam přídavných látek. Veškerá data byla získána se souhlasem představitelů neziskové organizace FÉR potravina. Získaný seznam přídavných látek obsahuje kódová označení jednotlivých aditiv, jejich názvy a další informace, jenž u přídavných látek tento web uvádí. Pro akvizici seznamu aditiv byl použit nástroj, jenž byl vyvinut v rámci oborového projektu KIV/OPSWI, který této práci předcházel.

2.2.2 Web organizace dTest

Nezisková organizace dTest vznikla v roce 1992 pod názvem Občanské sdružení spotřebitelů. Zpočátku se organizace věnovala pouze vydávání tištěného měsíčníku TEST s testy výrobků a služeb. V současnosti poskytuje komplexní poradenství spotřebitelům a provozuje několik veřejně přístupných databází, včetně katalogu přídavných látek.

Katalog přídavných látek můžeme navštívit na adrese *www.dtest.cz/ecka*. U každého aditiva v katalogu je uveden stručný popis a informace, zda se jedná o bezpečné či nebezpečné aditivum. Tento katalog je rovněž dostupný i uživatelům mobilních telefonů přes aplikaci dTest.

2.2.3 Mobilní aplikace Víš, co jíš?

Mobilní aplikace *Víš, co jíš?* byla vytvořena Odborem bezpečnosti potravin Úřadu pro potraviny Ministerstva zemědělství. Uživatelům umožňuje vyhledat potravinářské přídatné látky podle názvu, E kódu nebo případně podle jejich kategorie.

2.2.4 Cíl této práce

Mobilní aplikace a weby, jenž byly uvedeny v částech 2.2.1 až 2.2.3 umožňují uživateli vyhledávat přídavné látky manuálním zadáním jejich názvů či E kódů do vyhledávacího pole. Tento způsob vyhledávání aditiv vyžaduje úsilí a znalost z uživatelovy strany. Uživatel musí nejprve přečíst text složení potraviny, identifikovat v něm označení přídavných látek a následně jej manuálně zadat do vyhledávacího pole daného webu nebo aplikace. Vzhledem k tomu, že ne vždy jsou přídavné látky potraviny v textu složení zapsány E kódem, musí uživatel apriori vědět, jaká složka vybrané potraviny je aditivum a jaká není.

Aplikace FÉR potravina (viz část 2.2.1) uživateli umožňuje zjistit výskyt přídavných látek v potravině naskenováním jejího EAN kódu prostřednictvím fotoaparátu mobilního telefonu. Vyhledávání dle EAN kódu potraviny nevyžaduje ze strany uživatele přílišné úsilí ani znalost. Uživateli stačí naskenovat čárový kód potraviny fotoaparátem svého telefonu a nechat si zobrazit obsažené přídavné látky. Tento způsob vyhledávání však vyžaduje úsilí ze strany provozovatelů daného webu nebo mobilní aplikace, kteří musí texty složení číst, identifikovat v nich přídavné látky a následně propojit získanou informaci s čárovým kódem potraviny. Vzhledem k tomu, že na trhu neustále přibývají nové potraviny a u existujících potravin se jejich složení neustále mění, se jedná o nekončící proces. Cílem této diplomové práce je vytvoření mobilní aplikace, která by přídavné látky dokázala rozpoznávat automaticky přímo z fotografií složení potravin a snížila by tak na minimum úsilí vyžadované od jejích potenciálních uživatelů a provozovatelů.

3 Předzpracování obrazových dat

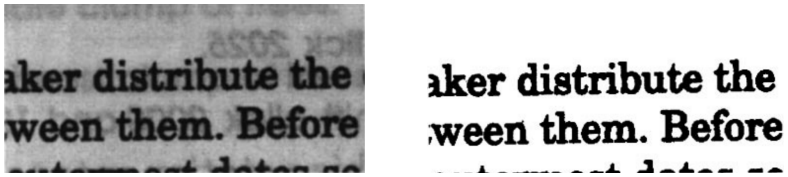
Pro digitalizaci dokumentů a textů byly v minulosti výhradně používány ploché skenery. Situace se ovšem změnila s nástupem digitálních fotoaparátů, které nám umožňují relativně snadno a rychle pořídit obraz textu. Digitální fotoaparáty, zejména pak ty integrované do mobilních telefonů, jsou stále častěji využívány jako alternativa ke klasickým skenerům. Rychlý rozvoj digitálních fotoaparátu není však reflektován vývojem OCR systémů. Pro zvýšení úspěšnosti optického rozpoznávání znaků (viz část 4) je nutné obrazová data pořízená digitálním fotoaparátem či mobilním telefonem předzpracovat a odstranit jejich nedokonalosti či defekty [25]. V částech 3.1 až 3.4 jsou uvedeny techniky a metody předzpracování obrazových dat, které zvyšují úspěšnost následné OCR analýzy.

3.1 Binarizace obrazu

Pod pojmem binarizace obrazu typicky rozumíme převod šedotónového digitálního obrazu na obraz bitonální (např. černobílý). Šedotónový obraz je reprezentován dvourozměrnou maticí, jejíž jednotlivé prvky udávají hodnotu intenzity jasové funkce v daném bodě obrazu. Bývá zpravidla uložen v paměti počítače v osmibitové barevné hloubce, tj. se 256 stupni šedi. Na každý obrazový bod (tzv. pixel) šedotónového obrazu je tedy zapotřebí alespoň 1 byte paměti. V případě bitonálního obrazu, který je též označován jako obraz binární, mohou jednotlivé obrazové body nabývat pouze dvou barev, přičemž typicky se jedná o barvu černou a bílou. Jednotlivé pixely binárního obrazu lze tedy reprezentovat pouze pomocí 1 bitu [13].

Proces binarizace je nedílnou součástí každého OCR systému, neboť výrazně snižuje paměťové a výpočetní nároky a zvyšuje tak efektivitu systému jako celku. Cílem binarizace je segmentace obrazových bodů ve vstupním šedotónovém snímku na dvě skupiny a to sice na černé pixely popředí, reprezentované bitem s hodnotou 1, a na bílé pixely pozadí s hodnotou 0 (viz obr 3.1) [8].

Základní myšlenka binarizace je velmi jednoduchá a spočívá v prahování pixelů šedotónového obrazu [31]. Jinými slovy je stanoven práh θ, jehož celočíselná hodnota se vzhledem k vlastnostem šedotónového obrazu může



(a) Původní (b) Binarizovaný

Obrázek 3.1: Binarizace obrazu

pohybovat v rozmezí hodnot 0 až 255. Hodnota pixelu binárního obrazu $p_{i,j}$ je následně vypočtena prahovacím vztahem 3.1, kde $i_{i,j}$ reprezentuje intenzitu obrazového bodu v původním šedotonovém snímku.

$$p_{i,j} = \begin{cases} 1 & \text{if } i_{i,j} < \theta \\ 0 & \text{if } i_{i,j} \geq \theta \end{cases} \tag{3.1}$$

V závislosti na strategii volby prahu dělíme binarizační algoritmy do dvou hlavních skupin [30]:

(a) Globální prahování: V případě globálního prahování je pro vstupní šedotónový obraz zvolena pouze jedna globální hodnota prahu, která je následně použita pro prahování všech pixelů vstupního obrazu. Globální metody jsou postaveny na předpokladu, že histogram jasu vstupního obrazu je bimodální. Bimodální histogram se skládá ze dvou navzájem dobře rozlišitelných skupin sloupců, které reprezentují třídy pixelů popředí a pozadí (viz obrázek 3.2). Cílem těchto metod je stanovení prahu, který by tyto skupiny, respektive třídy pixelů, optimálně rozděloval. Globální prahovací metody jsou populární pro svou jednoduchost a výpočetní nenáročnost. Jsou úspěšné při binarizaci obrazů, které se skládají z uniformního popředí a k němu kontrastního pozadí. Selhávají však při binarizaci obrazů, které obsahují navzájem nekontrastní popředí a pozadí, a obrazů poškozených nerovnoměrným osvětlením nebo jiným různorodým šumem, neboť v takovýchto případech histogram jasu často nesplňuje předpoklad bimodality. Zástupcem metod globálního prahování je např. Otsuova metoda, viz část 3.1.1.

(b) Lokální prahování: Pokud nelze nalézt globální hodnotu prahu nebo použití jediného prahu pro všechny obrazové body vede k nevalným výsledkům, používáme tzv. lokální prahovací metody. Základní myšlenkou těchto metod je použití několika lokálních prahů při binarizaci vstupního obrazu. Metody zpravidla provedou rozdělení vstupního obrazu do několika navzájem se přesahujících podobrazů menší velikosti a

v každém z nich je poté proveden výpočet lokálního prahu. Podobrazy by měly být dostatečně velké, aby v nich bylo obsaženo jak popředí tak pozadí. Pokud má histogram podobrazu bimodální charakter, je jako práh vybrána hodnota optimálně rozdělující třídy pixelů popředí a pozadí. V případě, že má histogram podobrazu charakter unimodální či jiný, může být hodnota prahu vypočtena pomocí interpolace lokálních prahů sousedních podobrazů. Dalším možným přístupem lokální binarizace je stanovování hodnoty prahu individuálně pro každý pixel vstupního obrazu. Pokud pro každý pixel vstupního obrazu vypočteme individuální hodnota prahu $T(i, j)$, pak hovoříme o tzv. *adaptivním prahování*. Příklady metod lokálního prahování jsou uvedeny v částech 3.1.2 a 3.1.3.

3.1.1 Otsuova metoda

Bezesporu nejpopulárnější globální binarizační metodou je algoritmus navržený Nobuyuki Otsuem v roce 1979, který je označován jako Otsuova metoda [1]. Algoritmus je postaven na předpokladu, že se vstupní šedotónový obraz skládá pouze ze dvou tříd pixelů, tj. pixelů popředí a pozadí. Práh t, oddělující tyto dvě třídy, stanovíme analýzou tvaru histogramu vstupního obrazu (viz obrázek 3.2). Algoritmus hledá hodnotu prahu, která by minimalizovala vážený vnitrotřídní rozptyl (angl. *intra-class variance*). Vážený vnitrotřídní rozptyl definujeme vzorcem 3.2 [13]:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{3.2}$$

Nejprve vytvoříme histogram obrazu a určíme pravděpodobnosti intenzit obrazových bodů $P(i)$, viz vztah 3.3:

$$P(i) = \frac{C_i}{w \times h}, i = 0..255 \tag{3.3}$$

kde i je hodnota intenzity, C_i je počet pixelů obrazu o intenzitě i, w je šířka obrazu a h je jeho výška. Odhad vah $q_1(t)$ a $q_2(t)$ jednotlivých tříd je proveden skrze součty pravděpodobností intenzit pixelů:

$$q_1(t) = \sum_{i=1}^{t} P(i) \qquad q_2(t) = \sum_{i=t+1}^{255} P(i) \tag{3.4}$$

Následně můžeme i vypočítat třídní průměry pomocí vztahů 3.5:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{i=t+1}^{255} \frac{iP(i)}{q_2(t)} \tag{3.5}$$

Dále vycházíme z poznatku, že celkový rozptyl histogramu σ^2 je dán součtem rozptylu vnitrotřídního $\sigma_w^2(t)$ a rozptylu mezitřídního $\sigma_b^2(t)$ (angl. *inter-class variance*) [13]:

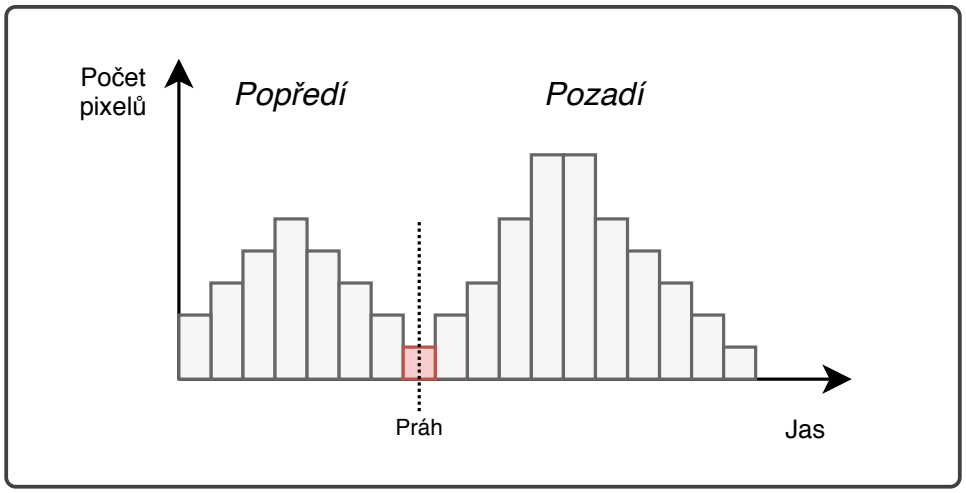
$$\sigma^2 = \sigma_w^2(t) + \sigma_b^2(t) \tag{3.6}$$

Vzhledem k tomu, že je celkový rozptyl konstantní a nezávislý na hodnotě prahu t, lze snadno odvodit, že úloha minimalizace vnitrotřídního rozptylu $\sigma_w^2(t)$ je totéž jako maximalizace rozptylu mezitřídního $\sigma_b^2(t)$, který je dán vztahem:

$$\sigma_b^2(t) = q_1(t)\left[1 - q_1(t)\right]\left[\mu_1(t) - \mu_2(t)\right]^2 \tag{3.7}$$

Máme-li definované potřebné matematické vztahy, můžeme hodnotu prahu zjistit následujícím algoritmem [13]:

1. Vypočítejte histogram a pravděpodobnosti intenzit jasu vyskytujících se v šedotónovém obrazu
2. Inicializujte počáteční hodnoty $\mu_1(0)$, $\mu_2(0)$, $\sigma_1(0)$ a $\sigma_2(0)$, viz vztahy 3.4 a 3.5.
3. Procházejte přes všechny hodnoty prahu $t = 0, ..., I_{max}$
   (a) Aktualizujte hodnoty $\mu_1(t)$, $\mu_2(t)$, $\sigma_1(t)$ a $\sigma_2(t)$
   (b) Vypočítejte hodnotu mezitřídního rozptylu $\sigma_b^2(t)$
4. Vyberte práh t, pro který je hodnota $\sigma_b^2(t)$ maximální



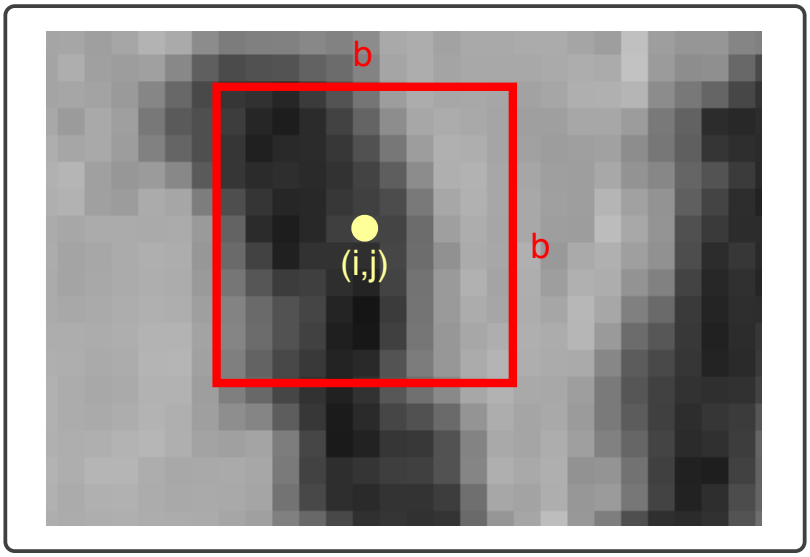
Obrázek 3.2: Znázornění prahu v histogramu

3.1.2 Niblackova metoda

Metoda navržená W. Niblackem v roce 1986 patří k nejstarším dohledatelným zástupcům metod lokálního prahování [31]. Problémem globálních binarizačních metod je, že často odstraňují drobné lokální detaily v binarizovaném obrazu. Tento problém měl odstranit adaptivní prahovací algoritmus W. Niblacka (zkráceně NBM).

Během adaptivního prahování pro každý pixel (i, j) vstupního šedotónového obrazu vypočteme jeho práh $T(i, j)$. V případě NBM je postup pro výpočet prahu $T(i, j)$ založen na konceptu plovoucího okénka předem dané velikosti $b \times b$, viz obr. 3.3. Odhad velikosti prahu provedeme pomocí rovnice 3.8:

$$T(i, j) = m(i, j) + k\sigma(i, j) \tag{3.8}$$



Obrázek 3.3: Plovoucí okénko

V rovnici vystupuje lokální průměr $m(i, j)$ a lokální směrodatná odchylka $\sigma(i, j)$ vypočtené z intenzit pixelů v daném okénku, viz vztah 3.9:

$$m = \frac{1}{b^2} \sum_{k=1}^{b} \sum_{l=1}^{b} B_{kl} \qquad \sigma = \sqrt{\frac{1}{b^2} \sum_{k=1}^{b} \sum_{l=1}^{b} (B_{kl} - m)^2} \tag{3.9}$$

kde B_{kl} je intenzita pixelu v plovoucím okénku. Součástí vztahu 3.8 je i manuálně volený parametr k, který hraje, spolu s velikostí plovoucího okénka b, velmi podstatnou roli.

Parametr k ovlivňuje míru zachování hranic tištěného objektu a jeho absolutní hodnota se pohybuje na intervalu $\langle 0, 1 \rangle$. Malé hodnoty k vedou ke

zmenšování ploch objektů tvořících popředí, což se projevuje výskytem poškozených nebo nekompletních znaků ve výstupním binárním obrazu. Velké hodnoty k se naopak projevují růstem plochy popředí a mohou vést k nečitelnosti textu. NBM je tedy velmi citlivý na volbu parametru k [8].

Za hlavní nevýhodu NBM považujeme nutnost vhodné aproximace velikosti plovoucího okénka. V případě volby příliš malého okénka zanáší algoritmus do binarizovaného obrazu šum a drobné detaily, které nejsou relevantní pro OCR. Pokud naopak použijeme velké okénko, začíná výstup algoritmu degradovat a blíží se výstupu globálních metod [31].

Během svého vývoje zaznamenal NBM řadu změn. Modifikace Niblackovovo algoritmu jsou v současnosti úspěšně používány v komerčních OCR systémech a systémech strojového vidění [15].

3.1.3 Sauvulova metoda

Niblackova metoda, viz část 3.1.2, byla v roce 2000 zdokonalená J. Sauvolou a M. Pietikainenem. Práh $T(i,j)$ v případě Sauvolovy metody vypočteme vztahem 3.10 [14]:

$$T(i,j) = m(i,j) \cdot \left[1 + k\left(\frac{\sigma(i,j)}{R} - 1\right)\right] \tag{3.10}$$

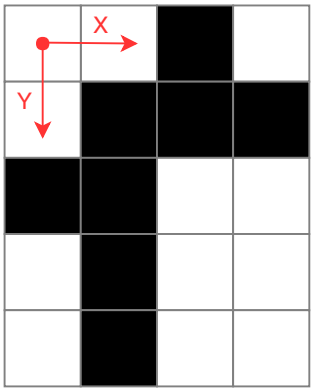
V rovnici opět vystupuje lokální průměr $m(i,j)$, směrodatná odchylka $\sigma(i,j)$ a parametr míry zachování hranic objektu k. Je však představen nový parametr R, jehož hodnota odpovídá maximální možné hodnotě směrodatné odchylky, tj. $R = 128$ pro osmibitový šedotónový obraz.

Použijeme-li algoritmus na oblasti šedotónového obrazu s vysokým kontrastem textu a pozadí, tak bude zřejmě platit $\sigma(i,j) \approx R$, z čehož plyne $T(i,j) \approx m(i,j)$. Při vysokém kontrastu se tedy Sauvulova metoda blíží k NBM. Významnější rozdíl pozorujeme až v případě oblastí s nízkým kontrastem. Při nízkém kontrastu bude zřejmě platit $T(i,j) < m(i,j)$. Výsledná nízká hodnota prahu tak umožní odstranění i relativně tmavých oblastí v pozadí. Sauvulova metoda tedy obecně dosahuje při binarizaci dokumentů s nízkým kontrastem lepších výsledků než NBM [33].

3.2 Binární morfologické operace

Binární obraz vzniklý prahováním šedotónového snímku často obsahuje řadu nedokonalostí. Cílem morfologického zpracování je právě odstranění těchto vad obrazu skrze analýzu tvaru prvků tvořících popředí. Morfologické operace se používají zejména pro odstranění malých objektů v popředí, u kterých

předpokládáme, že vznikly působením šumu, dále k rekonstrukci objektů poškozených a ke zdůraznění struktury objektů (kostra, ztenčování, zesilování) [29]. Morfologické operace lze zobecnit i na šedotónové a barevné obrazy, ačkoliv nejčastěji je používáme právě pro obrazy binární. V takovém případě hovoříme o tzv. **binární morfologii** [21]. Binární morfologické operace jsou implementovány v drtivé většině OCR systémů a systémů strojového vidění. Jejich popularita má základ zejména ve výpočetní nenáročnosti a snadné implementaci.



Obrázek 3.4: Souřadnice v binárním obrazu

Jakýkoliv binární obraz můžeme reprezentovat jako bodovou množinu celých čísel X v Eukleidovského prostoru E^2. Například binární obraz na snímku 3.4 můžeme vyjádřit jako bodovou množinu zapsanou vztahem 3.11:

$$X = \{(0,2),(1,1),(1,2),(1,3),(2,0),(2,1),(3,1),(4,1)\} \tag{3.11}$$

Binární morfologickou transformaci můžeme matematicky formulovat jako relaci s jinou množinou B používanou k prozkoumávání vstupního obrazu. Množinu B označujeme jako **strukturní element** (angl. *structuring element*) [16].

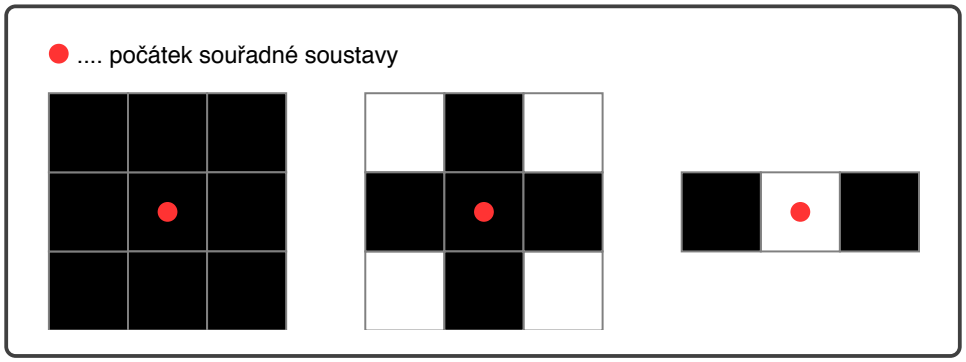
Průběh transformace si můžeme představit, jako by se strukturní element B systematicky pohyboval přes všechny body obrazu X. Bod v obrazu X, který se v daný moment shoduje s počátkem souřadné soustavy strukturního elementu, nazýváme *okamžitý bod*. Výsledek relace mezi strukturním elementem B a obrazem X je zapsán právě do okamžitého bodu [21].

Posunutím strukturního elementu B v binárním obrazu X tak, že jeho počátek souřadné soustavy překrývá bod $x \in X$, vznikne nová bodová množina B_x, kterou můžeme zapsat vztahem 3.12:

$$B_x = \left\{p \in E^2 : p = b + \vec{x}, \forall b \in B\right\} \tag{3.12}$$

Jedná se tedy o množinu vzniklou posunutím původní bodové množiny strukturního elementu B o vektor bodu x [16].

Volba počátku souřadné soustavy a rozložení jednotlivých bodů ve strukturním elementu ovlivňují jeho vlastnosti. Pokud má strukturní element stejné vlastnosti bez ohledu na směr jeho pohybu v obrazu X, tak říkáme, že je *izotropický* (viz obrázek 3.5) [35].



Obrázek 3.5: Izotropické strukturní elementy

Velmi důležitou vlastností všech morfologických operací je tzv. *dualita*. Tato vlastnost znamená, že pro každou morfologickou operaci $\psi(X)$ existuje duální operace $\psi(X)^*$, jejíž vlastnosti vyplývají z množinového doplňku. Dualitu lze formálně zapsat vztahem 3.13 [35]:

$$\psi(X) = (\psi(X^c)^*)^c \tag{3.13}$$

Binárních morfologických operací existuje celá řada. Stěžejní operace jsou **eroze** a **dilatace**, viz části 3.2.2 a 3.2.1. Z nich vychází operace **otevření** a **uzavření**, viz část 3.2.3. Do kategorie morfologických operací rovněž spadají i komplexnější operace, jako např. morfologický gradient, hit-or-miss, skelet apod. viz [35]

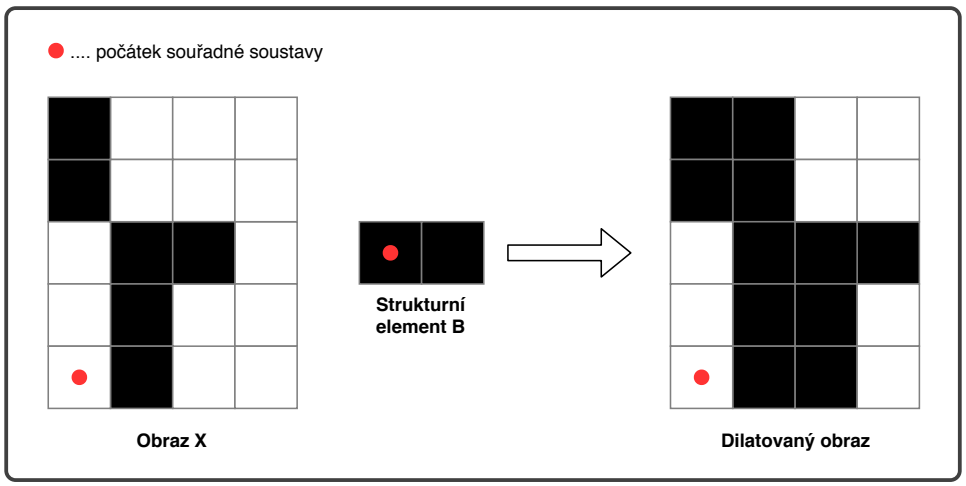
3.2.1 Binární dilatace

Dilatace je jednou ze dvou hlavních operací binární morfologie. Příklad dilatace můžeme vidět na obrázku 3.6. Dilataci obrazu X strukturním elementem B zapíšeme jako $X \oplus B$. Matematicky můžeme dilataci obrazu vyjádřit vztahem 3.14 [35]:

$$X \oplus B = \delta_B(X) = \{x | B_x \cap X \neq \emptyset\} \tag{3.14}$$

kde B_x je bodová množina vzniklá umístěním strukturního elementu v obraze X tak, že počátek jeho souřadné soustavy se kryje s bodem x. Dilatace tedy

označuje množinu bodů x vstupního obrazu X, pro které platí, že umístíme-li v nich počátek strukturního elementu při jeho systematickém posunu, tak dojde k průniku bodů obrazu X a elementu B. Zjednodušeně můžeme říci, že je to množina bodů, ve kterých B zasáhne X. Dilatace, která se řadí mezi



Obrázek 3.6: Binární dilatace

tzv. *rostoucí transformace*, má řadu důležitých vlastností jako jsou [17]:

$$\begin{aligned} \text{Komutativita:} \quad & X \oplus B = B \oplus X \\ \text{Asociativita:} \quad & X \oplus (B \oplus D) = (B \oplus X) \oplus D \\ \text{Rostoucí transformace:} \quad & X \subseteq Y \Rightarrow (X \oplus B) \subseteq (Y \oplus B) \\ \text{Invariance vůči posunu:} \quad & X_h \oplus B = (X \oplus B)_h \end{aligned}$$

Dilatace samotná se používá v systémech strojového vidění, zejména k zaplňování děr a trhlin v objektech tvořících popředí, např. použitím strukturního elementu velikosti 4×4 body, který obsahuje všech 16 bodů, dokážeme zacelit veškeré trhliny v objektu o tloušťce až 3 body (viz obr. 3.7) [21]. Vzhledem k tomu, že dilatace má nežádoucí tendenci zvětšovat objekty na úkor pozadí, není její samostatné použití příliš frekventované. Častěji se používá jako jeden ze základních elementů pro tvorbu komplexnějších morfologických operací.

3.2.2 Binární eroze

Eroze je po dilataci druhou elementární operací binární morfologie [35]. Erozi obrazu X skrze strukturní element B značíme $X \ominus B$. Jedná se o duální operaci k binární dilataci, tj. platí pro ně vztah 3.15:

$$X^C \oplus B = (X \ominus B)^C \tag{3.15}$$

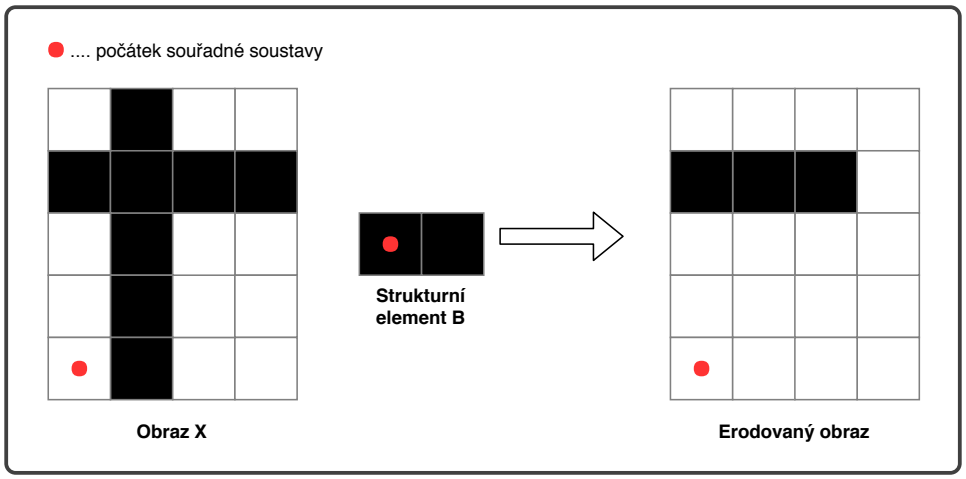
(a) Původní (b) Dilatovaný

Obrázek 3.7: Oprava znaku dilatací

Operaci eroze můžeme zapsat pomocí matematického vztahu 3.16:

$$X \ominus B = E_B = \{x | B_x \subseteq X\} \tag{3.16}$$

Jedná se tedy o množinu bodů x z bodové množiny obrazu X, pro které platí, že je-li umístěn počátek souřadné soustavy strukturního elementu B v daném bodě, pak celý strukturní element je zahrnut v bodové množině obrazu X. Graficky jednoduchý příklad eroze znázorňuje obrázek 3.8.



Obrázek 3.8: Binární eroze

Stejně jako dilatace je eroze rostoucí transformace a rovněž sdílí invariantnost vůči posunu, avšak již není komutativní. Mezi další podstatné vlastnosti eroze patří [17]:

$$\text{Zachování inkluzivity:} \qquad X \subseteq Y \Rightarrow (X \ominus B) \subseteq (Y \ominus B)$$
$$\text{Antiextenzivita:} \qquad (0,0) \in B \Rightarrow (X \ominus B) \subseteq X$$

Eroze se v OCR systémech a obecně v systémech strojového vidění používá k filtraci drobných nežádoucích objektů v popředí. Například erozí se

strukturním elementem velikosti 4×4 body dokážeme ze vstupního obrazu odstranit všechny čáry, drobné objekty a osamělé body tloušťky až 3 body (viz obr. 3.9). [21]

(a) Původní (b) Erodovaný

Obrázek 3.9: Eroze binárního obrazu

K nevýhodám eroze patří, že kromě odstranění drobného šumu a nežádoucích elementů dochází celkově ke zmenšování plochy popředí. Používá se zejména spolu s dilatací jako dílčí prvek složitějších morfologických operací nebo pro extrakci obrysů objektu, které získáme odečtením obrazu erodovaného od původního obrazu. Extrakci obrysu objektu skrze erozi lze matematicky zapsat vztahem 3.17:

$$F = X \setminus (X \ominus B) \tag{3.17}$$

kde F je bodová množina výsledného obrazu s obrysy objektů, B a X jsou množiny strukturního elementu a původního obrysu.

3.2.3 Binární otevření a uzavření

Kombinací eroze a dilatace získáme pravděpodobně nejdůležitější a nejpoužívanější operace binární morfologie, které nazýváme **otevření** a **uzavření** [16]. Binární otevření obrazu X pomocí strukturního elementu B značíme $X \circ B$, uzavření značíme $X \bullet B$.

Binární otevření je používáno zejména k filtraci obrazu. Výstupem je obraz s méně detaily než má obraz původní. Binární otevření má oproti erozi, která je rovněž používána k filtraci obrazu, velkou výhodu. Eroze neodstraňuje pouze nežádoucí prvky obrazu, ale ztenčuje i všechny ostatní. Binární otevření však uvede objekty ztenčené erozí do původního stavu provedením dilatace. Otevření je tedy eroze následovaná dilatací (viz vztah 3.18). [16]

$$X \circ B = (X \ominus B) \oplus B \tag{3.18}$$

Duální operací k binárnímu otevření je binární uzavření. Uzavřením tedy označujeme dilataci následovanou erozí a můžeme ji vyjádřit vztahem 3.19:

$$X \bullet B = (X \oplus B) \ominus B \tag{3.19}$$

Binární uzavření odstraňuje nedostatky dilatace a stejně jako dilatace je používáno k zacelování trhlin a děr objektů popředí. Oproti dilataci se však neprojevuje nežádoucím růstem popředí na úkor pozadí. Díky těmto vlastnostem je metoda uzavření v OCR populární metodou pro opravu rozpadlých znaků.

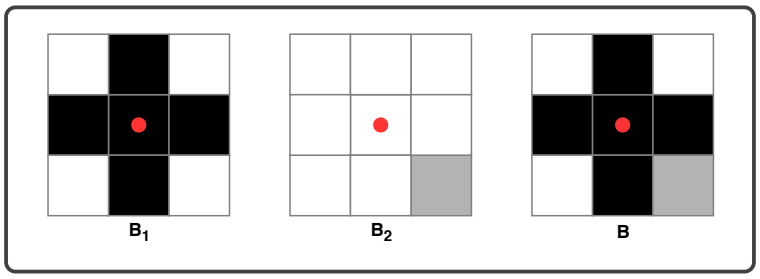
Binární otevření a uzavření jsou rostoucí, idempotentní a navzájem duální transformace [17]. Idempotentnost operace znamená, že její opětovné použití nemá vliv na předchozí výsledek, viz vztahy 3.20 a 3.21:

$$X \circ B = (X \circ B) \circ B \tag{3.20}$$

$$X \bullet B = (X \bullet B) \bullet B \tag{3.21}$$

3.2.4 Tref či miň

Další často používanou morfologickou operací je tzv. operace „tref či miň“ (angl. *hit-or-miss*) [17]. Tato transformace se od předchozích operací odlišuje použitím složeného strukturního elementu $B = (B_1, B_2)$. Kde dílčí množina B_1 reprezentuje vyžadované body náležící objektům popředí a B_2 reprezentuje body, od kterých vyžadujeme, aby náležely pozadí (viz obr. 3.10).



Obrázek 3.10: Složený strukturní element

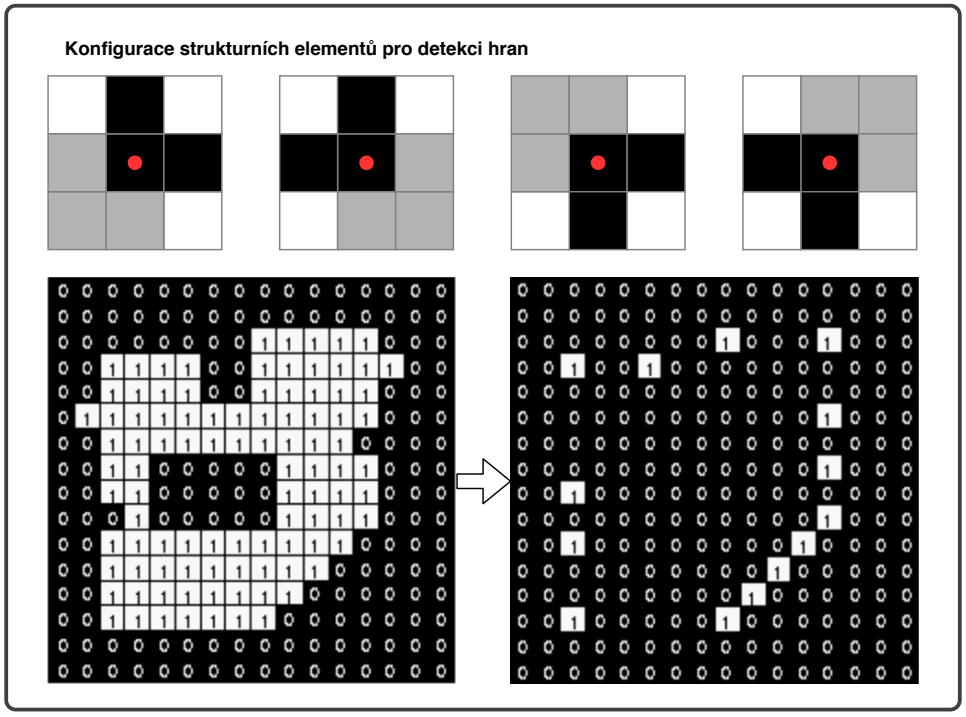
Máme-li takto definované množiny, tak dokážeme transformací „tref či miň“ indikovat shodu strukturního elementu s objekty popředí nebo jejich částmi. Transformaci bodové množiny obrazu X pomocí složeného strukturního operátoru B značíme $X \otimes B$. Matematicky ji můžeme vyjádřit pomocí vztahu 3.22:

$$X \otimes B = \{x \in X : (B_1)_x \subseteq X \wedge (B_2)_x \subseteq X^c\} \tag{3.22}$$

Jedná se tedy o množinu takových bodů $x \in X$, pro které jsou splněny dvě podmínky. Dílčí strukturní element B_1 umístěný svým počátkem v bodu x musí být celý obsažen v bodové množině obrazu X, tzn. celý musí náležet popředí. Druhý dílčí element B_2 rovněž umístěný svým počátkem do bodu x musí celý náležet doplňku X^c, tj. celý musí náležet pozadí. Transformaci „tref či miň“ lze definovat i pomocí eroze a dilatace (viz vztah 3.23).

$$X \otimes B = (X \ominus B_1) \cap (X^c \ominus B_2) \tag{3.23}$$

Cílem „tref či miň“ transformace je extrahovat z binárního obrazu množinu bodů se specifickou konfigurací sousedních pixelů, která je určena právě strukturním elementem. Používá se zejména pro ztenčování a zesilování bodových množin, detekci konvexních hran objektů (viz obr. 3.11), skeletizaci[1] apod.



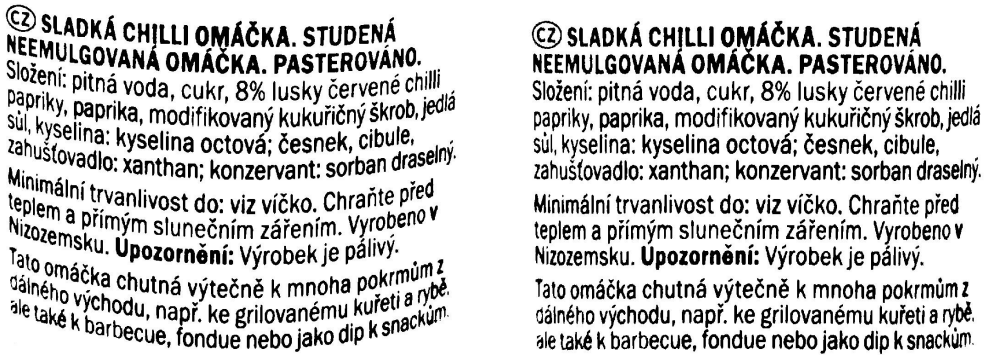
Obrázek 3.11: Strukturní elementy pro detekci hran objektů

3.3 Dewarping

Většina tradičních OCR systémů je postavena na předpokladu, že vstupní obraz dokumentu, jehož text chceme rozpoznat, obsahuje vodorovné řádky

[1]Skeletizace označuje proces redukce objektu na jeho kostru.

textu. V případě, že tento předpoklad není splněn, dochází k rapidnímu poklesu úspěšnosti rozpoznávání. Chceme-li tedy úspěšně aplikovat OCR knihovnu na obraz dokumentu, jehož řádky jsou zakřivené nebo pokroucené, musíme nutně provést jeho předzpracování, které spočívá právě v narovnání zakřivených linií textu (viz obr. 3.12). Proces narovnání křivého obrazu dokumentu označujeme anglickým pojmem *dewarping* [39].

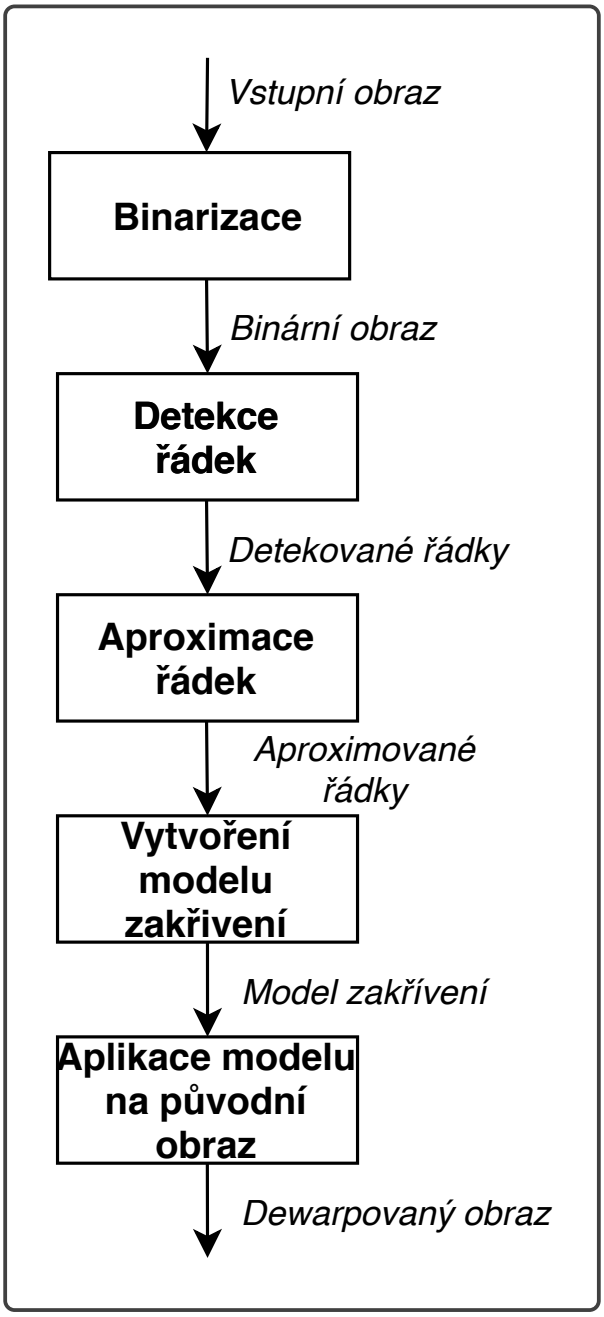


(a) Původní (b) Dewarpovaný

Obrázek 3.12: Dewarping textu

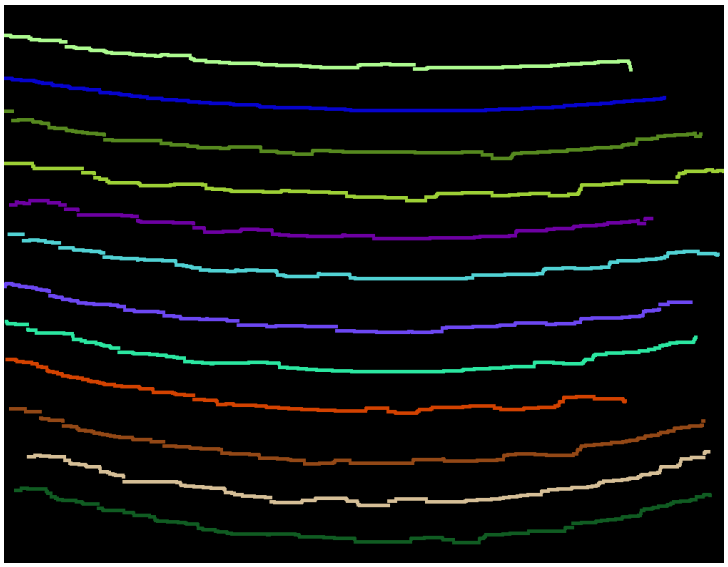
Pro digitalizaci dokumentů byly v minulosti výhradně používány ploché skenery, které skenovaný dokument fyzicky vyrovnají tak, aby spočíval v jedné rovině. Náprava zakřivení skenovaného dokumentu zpravidla nebyla nutná a s tímto předpokladem rovněž operovala drtivá většina OCR systémů. Situace se ovšem změnila s nástupem digitálních fotoaparátů, které nám umožňují relativně snadno a rychle pořídit obraz dokumentu. Digitální fotoaparáty, zejména pak ty integrované do mobilních telefonů, jsou stále častěji využívány jako alternativa ke klasickým skenerům. Dokument focený fotoaparátem ale není vyrovnáván jako v případě plochého skeneru. Taktéž již nejsme omezeni pouze na tištěné dokumenty. Vyfotit můžeme prakticky kterýkoliv text umístěný i na nerovinném povrchu, jako je např. láhev, sloup pouličního osvětlení apod. Rychlý rozvoj digitálních fotoaparátu není však reflektován vývojem OCR systémů, které se zakřiveným textem stále nedokáží úspěšně pracovat. Tento nedostatek tak dal vzniknout řadě metod a přístupů pro narovnání zakřivených dokumentů. [7]

Známou metodou pro narovnání zakřiveného textu je jednoduchý analytický algoritmus navržený Danem Bloombergem [3], s jehož implementací se můžeme setkat v knihovně pro zpracování obrazu *Leptonica*. Zjednodušené schéma Bloombergova algoritmu můžeme pozorovat na obrázku 3.13. Ta vstupní obraz dokumentu nejprve binarizuje. Nad binarizovaným obrazem je následně provedena detekce řádek, přičemž detekovány jsou jak řádky



Obrázek 3.13: Schéma Bloombergovy metody

rovné, tak i řádky zakřivené. Každou nalezenou řádku reprezentujeme množinou bodů (viz obr 3.14). Z obrázku 3.14 je patrné, že detekované řádky mohou mít hrbolatý a nehladký průběh. Dalším krokem je tedy aproximace řádek pomocí parametrické křivky. Ve většině případů je k tomuto účelu



Obrázek 3.14: Výstup detekce řádek

plně dostačující aproximace kvadratickou funkcí, viz vztah 3.24:

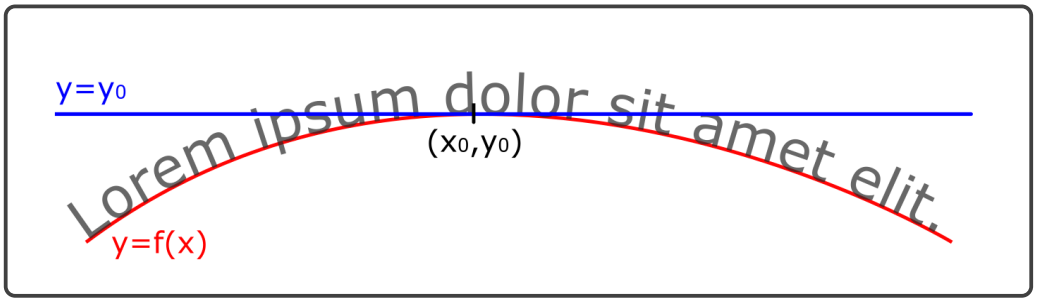
$$f(x) = c_2x^2 + c_1x + c_0 \tag{3.24}$$

kde c_2, c_1 a c_0 jsou konstanty a x je proměnná. Aproximace polynomem vyššího řádu má tendenci zdůrazňovat a zohledňovat tiskové defekty [3].

Nalezneme-li pro každou detekovanou řádku předpis kvadratické funkce, jenž jí aproximuje, pak můžeme přistoupit ke konstrukci modelu zakřivení. V případě Bloombergovy metody pod tímto modelem rozumíme **funkci vertikálního zakřivení** (angl. *vertical disparity function*), kterou označujeme $V(x, y)$. Jedná se o funkci, která každý bod vstupního obrazu mapuje do nové polohy ve výstupním obrazu. Aplikací funkce na všechny body původního obrazu, získáme obraz nový, který již neobsahuje zakřivení [3].

Z vlastností kvadratické funkce víme, že pro ni existuje právě jeden **stacionární bod**, tedy bod z definičního oboru $x_0 \in D_f(x)$ pro jehož hodnotu v první derivaci platí $f'(x_0) = 0$. Hodnotu kvadratické funkce ve stacionárním bodu x_0 označíme $y_0 = f(x_0)$. Mapování bodů řádky probíhá tak, že pro každou řádku určíme hodnotu stacionárního bodu (x_0, y_0) a všechny ostatní body (x, y) ležící na dané řádce mapujeme do bodu (x, y_0) (viz obr. 3.15). Pro každý bod x ležící na detekované řádce dokážeme tedy určit hodnotu funkce vertikálního zakřivení, pro kterou platí $V(x, y) = (x, y_0)$. Vertikální zakřivení bodů nenáležících řádce získáme pomocí interpolace známých hodnot $V(x, y)$ [3].

Kromě analytických metod, kterých existuje celá řada, je na poli dewarpingu rovněž zkoumána možnost využití umělých neuronových sítí a obecně

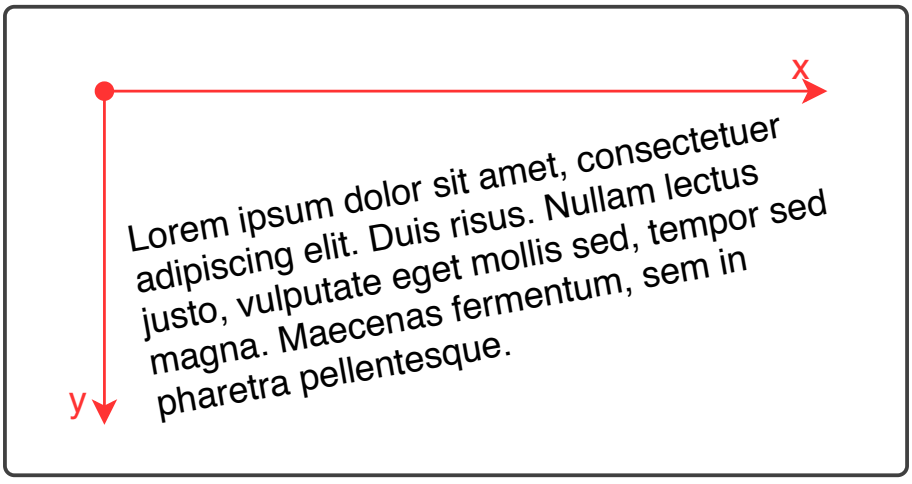


Obrázek 3.15: Stacionární bod

hlubokého učení [22].

3.4 Deskewing

Digitální obrazy dokumentů bývají často zkosené, tj. obsahují navzájem vodorovné řádky, které však již nejsou vodorovné s horizontální osou obrazu (viz obr 3.16). Zkosení obrazu bývá zejména způsobeno špatným umístěním skenovaného dokumentu na ploše skeneru. Je rovněž přítomné u téměř každého obrazu dokumentu, který byl pořízen digitálním fotoaparátem. Proces, který odstraňuje šikmost vstupního dokumentu, nazýváme anglickým slovem *deskewing* [5].



Obrázek 3.16: Zkosený obraz textu

V OCR existují 2 hlavní důvody k narovnání šikmého dokumentu. Prvním důvodem je jeho vzhled. Lidské oko dokáže detekovat i nepatrné zkosení o úhlu 0.25°. Druhým důvodem je, že OCR systémy dosahují na zešikmených dokumentech obecně horší úspěšnosti rozpoznání, zejména pak přesahuje-li

úhel zešikmení 0.57° (0.01 *rad*), neboť nad takovými obrazy hůře provádí segmentaci textu i následné rozpoznání znaků. Řada metod pro detekci a korekci zešikmení v obrazu dokumentu je postavena na *Houghově transformaci* [2], viz následující část 3.4.1.

3.4.1 Houghova transformace

Houghova transformace byla vytvořena profesorem kvantové mechaniky P. Houghem v roce 1959 za účelem detekce přímek a úseček ve fotografiích z bublinkové komory[2] [18]. Později byl algoritmus upraven tak, aby detekoval libovolné tvary, zejména kruhy a elipsy. Její současná podoba, která je dodnes používaná v systémech strojového vidění, byla navržena až v roce 1972 a dostala název „zobecněná Houghova transformace“ [11].

Nejjednodušším případem Houghovy transformace je tzv. lineární Houghova transformace, která slouží k detekci přímek v binárním obrazu. Přímku můžeme vyjádřit směrnicovým vztahem $y = kx + b$. Parametr k se nazývá *směrnice* a platí pro něj vztah $k = \tan\alpha$, kde α je úhel, který přímka svírá s kladnou poloosou x. Vzhledem k tomu, že směrnicovým tvarem nedokážeme popsat vertikální přímku, neboť v takovém případě leží hodnota parametru k v nekonečnu, byl pro potřeby Houghovy transformace zvolen alternativní zápis přímky *Hesseho normální formou* (viz vztah 3.25) [11].

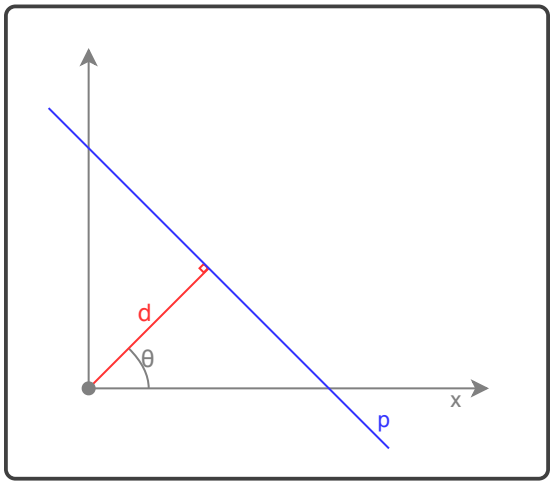
$$d = x\cos(\theta) + y\sin(\theta) \tag{3.25}$$

V Hesseho formě reprezentujeme přímku p dvojicí parametrů (d, θ). Parametr d je vzdálenost mezi počátkem souřadné soustavy a přímkou p. Parametr θ pak označuje úhel mezi osou x a přímkou, která prochází počátkem soustavy a je kolmá k přímce p. Přímka p popsaná Hesseho formou je graficky znázorněna na obrázku 3.17. Parametry detekovaných přímek (d, θ) generují tzv. *Hessův prostor*. Lineární Houghovu transformaci nad binárním obrazem můžeme tedy chápat jako jeho převod do množiny bodů Hessova prostoru.

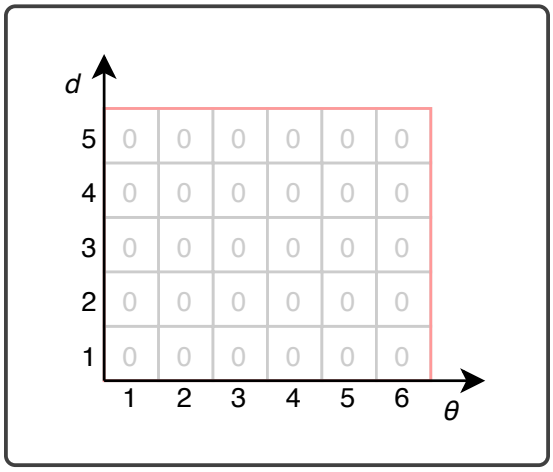
Algoritmus lineární Houghovy transformace používá 2D pole nazvané *akumulátor*. Počáteční hodnota všech skalárních prvků akumulátoru je 0. Jedna jeho dimenze odpovídá parametru θ, druhá parametru d. Vzhledem k nutnosti indexace akumulátoru musejí být hodnoty obou parametrů zaokrouhleny[3] na celá čísla (viz obr. 3.18).

[2]Bubliková komora je nádoba s kapalným vodíkem, která se používá k detekci pohybu elektricky nabitých částic.

[3]Zaokrouhlením dochází ke ztrátě přesnosti. Zde je zaokrouhlení uvedeno pouze pro zjednodušení dalšího postupu. Skutečné implementace používají důmyslnější metody převodu na celé číslo.



Obrázek 3.17: Znázornění parametrů Hesseho formy



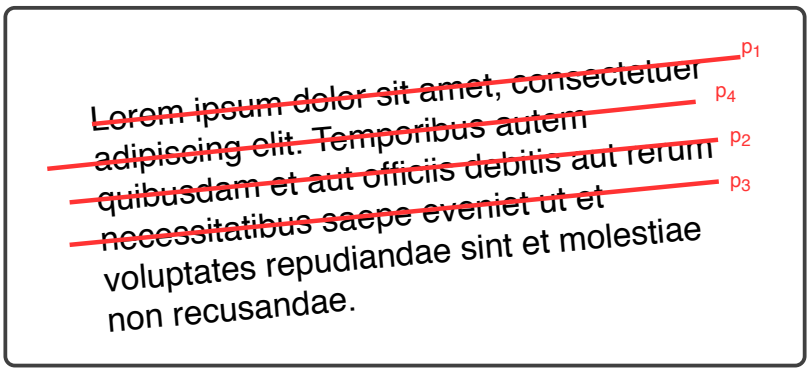
Obrázek 3.18: 2D akumulátor

Houghovův algoritmus prochází přes všechny body binárního obrazu (x, y). V případě, že pixel (x, y) je černý, respektive náleží popředí, hledáme předpisy přímek, které jím prochází. Vzhledem k tomu, že přímek procházejících bodem existuje nekonečně mnoho, musíme velikost úhlu θ omezit pouze na konečnou množinu hodnot M např. $M = \{-20, -19.8, .., 19.6, 19.8, 20\}$, a tak dostaneme i konečnou množinu přímek, které bodem (x, y) procházejí. Pro každý úhel z množiny $\theta \in M$ jsme v bodě (x, y) schopni prostým dosazením do vztahu zjistit i vzdálenost $d = x\cos(\theta) + y\sin(\theta)$, čímž získáme množinu bodů (d, θ) Hessova prostoru. Pro každý bod (d, θ) následně inkrementujeme odpovídající hodnotu v akumulátoru [32].

Po dokončení algoritmu představuje hodnota na pozici (d_1, θ_1) v akumulátoru počet černých bodů binárního obrazu, které přímka o předpisu

$d_1 = x\cos(\theta_1) + y\sin(\theta_1)$ protíná. Nejvýraznější detekované přímky ve vstupním obrazu jsou tedy ty s největší hodnotou.

Použijeme-li Houghův algoritmus nad binarizovaným obrazem dokumentu, pak by nejvýraznější nalezené přímky měly procházet právě skrze řádky textu, neboť na nich pozorujeme vysoký počet černých bodů (viz obr. 3.19). Z úhlů θ prvních N nejvýraznějších přímek jsme pak schopni odhadnout i orientaci samotného dokumentu a provést nápravu jeho zešikmení [32].



Obrázek 3.19: Detekce přímek v dokumentu

4 Optické rozpoznávání znaků

Optické rozpoznávání znaků neboli OCR (angl. *Optical Character Recognition*) je proces klasifikace optických vzorů, které odpovídají alfanumerickým nebo jiným znakům, ve vstupním digitálním obrazu [9]. Jedná se tedy o převod obrazové informace na text. OCR systémy nám umožňují transformovat naskenované dokumenty, PDF soubory či snímky pořízené digitálním fotoaparátem do textové podoby, která umožňuje jejich editaci a vyhledávání. OCR systémy jsou velmi důležitým nástrojem pro digitalizaci textů.

V této kapitole je stručně uveden historický vývoj OCR systémů, viz část 4.1. Dále zde nalezneme popis základních technik používaných v OCR, viz část 4.2. Přehled současných OCR systémů a softwarových knihoven je uveden v části 4.3.

4.1 Historický vývoj OCR systémů

Původ optického rozpoznávání znaků sahá až do roku 1900, kdy se ruský vědec Tyurin pokusil vytvořit zařízení, které by dokázalo převádět text na zvukové signály a umožnilo by tak nevidomým lidem číst pomocí sluchu. Ve svém snažení však selhal. Na jeho pokus úspěšně navázal Edward Fournier d'Albe a vzniklo zařízení, které dostalo název „optofon“ [9].

První skutečné OCR systémy začaly vznikat až v polovině 40. let 20. století. Jejich vývoj šel ruku v ruce s vývojem digitálních počítačů. Tyto jednoduché OCR systémy dokázaly s omezenými výsledky rozpoznávat tištěné znaky a některé z nich i znaky ručně psané. Pro nízkou kvalitu rozpoznávání nebyly tyto systémy však vhodné pro komerční využití.

První komerční OCR systémy začaly vznikat mezi lety 1960 až 1965. Jejich hlavním představitelem byl stroj IBM 1418. Systémy vytvořené v tomto období se označují jako **OCR systémy první generace**. Pro tyto systémy byly vyhrazeny speciální znakové sady, protože texty tištěné jinou znakovou sadou nedokázaly úspěšně rozpoznávat. Rozpoznávání znaků prováděly zejména technikou párování vzorů, viz část 4.2.

Období od roku 1965 až do počátku 70. let se nese ve znamení **druhé generace OCR systémů**. Systémy druhé generace byly schopny rozpoznávat běžně tisknuté znaky a obstojně rozpoznávaly i znaky psané. Rozpoznávání znaků již nebylo omezeno pouze na speciální znakové sady, jako tomu bylo u systému první generace. Nejznámější stroj tohoto druhu byl IBM 1287. Vzhledem k enormní ceně těchto komerčních OCR systémů byly využívány

pouze ve velkých společnostech.

Od roku 1986 se situace na trhu s OCR systémy začala měnit. Díky prudkému nárůstu výkonu výpočetní techniky a poklesu ceny hardwaru začaly být OCR systémy distribuovány ve formě softwarových balíčků. Tyto balíčky byly výrazně levnější, než předchozí konsolidovaná hardwarová řešení, a díky tomu i přístupnější pro širší veřejnost.

K největším pokrokům na poli OCR systémů došlo však až v období 90. let 20. století. Do jejich vývoje byly efektivně zapojeny poznatky z umělé inteligence, elektronického zpracování obrazu a zpracování přirozeného jazyka. Vznikly robustní OCR algoritmy využívající například umělé neuronové sítě, skryté Markovovy modely nebo teorii fuzzy množin. Další historické skutečnosti lze nalézt v [9].

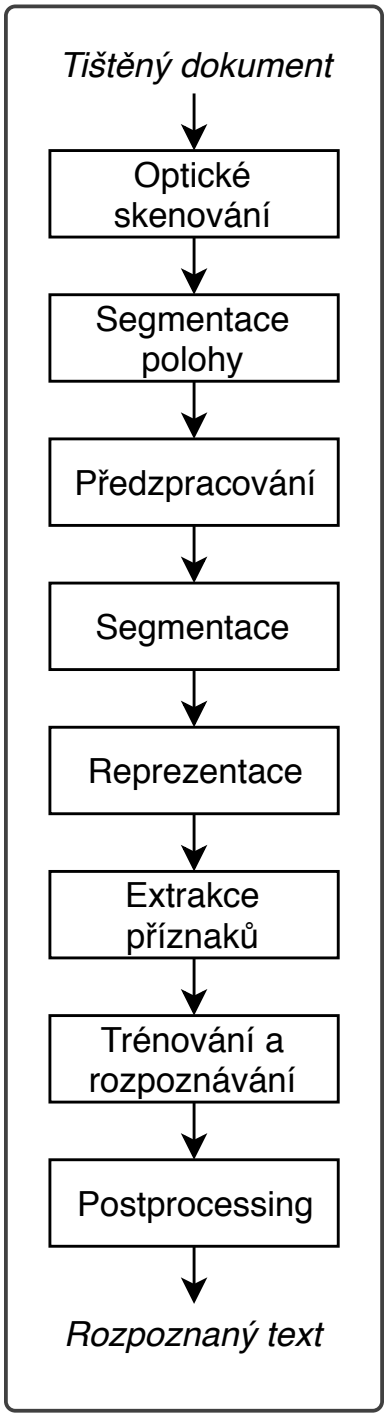
4.2 Komponenty OCR systému

Klasický OCR systém se skládá z několika na sebe vzájemně navazujících komponent, viz obr. 4.1 [9]. Každá komponenta OCR systému je zodpovědná za specifickou fázi optického rozpoznávání znaků. Nejprve proběhne tzv. **optické skenování** (angl. *optical scanning*). Cílem této fáze je získat digitální obraz tištěného dokumentu. Tištěný dokument je digitalizován pomocí skeneru. Výstupem činnosti skeneru je obvykle šedotónový obraz původního dokumentu, který je následně binarizován, viz část 3.1. Proces binarizace je součástí každého OCR systému, neboť výrazně snižuje paměťové a výpočetní nároky a zvyšuje tak efektivitu systému jako celku.

Následuje fáze **segmentace polohy** (angl. *location segmentation*). Cílem této fáze je určit v binárním obrazu dokumentu oblasti, které obsahují text, a separovat je od mimotextových elementů dokumentu, jako jsou například grafy, obrázky, tabulky apod. Výstupem této fáze je obraz dokumentu obsahující pouze textovou informaci.

Ve třetí fázi proběhne **předzpracování obrazu** (angl. *preprocessing*), protože binární obraz dokumentu získaný optickým skenováním a následnou binarizací bývá zatížen defekty. Tyto defekty se negativně podepisují na kvalitě následného rozpoznávání znaků. Základem každého úspěšného OCR systému je proto jejich eliminace, která se skládá z [9]:

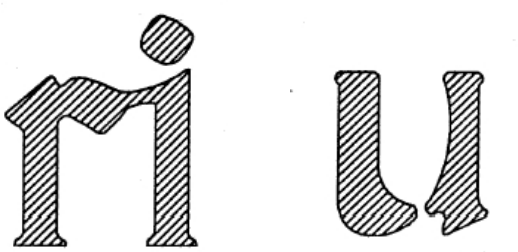
(a) Redukce šumu: Obraz vzniklý optickým skenováním dokumentu a následnou binarizací je často poškozen nežádoucím šumem, který se projevuje zejména erozí nebo dilatací jednotlivých znaků. Vlivem šumu se znaky mohou rozpadnout nebo obsahovat trhliny. Často dochází i k jejich propojení se znaky sousedními, viz obr. 4.2. Těmto defektům lze



Obrázek 4.1: Komponenty OCR systému.

částečně předcházet vhodným výběrem binarizačního algoritmu. Rovněž existují techniky pro jejich nápravu. Rozpadlé znaky jsou nejčastěji opravovány pomocí morfologických operací, viz část 3.2. Dalším nežá-

doucím jevem je výskyt tzv. falešných bodů (angl. *spurious points*), které můžeme vidět na obrázku 4.3. Nejčastější příčinou výskytu falešných bodů v binárním obrazu dokumentu jsou drobné nerovnosti skenovaného povrchu dokumentu nebo špatná vzorkovací frekvence snímacího zařízení skeneru. Falešné body lze relativně snadno odstranit morfologickými operacemi nebo vyhlazením šedotónového obrazu před provedením binarizace.



(a) Spojení (b) Rozpad

Obrázek 4.2: Defekty znaků

Obrázek 4.3: Falešné body v binárním obrazu

(b) Normalizace: Cílem normalizace je zisk standardizovaných dat a odstranění veškerých odchylek, které by mohly snižovat úspěšnost rozpoznávání. Vzhledem k tomu, že většina dokumentů obsahuje znaky různých velikostí, provádí OCR systémy normalizaci jejich velikosti. Dalším druhem normalizace, používaným v OCR systémech, je normalizace šikmosti textu neboli deskewing, viz část 3.4.

Po předzpracování následuje fáze **segmentace textu**, jejímž cílem je segmentace obrazu na jednotlivá slova či znaky. Většina OCR algoritmů segmentuje obraz na jednotlivé znaky, které jsou poté samostatně klasifikovány.

Pátou fází optického rozpoznávání znaků je tzv. **reprezentace**, při které obrazy jednotlivých znaků, které jsme získali segmentací textu, musí být určitým způsobem reprezentovány. V nejjednodušším případě bychom znak

mohli reprezentovat přímo jeho binárním nebo šedotónovým obrazem, ale tento přístup však není výpočetně optimální. Komplexnější OCR systémy se tedy snaží zvýšit úspěšnost klasifikace znaků a snížit složitost a čas jejího výpočtu skrze výběr vhodné reprezentace znaku. Dále musí být splněno, že reprezentace znaků jedné třídy (např. třída znaků „A“) se od sebe liší minimálně, zatímco reprezentace znaků různých tříd (např. „A“ a „B“) se vzájemně velmi odlišují. Techniky reprezentace obrazů znaků lze rozdělit do tří základních kategorií [9]:

(a) Globální transformace: Digitální obraz znaku zpravidla obsahuje více informací, než je nutné pro potřeby klasifikace znaku. Vzhledem k tomu, že digitální obraz není nic jiného než diskrétní signál, tak jej stejně jako signál můžeme reprezentovat lineární kombinací jednodušších matematických funkcí. Koeficienty této lineární kombinace pak tvoří kompaktní kódování, kterým obraz daného znaku reprezentujeme. Mezi nejčastěji používané globální transformace obrazu patří Fourierova transfromace, Gaborova transformace a vlnková transformace (angl. *wavelet transform*).

(b) Statistické reprezentace: Digitální obraz můžeme reprezentovat skrze statistické rozložení jeho obrazových bodů. Tento druh reprezentace je zpravidla odolný vůči změnám stylu písma a je velmi kompaktní. Na druhou stranu již neumožňuje rekonstrukci původního obrazu znaku. Pro statistickou reprezentaci znaku jsou používány například následující metody:

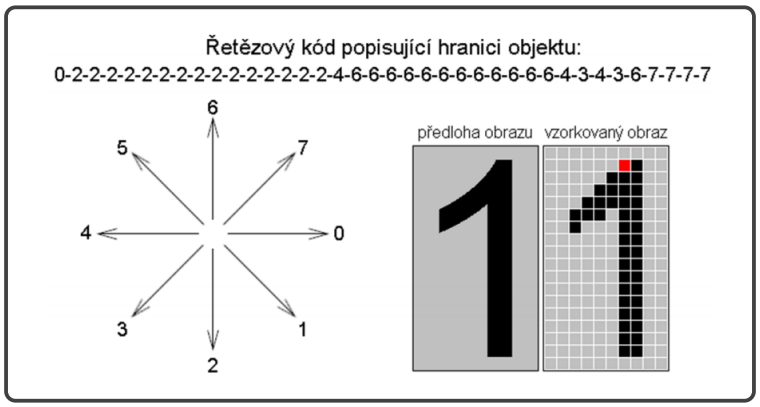
   (i) Zónování (angl. *zoning*): Binární obraz znaku je rozdělen do několika dílčích podobrazů, které se navzájem mohou či nemusí překrývat. V každém podobrazu je následně vypočten určitý statistický ukazatel např. hustota výskytu černých obrazových bodů. Jednotlivé statistické ukazatele dílčích podobrazů poté slouží k reprezentaci obrazu původního.

   (ii) Křížení (angl. *crossing*): Obraz znaku je překryt několika úsečkami obecně různé orientace a délky. Jako statistické rysy reprezentující daný znak jsou poté brány například počty bodů, kterými úsečky prochází. Jedná se o přístup podobný Houghově transformaci, viz část 3.4.1.

(c) Geometrická a topologická reprezentace znaku: Mnohé vlastnosti mohou být reprezentovány pomocí geometrických a topologických metod. Tyto metody můžeme rozdělit do několika skupin [9]:

(i) Extrakce a počítání topologických struktur: Základním rysem této skupiny algoritmů je vyhledávání předem definovaných struktur (např. oblouk, čára apod.) v digitálním obrazu znaku. Počet a vzájemná poloha těchto nalezených strukturních primitiv pak tvoří deskriptivní reprezentaci znaku.

(ii) Kódování: Nejčastěji používané kódovací schéma v oblasti OCR je takzvaný *Freemanův řetězový kód*. Freemanův kód nám umožňuje popsat hranici znaku pomocí řetězce symbolů s určenými směry. Převod binárního obrazu znaku do Freemanova řetězového kódu můžeme vidět na obrázku 4.4.



Obrázek 4.4: Freemanův řetězový kód

(iii) Grafová reprezentace: Znaky jsou nejprve rozděleny na topologická primitiva, jako jsou například úsečky, smyčky apod. Tato primitiva jsou následně reprezentována skrze relační graf.

Šestou komponentou OCR systému je tzv. **extrakce příznaků** (angl. *feature extraction*). Cílem extrakce příznaků je zachycení nejdůležitějších charakteristik znaku, což úzce souvisí s výše uvedenou reprezentací znaků. Některá literatura fázi reprezentace znaku a fázi extrakce příznaků nerozlišuje. Extrahované příznaky jsou hodnoceny z hlediska jejich odolnosti vůči šumu, variacím stylů písma a odolnosti proti rotaci a translaci (posun) znaku. Dalším kritériem je jejich praktická použitelnost. Algoritmus extrakce příznaků nesmí být výpočetně náročný, rovněž výsledné příznaky musí být nenáročné na paměť a čas zpracování. Extrakce příznaků předchází neméně důležité úloze v rozpoznávání znaků, kterou je klasifikace. Klasifikací v OCR rozumíme proces identifikace jednotlivých znaků a jejich přiřazení do odpovídající klasifikační třídy.

Trénování klasifikátoru probíhá tak, že jsou mu prezentovány příznaky znaků z jednotlivých tříd. Na základě těchto příznaků stroj vytvoří prototyp/popis každé třídy. Příznaky extrahované z obrazu znaku, jenž chceme rozpoznat, jsou klasifikátorem porovnávány s prototypy jednotlivých tříd. Neznámý znak je následně přiřazen ke třídě, s jejímž prototypem se nejvíce shoduje. Metody klasifikace se liší v závislosti na zvolené reprezentaci znaku. Základní techniky klasifikace jsou [9]:

(a) Párování vzorů (angl. *template matching*): Základem párování vzorů je porovnávání uložených příznaků (tzv. prototypů), které reprezentují jednotlivé třídy, s příznaky znaku, který má být klasifikován. Párovací techniky jsou klasifikovány do 2 tříd:

   (i) Přímé párování (angl. *direct matching*): Šedotónový nebo binární obraz vstupního znaku je přímo porovnáván s uloženými prototypy jednotlivých tříd. Porovnávání probíhá srze výpočty podobností metrik, jako jsou například Mahalanobisova, Eukleidova a Jaccardova vzdálenost. Znak je následně přiřazen do třídy s největší podobností.

   (ii) Elastické párování (angl. *elastic matching*): Jeden znak se snažíme zdeformovat takovým způsobem, aby co nejvíce připomínal znak druhý, tzn. chápeme jej jako jakousi elastickou šablonu, jejíž kontury chceme zarovnat s konturami znaku druhého. Míra podobnosti znaků je poté odvozena na základě složitosti provedené deformace.

(b) Umělé neuronové sítě: Nejpopulárnějším a nejčastěji používaným klasifikátorem v OCR jsou právě umělé neuronové sítě. Architekturu neuronových sítí lze rozdělit do dvou hlavních skupin a to sice na sítě s dopředným šířením (angl. *feedforward networks*) a na sítě se šířením zpětným (angl. *feedback networks*). Nejběžněji používanými neuronovými sítěmi v oblasti OCR je vícevrstvý perceptron, který patří mezi sítě s dopředným šířením, a Kohonenova samoorganizační mapa, která je zástupce sítí se zpětným šířením. Kohenenova mapa je v poslední době často skloňována zejména pro její schopnost dobře klasifikovat ručně psaných textů [9].

Osmou a poslední komponentou OCR systému je **postprocessing**. Postprocessing v OCR systémech nejčastěji zahrnuje proces seskupování znaků následovaný detekcí a korekcí chyb. Výsledkem prostého rozpoznávání symbolů v textu je seznam individuálních znaků. Tyto symboly musí být vhodně

seskupeny dohromady tak, aby vytvořily jednotlivá slova nebo čísla obsažená v původním textu. Seskupování znaků do řetězců je založeno na jejich vzájemné poloze v digitalizovaném dokumentu. Znaky, které jsou dostatečně blízko sebe, jsou seskupeny dohromady. Tato úloha je dobře řešitelná za předpokladu, že vzdálenost mezi jednotlivými slovy je výrazně větší než vzdálenost mezi znaky, ze kterých jsou slova složena.

Až do procesu seskupování znaků je každý znak zpracováván samostatně a kontext jeho výskytu není analyzován. Pro potřeby pokročilého rozpoznávání textů nejsou systémy založené na identifikaci jednotlivých znaků již dostatečné, neboť ani nejlepší OCR systémy nedokáží se 100% úspěšností identifikovat jednotlivé symboly, a nutně tak dochází k chybám. Tyto chyby lze do určité míry detekovat a opravovat právě analýzou kontextu výskytu jednotlivých znaků. Algoritmy provádějící detekci a korekci chyb lze rozdělit do dvou skupin [9].

První skupina algoritmů zkoumá pravděpodobnost, s jakou se může pozorovaný sled znaků vyskytnout. Pravděpodobnosti výskytu posloupností znaků jsou sestaveny samostatně pro každý podporovaný jazyk. Například v anglickém jazyce je pravděpodobnost výskytu znaku „k“ po znaku „h“ ve stejném slově 0%, pokud je tedy takováto kombinace znaků detekována, OCR systém předpokládá, že se jedná o chybu.

Druhá skupina algoritmů pro detekci a korekci chyb je založena na používaní slovníku platných slov daného jazyka. V případě, že OCR systém rozpozná slovo, které není přítomné ve slovníku, detekuje chybu a následně se dané slovo pokusí nahradit nejpodobnějším platným slovem. Algoritmy využívající slovníkový přístup patří k nejúspěšnějším a nejčastěji používaným korekčním metodám v OCR. Jejich nevýhodou je však to, že nedokáží zachytit chyby, které transformují platné slovo na slovo jiné, vyskytující se také ve slovníku. Dalším negativem je náročnost operace vyhledávání ve slovníku, která se nepříznivě podepisuje na celkové době běhu OCR algoritmu.

4.3 Přehled OCR systémů

V této části jsou stručně uvedeny současné nejznámější nástroje a knihovny pro optické zpracovávání znaků. Největší pozornost je věnována open-source knihovně Tesseract OCR, viz část 4.3.1. Popis komerčních OCR knihoven a systémů nalezneme v částech 4.3.2 až 4.3.4.

4.3.1 Tesseract OCR

Vývoj Tesseractu byl zahájen v laboratořích firmy Hawlett & Packard v anglickém městě Bristol v rámci disertační práce [27]. Firma HP si uvědomila jeho potenciál a mezi lety 1984 až 1994 investovala nemalé prostředky do jeho rozvoje. V roce 2005 byl Tesseract uvolněn pro veřejnost jako otevřený software[1].

Tesseract přišel s řadou revolučních konceptů a technik, které se silně podepsaly na dalším vývoji OCR systémů. Například je známo, že Tesseract není trénován, aby rozpoznával poškozená data, místo toho používá klasifikátor, který je deformované znaky schopen detekovat a nahrazovat. Tento přístup umožňuje Tesseraktu znatelně zmenšit a zefektivnit databázi trénovacích dat [27].

Tesseract podporuje kódování UTF-8 a aktuálně je schopen rozpoznávat více než 100 jazyků. Rovněž je považován za nejpřesnější open-source OCR systém [27]. Oblibu ve vědecké komunitě mu přinesla jeho trénovatelnost. Tesseract lze relativně snadno natrénovat k tomu, aby rozpoznával znakové sady a jazyky, které nejsou v základu podporované.

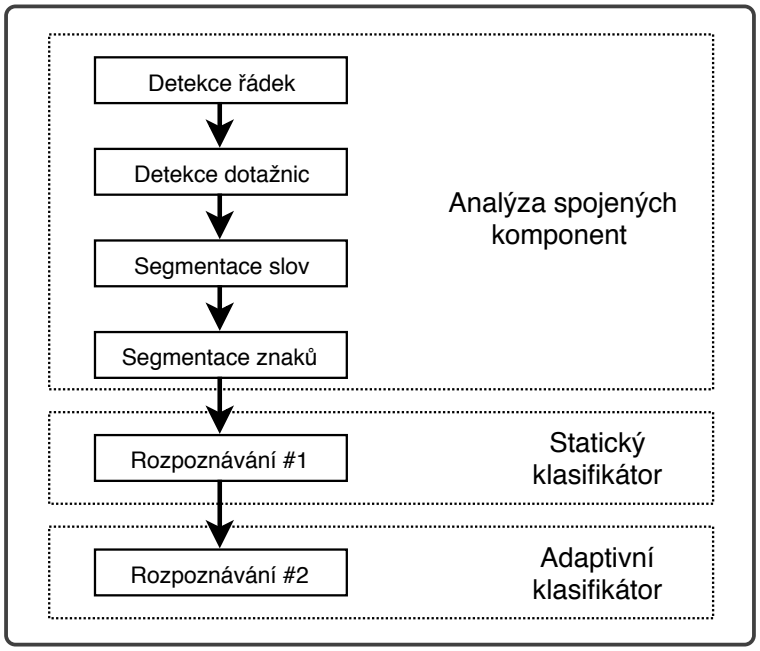
Architektura

Komponenty knihovny Teseract OCR se příliš neodlišují se od komponent klasického OCR systému, viz obr. 4.5. Proces klasifikace je rozdělen do dvou částí. V první části je provedena analýza spojených komponent (angl. *connected component analysis*), která se skládá z detekce řádek, dotažnic a následné segmentace znaků a slov. V části druhé probíhá samotná klasifikace znaků [27].

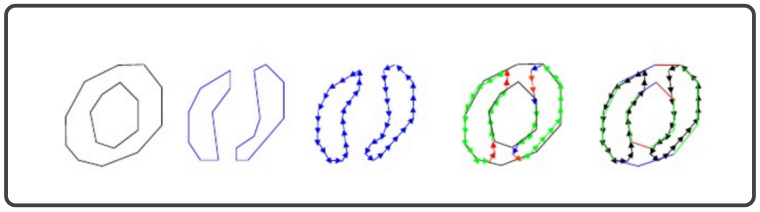
Klasifikace znaků

V první fázi je znak klasifikován statickým klasifikátorem, ve druhé fázi je použit klasifikátor adaptivní. Znak je redukován pouze na obrysy (angl. *outline*). Z těchto obrysů jsou extrahovány příznakové vektory fixní velikosti. Neúplné nebo poškozené obrysy dokáže Tesseract OCR opravit pomocí polygonální aproximace (viz obrázek 4.6) [27]. Každý znak, který byl statickým klasifikátorem úspěšně zařazen do třídy, je následně použit pro trénování adaptivního klasifikátoru. Význam adaptivního klasifikátoru spočívá v jeho schopnosti klasifikace znaků nerozpoznatelných klasifikátorem statickým [27].

[1]Otevřený software (angl. *open-source software*) je software s otevřeným zdrojový kódem. Otevřenost zde znamená jak technickou dostupnost kódu, tak dostupnost legální.



Obrázek 4.5: Komponenty Tesseract OCR



Obrázek 4.6: Oprava obrysů znaku

Podpora jazyků

Tesseract OCR je vyvíjen a testován primárně v anglickém jazyce. Velmi dobrých výsledků dosahuje i při rozpoznávání jazyků, které s anglickým jazykem sdílí společnou abecedu. V případě jazyků nepoužívajících latinské písmo dochází však k patrnému propadu přesnosti rozpoznávání (viz tabulka 4.1). Dnes je Tesseractem podporováno přes 100 jazyků včetně češtiny a neustálé přibývají nové [27].

Výstup Tesseractu

Nástroj Tesseract OCR dokáže prezentovat výstup OCR analýzy obrazu v několika variantách [19]:

- **text UTF-8**: Prostý text, který Tesseract OCR rozpoznal v daném obraze. Text je tvořen nejpravděpodobnějšími výsledky analýzy.

Jazyk	Chybovost rozp. znaků %	Chybovost rozp. slov %
Angličtina	0.47	6.40
Italština	0.54	5.41
Ruština	0.67	5.57
Čínština	2.52	6.29
Hebrejština	3.20	10.58
Japonština	4.26	18.72
Vietnamština	6.06	19.39
Hindština	6.46	28.62
Thajština	21.31	80.53

Tabulka 4.1: Tabulka chybovostí Tesseract OCR z roku 2014 [27]

- **hOCR**: Jedná se o standardizovanou formu zápisu OCR analýzy. Svým formátem připomíná HTML. Obsahuje míru jistoty jednotlivých rozpoznaných slov.
- **pravděpodobnostní mřížka**: Obsahuje míru spolehlivosti jednotlivých rozpoznaných znaků a slov, jejich pozice a možné varianty.

4.3.2 Google Cloud Vision API

Google Cloud Vision API je součástí Google Cloud Platform. Jeho cílem je umožnit vývojářům jednoduchou integraci pokročilých technik strojového vidění s jejich softwarem či aplikací.

Vision API umožňuje detekci a označení objektů vyskytujících se na fotografii. Aktuálně zvládá detekovat tisíce různých objektů od dopravních prostředků až po faunu a floru. Poskytuje funkci detekce obličeje, jejíž součástí je i odhad emočních stavů. Jeho komerčně úspěšnou funkcionalitou je i tzv. moderování obsahu (angl. *content moderation*), které provádí automatickou censuru závadných fotografií sexuálního nebo násilného charakteru.

V neposlední řadě je jeho součástí i OCR modul s automatickou detekcí rozpoznávaného jazyka a podporou více než 12 grafických formátů (od `jpeg` po `tiff`). OCR modul umožňuje i detekci ručně psaného písma, ale tato funkcionalita je zatím pouze v betaverzi.

4.3.3 ABBYY OCR

Moskevská společnost ABBYY byla založena Davidem Yangem v roce 1989 a v současnosti provozuje největší a nejúspěšnější komerční OCR platformu.

Vzhledem ke komerčnímu charakteru tohoto softwaru nejsou blíže známy jeho implementační detaily.

4.3.4 Aspire OCR

Jedná se o komerční OCR systém vyvíjený od roku 1997. Jeho součástí je i SDK[2] umožnující snadnou integraci poskytované OCR funkcionality do aplikací napsaných v jazyce Java, C#, C++, Python nebo Delphi.

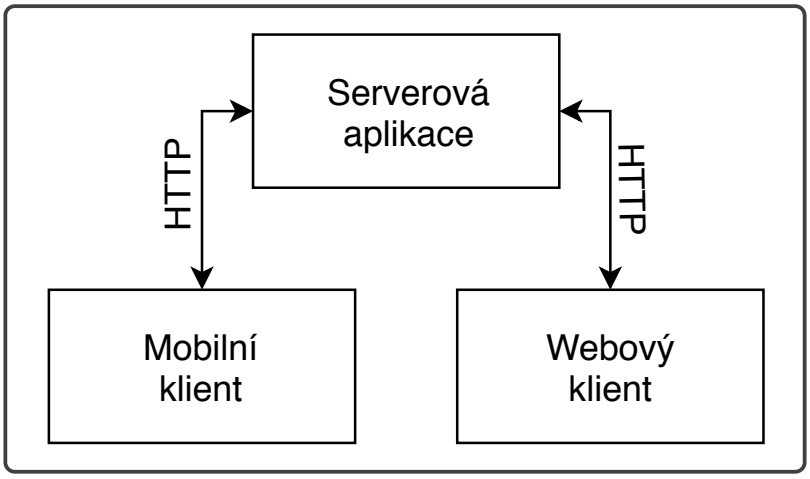
[2]Software development kit (SDK nebo devkit) je typická sada vývojových nástrojů umožňující vytváření aplikací pro určité softwarové balíčky, frameworky, počítačové systémy, herní konzole, operační systémy nebo podobné platformy.

5 Návrh řešení

Výstupem diplomové práce bude systém pro automatickou extrakci přídavných látek z fotografií složení potravin. V této kapitole budou popsány jeho komponenty, uživatelé a základní případy užití, viz část 5.3. Součástí systému bude rovněž mechanismus pro automatické ověření kvality extrahování aditiv z fotografií složení potravin, viz část 5.2.

5.1 Systém pro automatickou extrakci přídavných látek z fotografií složení potravin

Systém se bude skládat ze tří komponent. Jmenovitě bude tvořen mobilním klientem, webovým klientem a serverovou aplikací. Komunikace mezi jednotlivými komponentami bude probíhat prostřednictvím HTTP protokolu. Jednoduché komunikační schéma můžeme vidět na obrázku 5.1.



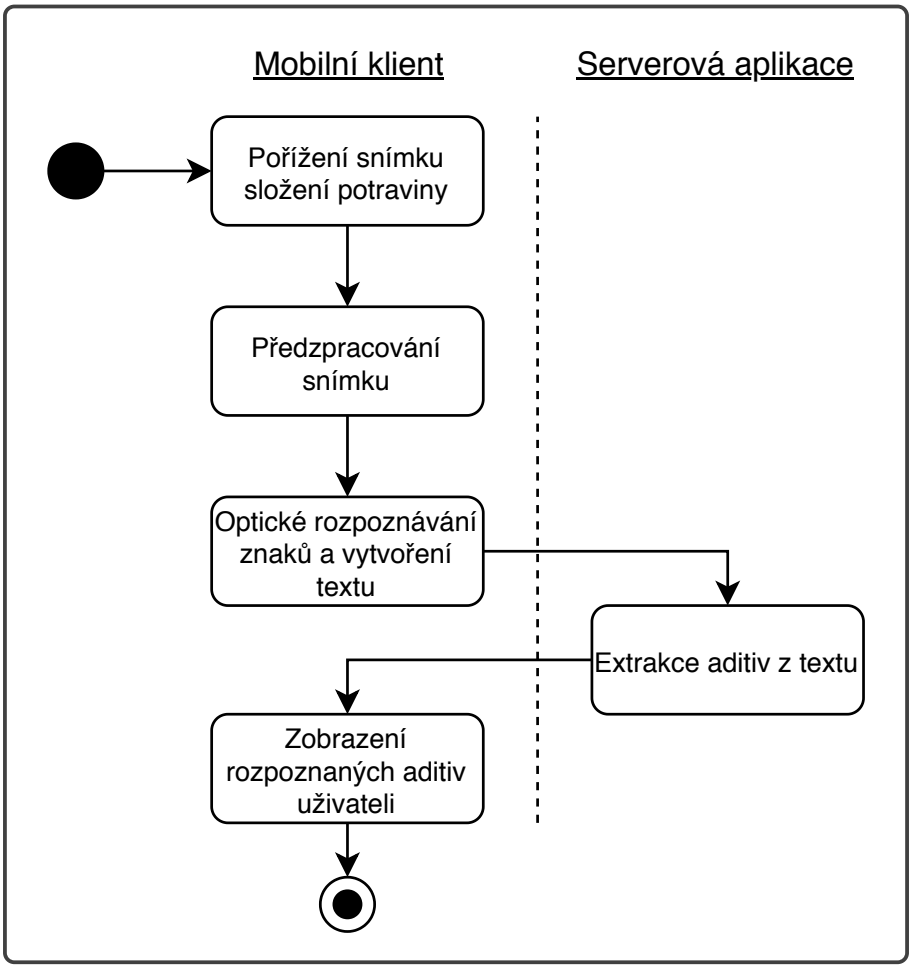
Obrázek 5.1: Komponenty systému

Mobilní klient bude zodpovědný zejména za pořízení snímku složení potraviny, jeho předzpracování a následné rozpoznání textu se složením potraviny, viz část 5.1.1. Primárním účelem serverové aplikace bude správa dat zpracovávaných systémem a jejich ukládání do relační databáze. Zodpovědností serverové aplikace bude rovněž extrakce aditiv z rozpoznaného textu

složení, viz část 5.1.3. Webový klient bude uživateli poskytovat grafické rozhraní pro správu uložených dat a konfiguraci systému a procesu extrakce aditiv, viz část 5.1.2.

5.1.1 Mobilní klient

Mobilní klient bude implementován jako aplikace pro platformu Android. Tato aplikace umožní uživateli pořídit fotografii složení potraviny skrze fotoaparát integrovaný v mobilním zařízení, provede předzpracování pořízeného snímku a následné rozpoznání textu se složením potraviny. Rozpoznaný text bude odeslán serverové aplikaci. Ta z rozpoznaného textu vyextrahuje označení přídavných látek, výsledek extrakce následně předá zpět mobilní aplikaci, která jej zobrazí uživateli. Proces extrakce aditiv je zachycen stavovým diagramem na obrázku 5.2.



Obrázek 5.2: Diagram extrakce aditiv

Kromě rozpoznávání aditiv na snímku složení potraviny bude mobilní aplikace plnit další funkcionality, které vyplynuly ze vzájemné komunikace s vedoucím práce:

- **Přihlášení uživatele**: Funkcionalita aplikace bude přístupná pouze registrovaným uživatelům. Registrovanému uživateli umožní aplikace přihlášení zadáním uživatelského jména a hesla.
- **Registrace uživatele**: Součástí aplikace bude formulář pro registraci nových uživatelů.
- **Vyhledání aditiva**: Aplikace umožní uživateli vyhledávat v katalogu přídavných látek. Každé aditivum v katalogu bude vyhledatelné na základě jeho jména a E kódu.
- **Zobrazení detailu aditiva**: Pro každou přídavnou látku vyhledanou v katalogu nebo extrahovanou ze složení potraviny si bude uživatel moci zobrazit její detailní popis, možné vedlejší účinky apod.
- **Testování**: Aplikace bude rovněž sloužit k automatickému ověřování úspěšnosti extrakce aditiv. Průběh testování je detailně popsán v části 5.2.

Z hlediska mimofunkčních vlastností bude při vývoji aplikace kladen důraz na:

- **Uživatelskou přívětivost**: Výsledná aplikace musí být uživatelsky přívětivá a její používání intuitivní. Uživatel bude aplikací informován o průběhu extrakce aditiv z fotografie.
- **Ošetření vstupů**: Žádný uživatelský vstup nesmí způsobit pád aplikace. Chybové stavy budou ošetřeny a uživatel upozorněn při jejich výskytu.

5.1.2 Webový klient

Webový klient bude implementován v JavaScriptovém frameworku Angular. Primárním účelem webového klienta bude správa dat a konfigurace systému a procesu extrakce aditiv. Jmenovitě webový klient umožní provádění následujících úkonů:

- **Přihlášení uživatele**: Přístup do klientské aplikace bude zabezpečen přihlašovacími údaji.

- **Registrace administrátora**: Klient umožní registraci nového administrátora systému, viz část 5.3.
- **Správa aditiv**: Klient umožní uživateli modifikaci katalogu přídavných látek. Uživatel bude moci přidat nové aditivum, případně modifikovat nebo odstranit existující.
- **Správa uživatelů**: Klient bude poskytovat uživatelské rozhraní pro správu registrovaných uživatelů systému. Registrovaný uživatel bude moci být prostřednictvím webového klienta odstraněn, případně dočasně zablokován.
- **Anotace fotografie složení**: Klient umožní uživateli nahrát fotografii složení potraviny do systému a provést její anotaci. Anotované fotografie složení budou použity pro následné ověření kvality procesu extrakce aditiv, viz část 5.2.
- **Zobrazení výsledků testování**: Uživatel si bude moci prostřednictvím webového klienta prohlédnout výsledky testování úspěšnosti extrakce aditiv.

5.1.3 Serverová aplikace

Primárním účelem serverové aplikace bude správa dat zpracovávaných systémem a jejich ukládání do relační databáze MySQL 8. Server bude klientským aplikacím poskytovat webové služby pro přístup k datům, jejich modifikaci apod. Komunikace mezi serverem a klientskými aplikacemi bude probíhat prostřednictvím protokolu HTTP. Přístup ke službám serverové aplikace bude zabezpečený otevřeným standardem Oauth2, průběh autentizace je blíže popsán v části 6.1.4. Z hlediska procesu extrakce označení přídavných látek bude hlavní zodpovědností serverové aplikace rozpoznání aditiv v textu složení, viz část 5.1.1.

5.2 Ověření úspěšnosti extrakce aditiv

Součástí systému bude mechanismus pro automatické ověření kvality extrahování aditiv z fotografií složení potravin. Automatické testování kvality bude sloužit zejména k vyhodnocování výsledků extrakce aditiv během vývoje systému. Dále bude uplatněn při regresním testování extrakce, jehož cílem bude ověřovat, že změna v implementaci procesu extrahování aditiv negativně neovlivnila jeho úspěšnost, respektive jiné stávající vlastnosti.

5.2.1 Testovací kolekce fotografií složení

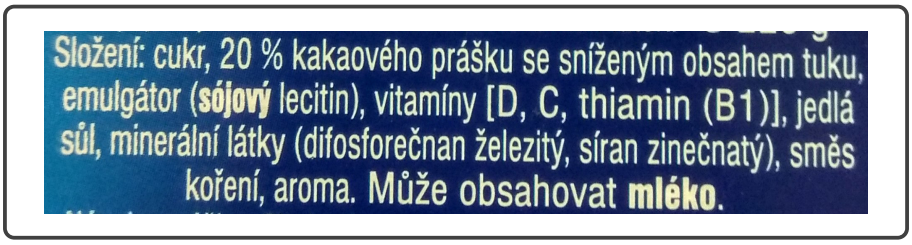
Ověřování úspěšnosti extrakce bude prováděno prostřednictvím testovací množiny fotografií složení potravin. Proces vytváření testovací kolekce bude následující:

1. Pomocí webového klienta nahraje uživatel testovací fotografii na server, zde bude fotografie fyzicky uložena a rovněž pro ni bude vytvořen záznam v databázi.

2. Každou fotografii nahranou na server bude uživatel moci anotovat. Anotaci fotografie provedeme skrze grafické rozhraní webového klienta. Součástí anotace bude manuální označení přídavných látek, které se na snímku vyskytují. Dále v ní nalezneme kompletní přepis vyfotografovaného složení, označení dalších vlastností snímku apod. Detailní popis anotace je součástí oddílu 5.2.2.

3. Testovací množinu budou tvořit plně anotované fotografie.

5.2.2 Anotace fotografie složení potraviny



Obrázek 5.3: Příklad testovací fotografie

Testovací fotografie musí být ještě před nahráním na server manuálně oříznuta tak, aby na ní byl vyobrazen pouze českojazyčný text složení potraviny. Příklad testovací fotografie můžeme vidět na snímku 5.3. Její následná anotace se bude skládat z těchto položek:

- **Přepis textu složení**: Součástí anotace bude kompletní a doslovný přepis textu složení zachyceného na fotografii, tzn. včetně oddělovačů, závorek, interpunkce apod.

- **Obsažená aditiva**: Uživatel označí potravinářské přídavné látky vyskytující se na fotografii složení potraviny, např. v případě snímku 5.3 by měly být označeny následující aditiva: lecitin (E332), difosforečnan železitý (E450).

- **Ohodnocení kvality fotografie**: U každé fotografie bude uvedeno subjektivní hodnocení její kvality/čitelnosti pro uživatele, jež jí anotoval. Kvalita fotografie bude rozdělena do tří úrovní: špatná, průměrná, výborná.

- **Intenzita textu**: Označíme intenzitu barvy textu složení a její vztah k intenzitě pozadí. Uvažujeme dva základní případy: tmavý text na světlém pozadí, světlý text na tmavém pozadí.

- **Zakřivení**: Text složení může být zakřivený, např. jedná-li se o složení uvedené na obalu láhve, či jiném nerovném povrchu. Při anotaci budeme rozlišovat pouze dvě úrovně: nezakřivený text, zakřivený text.

- **Zešikmení**: Zešikmený je takový text, který obsahuje vzájemně vodorovné řádky textu, která však nejsou vodorovné s horizontální osou obrazu. Opět budeme rozlišovat dva stupně: nezešikmený text, zešikmený text.

5.2.3 Způsob ověření úspěšnosti extrakce aditiv

Úspěšnost procesu extrakce bude vyčíslena využitím přesnosti, úplnosti a F-míry [28]. Pro pochopení výpočtu těchto metrik se nejprve musíme seznámit s **maticí záměn** (angl. *confusion matrix*). Účelem matice záměn je zobrazit, v kolika případech se výstup systému shoduje s očekáváním a v kolika případech se systém dopustil chyby, viz obr. 5.4. Symboly „+“ a „-“ signalizujeme výskyt respektive absenci aditiva v dané řádce nebo sloupci.

	Výsledek extrakce	
Výstup anotace	**+**	**-**
+	TP	FN
-	FP	TN

Obrázek 5.4: Matice zmatení

V matici 5.4 můžeme vidět 4 možné druhy případů, které mohou nastat při vyhodnocování úspěšnosti extrakce:

- **TP** - Správně pozitivní: Jedná se o počet aditiv, které systém správně extrahoval. Jinými slovy tak označujeme velikost průniku množiny extrahovaných aditiv a množiny aditiv anotovaných uživatelem.

- **FP** - Falešně pozitivní: Jde o počet přídavných látek systémem falešně extrahovaných z testované fotografie. Jedná se o taková aditiva, která systém extrahoval z fotografie, ale která nejsou uvedena v její manuální anotaci.
- **FN** - Falešně negativní: Označujeme tak počet aditiv, která systém nevyextrahoval z fotografie navzdory tomu, že je uživatel uvedl v její anotaci.
- **TN** - Správně negativní: Jedná se o počet aditiv, která systém nevyextrahoval z fotografie a jež rovněž nejsou uvedena v její anotaci.

Výpočet F-míry pro jednu konkrétní testovací fotografii složení bude proveden dosazením výše určených četností do vztahu 5.1 [28]:

$$F = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{5.1}$$

Případně lze F-míru definovat jako harmonický průměr přesnosti P a úplnosti R (viz vztah 5.2).

$$F = \frac{2 \times P \times R}{P + R} \tag{5.2}$$

Přesnost P a úplnost R vypočteme následovně:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5.3}$$

Běžná F-míra nám vyčíslí úspěšnost extrakce označení přídavných látek pro jednu testovanou fotografii. Pro vyhodnocení úspěšnosti extrakce nad všemi fotografiemi v testovací kolekci budou použity **makro** a **mikro průměr** získaných hodnot [26].

Makro průměr F-míry je obyčejný aritmetický průměr dílčích F-mír F_t jednotlivých fotografií v testovací kolekci o počtu T fotografií, jeho výpočet provedeme pomocí vztahu 5.4:

$$F_{\mathrm{macro}} = \frac{\sum_{t=1}^{T} F_t}{T} \tag{5.4}$$

Micro průměr F-míry vypočteme dosazením do vzorce 5.5:

$$F_{\mathrm{micro}} = \frac{2\mathrm{TP}_{\mathrm{total}}}{2\mathrm{TP}_{\mathrm{total}} + \mathrm{FP}_{\mathrm{total}} + \mathrm{FN}_{\mathrm{total}}} \tag{5.5}$$

kde $\mathrm{TP}_{\mathrm{total}}$, $\mathrm{TP}_{\mathrm{total}}$, $\mathrm{FN}_{\mathrm{total}}$ jsou součty dílčích hodnot dosažených pro fotografie v testovací množině, viz vztah 5.6:

$$\mathrm{TP}_{\mathrm{total}} = \sum_{t=1}^{T} \mathrm{TP}_t \qquad \mathrm{FP}_{\mathrm{total}} = \sum_{t=1}^{T} \mathrm{FP}_t \qquad \mathrm{FN}_{\mathrm{total}} = \sum_{t=1}^{T} \mathrm{FN}_t \tag{5.6}$$

Obdobně budou vypočítány i mikro průměry přesnosti a úplnosti, viz vztah 5.7:

$$P_{\text{micro}} = \frac{\text{TP}_{\text{total}}}{\text{TP}_{\text{total}} + \text{FP}_{\text{total}}} \qquad R_{\text{micro}} = \frac{\text{TP}_{\text{total}}}{\text{TP}_{\text{total}} + \text{FN}_{\text{total}}} \tag{5.7}$$

Mikro průměr má při vyhodnocování úspěšnosti extrakce nad kolekcí, jejíž fotografie složení se navzájem znatelně odlišují počtem obsažených aditiv vetší výpovědní hodnotu než makro průměr [26].

5.2.4 Proces ověření úspěšnosti extrakce aditiv

Proces automatického testování úspěšnosti extrakce aditiv se bude skládat z těchto dílčích kroků:

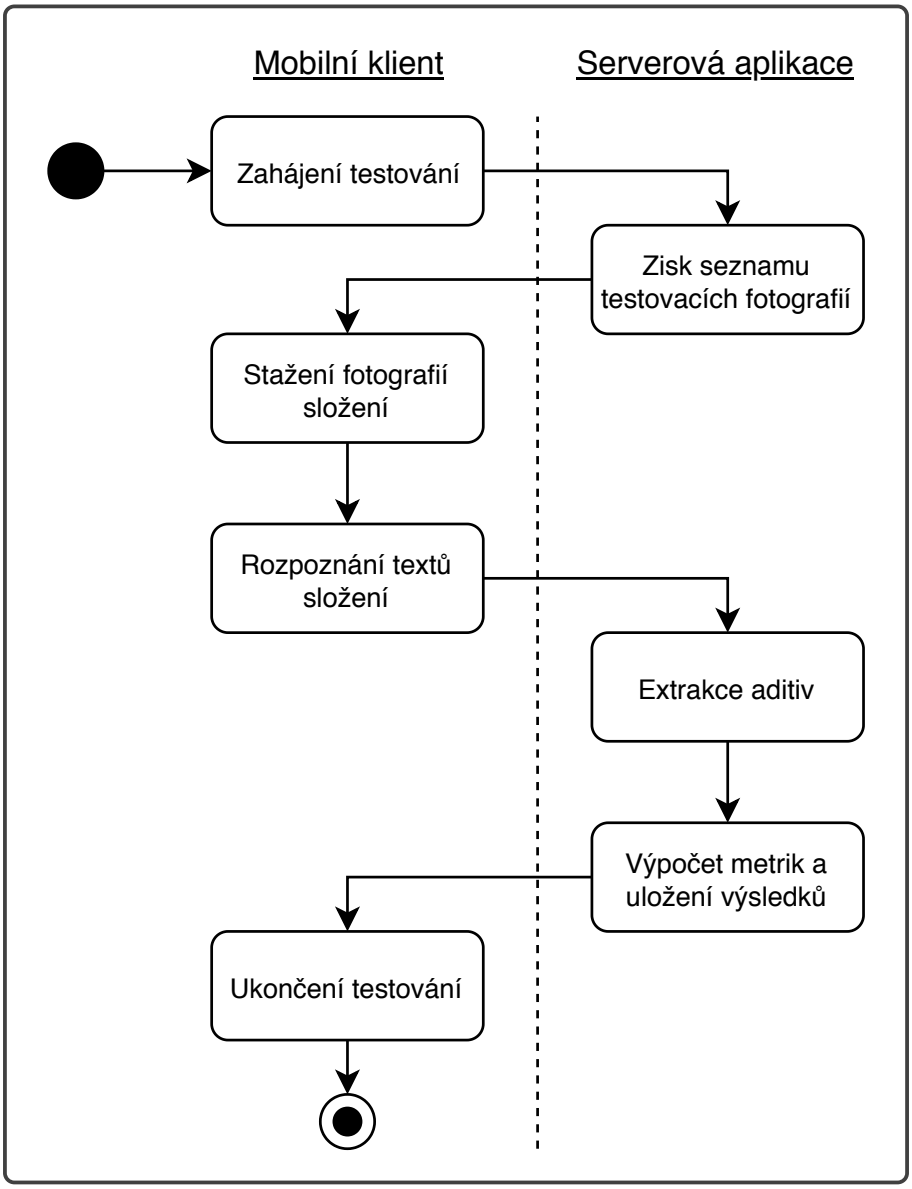
1. **Zahájení testování**: Testování bude zahájeno na mobilním klientu stiskem příslušného tlačítka.
2. **Zisk seznamu fotografií**: Mobilní klient si od serverové aplikace vyžádá seznam testovacích fotografií.
3. **Rozpoznání textů složení**: Jednotlivé fotografie ze seznamu budou klientem staženy, předzpracovány a bude nad nimi provedeno rozpoznávání znaků. Rozpoznané texty následně odešle do serverové aplikace.
4. **Extrakce aditiv**: Serverová aplikace ze získaných textů složení extrahuje označení přídavných látek.
5. **Vyhodnocení úspěšnosti**: Serverová aplikace vyhodnotí úspěšnost implementace extrakce aditiv skrze výpočty hodnotících metrik, viz část 5.2.3.
6. **Uložení výsledků**: Výsledky testu budou vhodně reprezentovány a uloženy.

Zjednodušený diagram průběhu testovaní můžeme vidět na obrázku 5.5.

5.3 Uživatelské role - případy užití

V systému budou vystupovat tři typy uživatelů:

- **Nepřihlášený uživatel**: Představuje uživatele nepřihlášeného do systému.

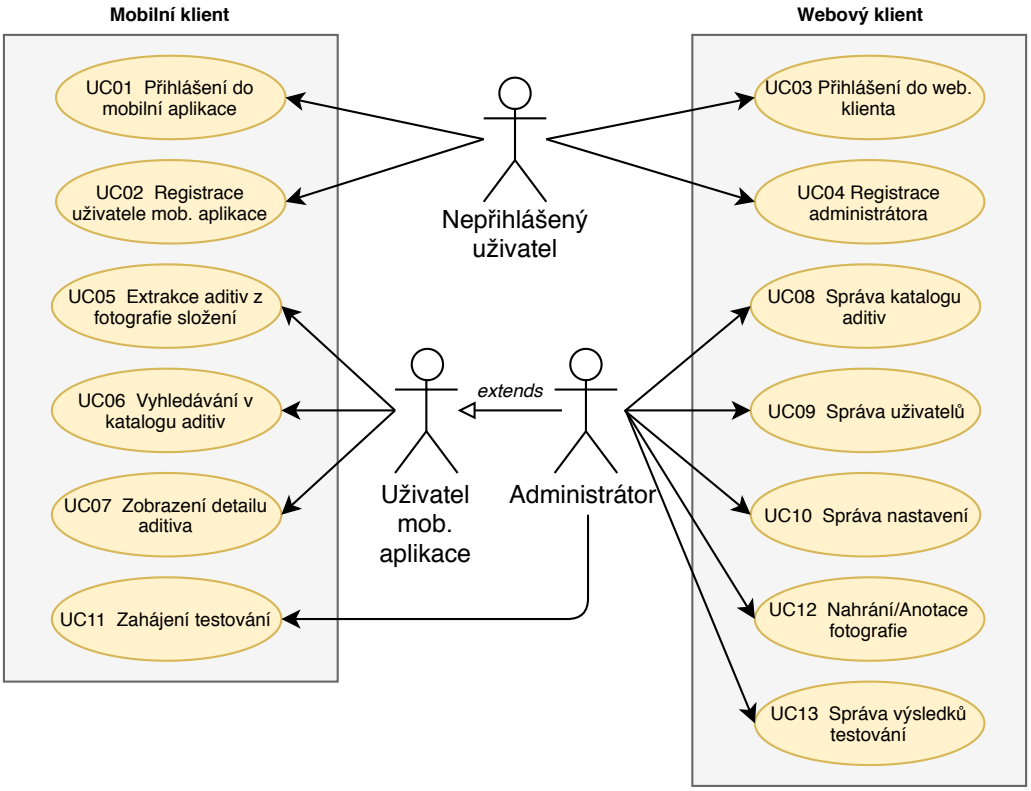


Obrázek 5.5: Diagram průběhu testování

- **Uživatel mobilního klienta**: Reprezentuje uživatele mobilního klienta. Jeho působnost v systému je omezena pouze na používání mobilní aplikace.
- **Administrátor**: Reprezentuje správce systému. Může využívat jak mobilní aplikaci, tak i webového klienta.

Uživatel mobilního klienta se do systému bude moci zaregistrovat prostřednictvím registračního formuláře v mobilní aplikaci, jeho účet bude aktivován okamžitě po provedení registrace. Registrování administrátora bude

umožněno pouze skrze formulář, jenž bude součástí webového klienta. Administrátorský účet bude bezprostředně po registraci neaktivní, tj. nebude moci být nijak využíván, dokud jej jiný administrátor explicitně neaktivuje. Stručný diagram případů užití můžeme vidět na obrázku 5.6.



Obrázek 5.6: Základní případy užití

Případy užití systému pro nepřihlášeného uživatele budou následující:

1. **UC01 Přihlášení do mobilní aplikace**: Nepřihlášený uživatel se může registrovat jako nový uživatel mobilní aplikace.
2. **UC02 Registrace uživatele mobilní aplikace**: Nepřihlášený uživatel se může přihlásit do mobilní aplikace.
3. **UC04 Registrace administrátora**: Nepřihlášený uživatel se může registrovat jako nový administrátor.
4. **UC03 Přihlášení do webového klienta**: Nepřihlášený administrátor se může přihlásit do webového klienta.

Případy užití systému pro uživatele mobilní aplikace budou následující:

1. **UC05 Extrakce aditiv z fotografie složení**: Uživatel mobilní aplikace může spustit proces extrakce přídavných látek z fotografie obalu potraviny.

2. **UC06 Vyhledávání v katalogu aditiv**: Uživatel mobilní aplikace může vyhledávat v katalogu přídavných látek.

3. **UC07 Zobrazení detailu aditiva**: Pro každé aditivum nalezené v katalogu nebo extrahované z fotografie obalu potraviny může uživatel zobrazit jeho detailní popis.

Administrátor může v systému provádět všechny akce, které má k dispozici uživatel mobilní aplikace, jeho možnosti jsou navíc rozšířeny o další případy užití týkající se webové aplikace:

1. **UC08 Správa katalogu aditiv**: Administrátor má umožněno spravovat katalog přídavných látek. Může přidávat nová aditiva a odstraňovat nebo modifikovat aditiva existující.

2. **UC09 Správa uživatelů**: Administrátor může registrovaného uživatele odstranit, zablokovat nebo aktivovat jeho účet. Rovněž může aktivovat účet jiného administrátora, který dosud nebyl aktivován.

3. **UC10 Správa nastavení**: Administrátor může spravovat parametry procesu extrakce aditiv.

4. **UC11 Zahájení testování**: Administrátor může zahájit testování úspěšnosti extrakce aditiv.

5. **UC12 Nahrání/Anotace fotografie složení**: Administrátor může nahrát testovací fotografii na server a následně pro ni provést anotaci.

6. **UC13 Správa výsledků testování**: Administrátor může zobrazit nebo odstranit výsledky testování úspěšnosti extrakce přídavných látek.

6 Architektura systému

V této kapitole bude popsán návrh architektury systému pro rozpoznávání přídavných látek z fotografií složení potravin. Rovněž zde budou uvedeny technologie a principy, které budou použité při jeho budoucím vývoji, viz části 6.1 až 6.3.

6.1 Serverová aplikace

Serverová aplikace bude implementována v programovacím jazyku Java 8. Architekturu a strukturu serverové aplikace podřídíme frameworku **Spring** [40]. Jedná se o populární framework pro vývoj podnikových aplikací, informačních systémů apod. Spring je postaven na principech *IoC*[1] a *Dependency Injection*[2], přebírá na sebe zodpovědnost za vytvoření a provázání závislostí mezi jednotlivými částmi systému, a výrazně tak ulehčuje a zpřehledňuje vývoj. Konkrétně použijeme jeho variantu **Spring Boot**, která umožňuje jednoduchý vývoj standalone aplikací[3].

Architektura aplikace bude rozdělena do tří dílčích vrstev. Jmenovitě se jedná o vrstvu řídící, která je též označována jako webová, viz část 6.1.3, a vrstvu servisní a vrstvu doménovou, viz části 6.1.2 a 6.1.1. Jejich uspořádání můžeme vidět na obrázku 6.1. Logiku jednotlivých vrstev můžeme dále pomyslně rozdělit na část veřejnou (angl. *public*) a část privátní (angl. *private*), viz obr 6.1. V části veřejné se nachází logika, která je dostupná klientským aplikacím, zatímco logika privátní části je dostupná pouze aplikaci serverové.

6.1.1 Doménová vrstva

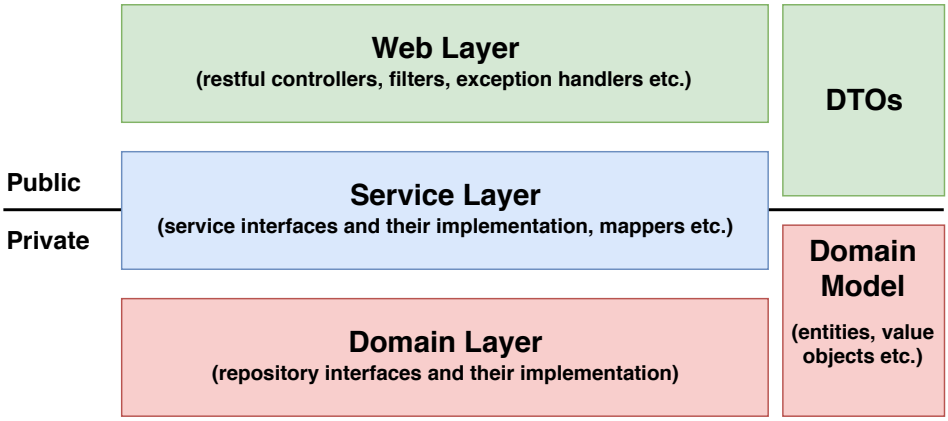
Doménová vrstva bude obsahovat několik doménových entit[4] sloužících pro perzistenci dat zpracovávaných systémem. K definování entit použijeme třídy

[1] Jedná se o zkratku anglického *Inversion of Control*. Označujeme tak způsob vytváření programového kódu, kdy jeho potřebné závislosti jsou nastavovány z vnějšího prostředí.

[2] *Dependency Injection* je konkrétní technika IoC.

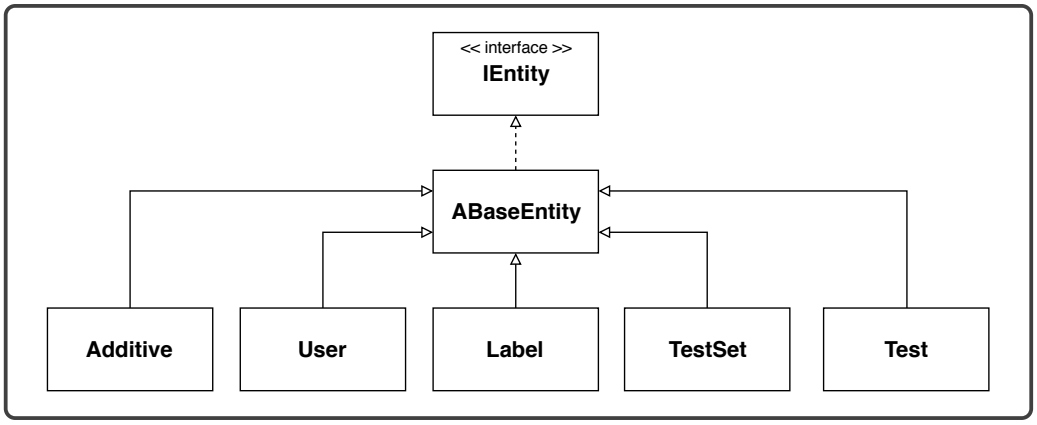
[3] Jedná se o webové aplikace, jejichž součástí je vestavěný webový server, tj. jsou plně funkční bez nutnosti jejich nasazení na externí webový server, jako jsou např. Tomcat nebo Jetty.

[4] Entita je objekt, který reprezentuje data v databázi. Typicky entitní třída reprezentuje tabulku v relační databázi a každá instance této třídy koresponduje jedné řádce tabulky.



Obrázek 6.1: Vrstvy serverové aplikace

jazyka Java rozšířené o anotace ze standardu Java Persistence API. Tyto anotace slouží zejména ke konfiguraci objektově relačního mapování (ORM). Umožňují nám tedy relativně snadno definovat konverzi mezi entitními třídami jazyka Java a tabulkami v relační databázi. Pro samotné objektově relační mapování použijeme framework jazyka Java **Hibernate ORM**. Základní entitní třídy a jejich hierarchii můžeme vidět na obrázku 6.2.



Obrázek 6.2: Diagram entitních tříd

V systému budou figurovat následující entitní třídy:

- **ABaseEntity**: Jedná se o společného abstraktního předka všech ostatních doménových entit. Ostatní entitní třídy jej budou rozšiřovat a dále specializovat. Jediným atributem této třídy je automaticky generovaný celočíselný primární klíč, který bude využit pro jednoznačnou identifikaci záznamu entitní třídy v relační tabulce v databázi.
- **Additive**: Tato entitní třída reprezentuje potravinářskou přídavnou

látku uloženou v systému. V jejích atributech budou uloženy detailní informace o dané přídavné látce. Jmenovitě bude obsahovat její E kód, název, skóre škodlivosti, charakteristiky, výrobní postup, vedlejší účinky apod. Všechny instance třídy Additive tvoří v aplikaci katalog přídavných látek.

- **User**: Entita tohoto typu reprezentuje uživatele registrovaného do systému. V atributech entitní třídy nalezneme jeho uživatelské jméno, heslo v zašifrované podobě, emailovou adresu, datum registrace, informaci o jeho roli v systému atd.

- **Label**: Entita Label reprezentuje fotografii složení potraviny. Obsahuje cestu k umístění dané fotografie na pevném disku serveru a její anotované vlastnosti.

- **Test**: Tato entita reprezentuje výsledek testování úspěšnosti extrakce aditiv nad jednou konkretní testovací fotografií složení.

- **TestSet**: Entita TestSet reprezentuje kompletní výsledek testování úspěšnosti extrakce aditiv nad všemi testovacími fotografiemi. Jedná se tedy o seskupení dílčích výsledků reprezentovaných třídou Test, doplněné o hodnoty celkových metrik úspěšnosti.

Součástí doménové vrstvy budou rovněž objekty DAO (*Data Access Objects*), jenž zapouzdřují a abstrahují přístup k databázi. Každé entitní třídě bude náležet jeden DAO objekt, který nad ní umožní provádět CRUD[5] operace.

6.1.2 Servisní vrstva

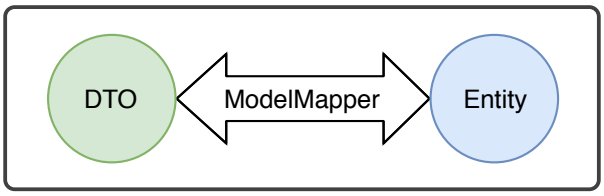
V servisní vrstvě se soustředí veškerá komplexnější výpočetní logika a operace s entitními třídami, včetně jejich mapování na DTO[6] objekty a naopak. Hlavní zodpovědností servisní vrstvy bude přenos dat mezi vrstvou řídící a doménovou.

Mapování obstará knihovna **ModelMapper** (viz obr 6.3). Výhodou mapování entit na DTO objekty je, že mezi komponentami systému nemusíme přenášet celý obsah entitní třídy, ale pouze jeho podmnožinu definovanou atributy DTO objektu. Rovněž DTO objekty můžeme přesunout do samostatného artefaktu, jenž nebude záviset na obsáhlých rámcích typu Hibernate nebo Spring, čímž znatelně snížíme paměťové nároky a čas překladu

[5]CRUD (Create, Read, Update, Delete) je zkratka používaná v programování. Shrnuje čtyři základní operace se záznamem v databázovém úložišti.

[6]Jedná o zkratku z anglického *Device Transfer Object*. Označujeme tak objekt, který zapouzdřuje data přenášená mezi subkomponentami systému.

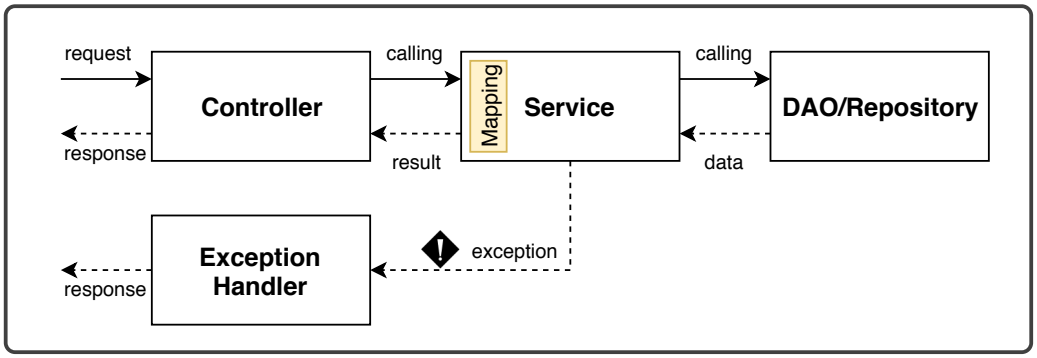
klientských programů (viz část 6.1.5). Jméno DTO objektu se bude skládat ze jména entitní třídy a přípony DTO, např. AdditiveDTO a UserDTO.



Obrázek 6.3: Mapování entity a DTO

6.1.3 Řídící vrstva

V řídící vrstvě nalezneme kontrolery, které budou přijímat a obsluhovat HTTP požadavky odeslané webovým nebo mobilním klientem. Na úrovni řídící vrstvy rovněž proběhne validace uživatelských vstupů a ošetření potenciálních chybových stavů, včetně výjimek jazyka Java. Zjednodušené schéma zpracování požadavku zobrazuje obrázek 6.4.



Obrázek 6.4: Zpracování požadavku

Datová výměna mezi klientskými komponentami a řídící vrstvou serverové aplikace bude probíhat ve formátu JSON. Serializaci a deserializaci přenášených požadavků, odpovědí a DTO objektů obstará knihovna **Jackson**.

6.1.4 Zabezpečení přístupu

Komunikace mezi klientskými aplikacemi a serverovým rozhraním bude zabezpečena otevřeným standardem **OAuth 2.0** [6]. Konkrétně bude použita implementace OAuth 2.0 dostupná ve frameworku **Spring Security**. Standard OAuth definuje 4 role:

- **Uživatel** (*Resource owner*): Uživatel vystupuje v roli vlastníka zdrojů. Autorizuje klientskou aplikaci pro přístup k svému uživatelskému účtu, datům a svým zdrojům obecně.

- **Klient** (*Client*): Klient je aplikace, která chce získat přístup k uživatelovu účtu a jemu dostupným zdrojům. Předtím, než tak učiní, musí být její požadavky autorizovány uživatelem a autorizace následně musí být validována autorizačním serverem.

- **Autorizační server** (*Authorization Server*): Autorizační server spravuje uživatelův účet, ověřuje jeho identitu a následně vydává bezpečnostní token klientským aplikacím. Pod pojmem token rozumíme alfanumerický řetězec o délce 64 znaků, kterým klientská aplikace proklamuje serveru se zdroji, že ji uživatel autorizoval k manipulaci se svými daty apod.

- **Server se zdroji** (*Resource Server*): Jedná se o server, který spravuje uživatelova data a poskytuje mu služby nebo další zdroje.

Pro zjednodušení architektury systému bude funkcionalita autorizačního serveru přímo integrována do serverové aplikace. Rozhraní serveru bude tedy obsluhovat jak požadavky na autorizaci, tak požadavky pro přístup k datům (viz obr. 6.5).

Autorizace klientské aplikace pro přístup ke službám a datům poskytovaným serverovou aplikací se bude skládat z následujících kroků:

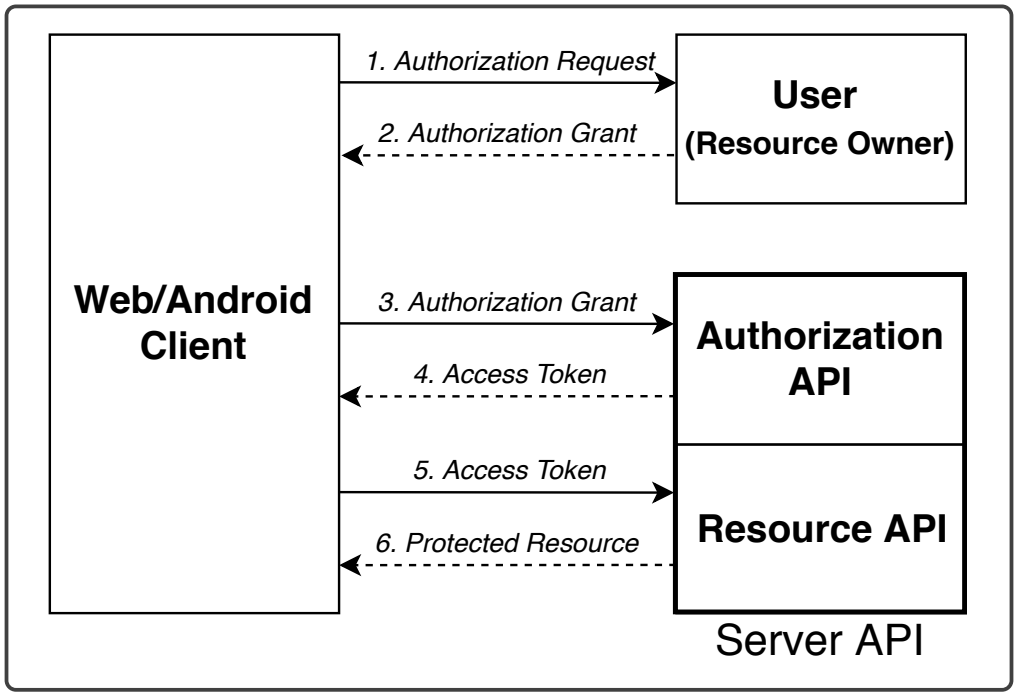
1. Klientská aplikace vyzve uživatele k tomu, aby se autentikoval zadáním uživatelského jména a hesla do přihlašovacího formuláře.

2. Uživatel vyplní a potvrdí přihlašovací formulář a udělí tak aplikaci souhlas s pokusem o autorizaci.

3. Zadané přihlašovací údaje budou odeslány do autorizačního rozhraní serverové aplikace.

4. Pokud budou zadané přihlašovací údaje validní, tak autorizační server vydá klientské aplikaci bezpečnostní klíč.

5. Při přístupu k chráněným zdrojům, které serverová aplikace uživateli poskytuje, se bude klient autorizovat platným bezpečnostním klíčem. Platnost bezpečnostního klíče bude omezena na 2 hodiny.



Obrázek 6.5: Průběh autorizace klienta

6.1.5 Maven artefakty

Při vývoji serverové aplikace bude použit **Maven**, což je nástroj pro správu, řízení a automatizaci překladu programů [24]. Maven umožňuje rozdělit rozsáhlý program na dílčí části, tzv. artefakty, a definovat mezi nimi závislosti. Artefakty dále můžeme agregovat a uspořádávat do hierarchické struktury. Pro konfiguraci artefaktů obecně platí, že artefakty, které jsou níže postavené v hierarchii, přebírají nastavení svých předků. Modulární struktura urychluje překlad programu, zpřehledňuje jeho strukturu a kladně se podepisuje na jeho znovupoužitelnosti.

Dalším pilířem nástroje Maven je používaní nejrůznějších pluginů, kterými dokážeme modifikovat překlad artefaktů. Mohou být spuštěny manuálně, případně lze jejich spuštění navázat na určitou fázi překladu.

Maven strukturu serverové aplikace můžeme vidět na obrázku 6.6, ze kterého je patrné její rozdělení do tří artefaktů:

- **server-parent**: Jedná se o rodičovský artefakt, který agreguje a konfiguruje zbylé dva artefakty.
- **server-application**: V tomto artefaktu bude obsažena veškerá aplikační logika serveru, tj. najdeme zde implementaci kontrolerů, servisních objektů, repositářů, entitních tříd apod. Artefakt bude záviset na

frameworcích Spring a Hibernate ORM.

- **server-api**: Tento artefakt bude obsahovat implementaci všech DTO objektů, serverových požadavků a odpovědí, definice chybových stavů apod. Centralizujeme v něm tedy definice dat sdílených a používaných napříč všemi komponentami systému. Artefakt bude mít minimální množství závislostí na externí knihovny a frameworky.

Při překladu artefaktu `server-api` dojde k automatickému spuštění pluginu, který jeho obsah přegeneruje z jazyka Java 8 do jazyka Typescript. Výstupem činnosti pluginu bude modul nazvaný `server-api.ts` obsahující definice sdílených objektů v jazyce Typescript.

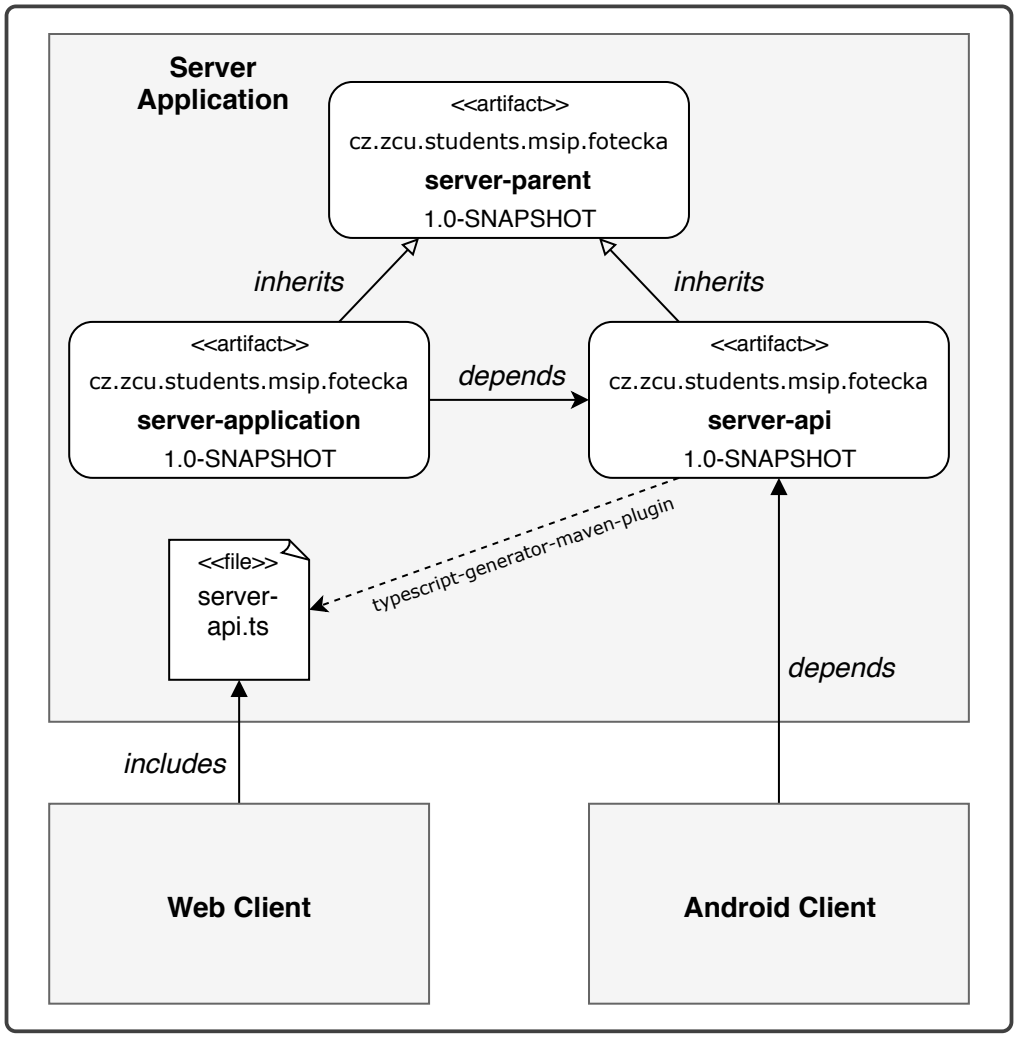
Artefakt `server-api` bude používán při vývoji mobilní aplikace. Webový klient bude záviset na vygenerovaném modulu jazyka Typescript. Tímto způsobem dosáhneme toho, že všechny komponenty systému získají přístup k definicím sdílených dat bez nutnosti duplikace programového kódu.

6.2 Mobilní klient

Mobilní klient bude implementován jako aplikace pro platformu Android. Při jejím vývoji bude použit programovací jazyk Java. Architektura aplikace bude postavena na moderních knihovnách **Retrofit2** a **RxAndroid** a jejich vzájemné integraci [23]. Zjednodušené schéma popisující její architekturu můžeme vidět na obrázku 6.7.

Mobilní aplikace nebude vybavena perzistentní vrstvou. Jejím primárním zdrojem dat bude REST API poskytované serverovou aplikací. Komunikaci s rozhraním obstará HTTP klient implementovaný v knihovně Retrofit2. Předností této knihovny je, že umožňuje asynchronní a typově bezpečnou datovou výměnu prostřednictvím HTTP protokolu. Komunikace bude realizována ve strukturovaném formátu JSON.

K odeslání HTTP požadavků, jejich konfiguraci a následnému zpracování odpovědí slouží v knihovně Retrofit2 tzv. služby (angl. *Retrofit Services*) [23]. K definici služby používáme interface jazyka Java a příslušné knihovní anotace. Jednoduchou službu pro vyhledání přídavných látek můžeme vidět v úryvku kódu 6.1. Využitím anotace stanovíme typ dotazovací metody (GET, POST, DELETE apod.) a URL koncového bodu serverového rozhraní, který provede obsluhu požadavku. Návratovou hodnotou metod rozhraní je typicky tzv. `Observable` objekt z knihovny **RxAndroid**, s jehož pomocí dokážeme na příchozí data asynchronně reagovat. Samotné HTTP volání pak není blokující a lze jej přímo použít i z prezenční vrstvy aplikace.

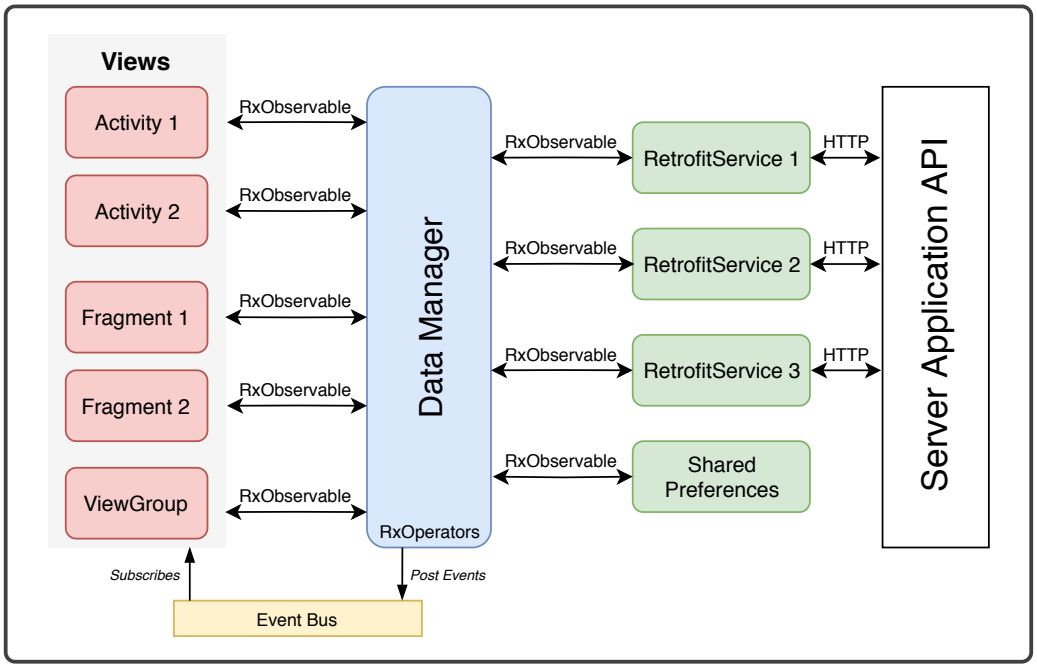


Obrázek 6.6: Artefakty serverové aplikace

```
public interface IAdditiveService {
    @GET("/additives/search/{text}")
    Observable<List<AdditiveDTO>> search(String text);
}
```

Kód 6.1: Příklad Retrofit služby

Spojením knihoven Retrofit a RxAndroid přinášíme do návrhu aplikace prvky tzv. reaktivního programování, které chápe data jako asynchronní proudy a události, které jsou pozorovatelné právě pomocí `Observable` objektů [23].



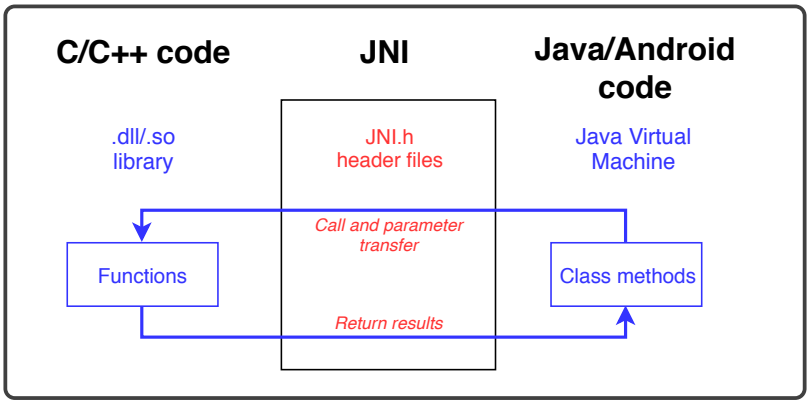
Obrázek 6.7: Architektura mobilního klienta

6.2.1 Předzpracování obrazových dat a OCR

Z části 5.1.1 je patrné, že nejpodstatnější úlohou mobilního klienta je předzpracování snímku složení potraviny a následné provedení optického rozpoznávání znaků v předzpracovaném snímku. Optické rozpoznávání znaků bude prováděno knihovnou Tesseract OCR. Hlavním důvodem pro výběr této konkrétní knihovny je její nekomerční charakter a z něj plynoucí bezplatnost. Rovněž dosahuje vysoké úspěšnosti rozpoznávání, viz část 4.3.1. Klasická verze Tesseract OCR je knihovnou jazyka C/C++ a nelze ji tak přímo nasadit v Android aplikaci, proto byla pro vývoj mobilního klienta zvolena její modifikace **tess-two**[7] vytvořená Robertem Theisem. Tato modifikace rozšiřuje původní knihovnu Tesseract OCR a umožňuje její použití v mobilních aplikacích platformy Android. Tato kompatibilita je docílena skrze sadu nástrojů **Android NDK** a rozhraní **JNI** (*Java Native Interface*).

JNI umožňuje propojení programového kódu jazyka Java s nativními knihovnami jazyka C/C++ (viz obr. 6.8) [36]. JNI využíváme zejména v případech, kdy je pro nás kritická rychlost vykonávání programu, neboť nám umožňuje vyčlenit náročné operace a komplikované výpočty do samostatné nativní knihovny, kterou následně optimalizujeme a kompilujeme pro cílovou platformu, a je tudíž mnohem efektivnější než standardní Java kód běžící na

[7] https://github.com/rmtheis/tess-two



Obrázek 6.8: Java Native Interface

virtuálním stroji. Vývoj nativních knihoven pro platformu Android a jejich následné propojení s Java kódem přes JNI můžeme realizovat pomocí sady nástrojů nesoucích označení Android NDK (*Native Development Kit*).

S ohledem na náročnost operací pro předzpracování obrazových dat bude logika předzpracování snímků složení rovněž implementována jako samostatná nativní knihovna. Při jejím vývoji budou použity otevřené C/C++ knihovny pro manipulaci s obrazem **OpenCV** a **Leptonica** [4]. Kompilaci knihovny a její propojení s Java kódem obstará Android NDK.

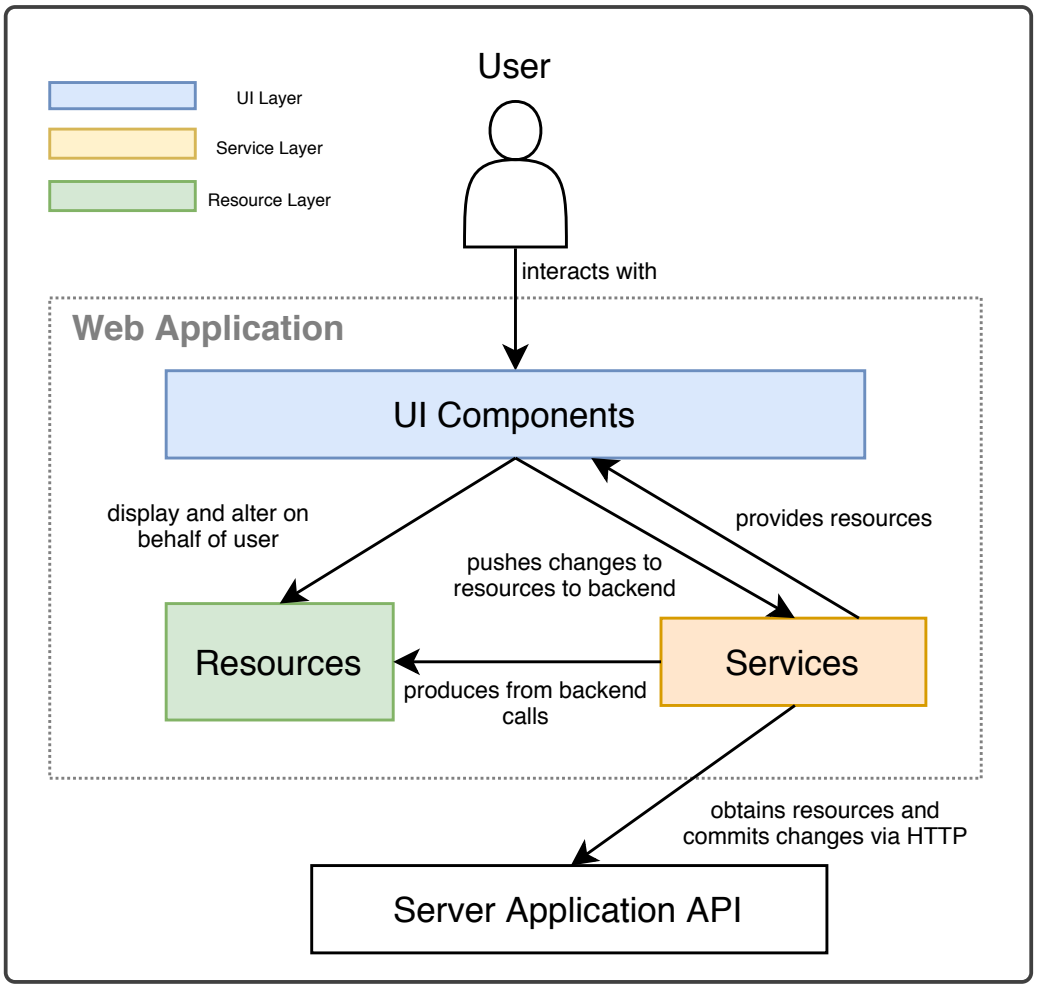
6.3 Webový klient

Webový klient bude realizován v JavaScriptovém aplikačním frameworku **Angular 6**. Jedná se o moderní a velmi populární framework pro vývoj webových aplikací. Je postaven na programovacím jazyku Typescript, který je nadmnožinou Javascriptu a rozšiřuje jej o typovou kontrolu a bezpečnou typovou konverzi. Angular 6 je založen na principech IoC a již v základu umožňuje Dependency Injection bez nutnosti použití dodatečných knihoven a frameworků. Jeho součástí je RxJS, knihovna pro asynchronní zpracování dat a neblokující HTTP volání. Jednoduché schéma architektury webového klienta můžeme vidět na obrázku 6.9. Ze schématu je patrné její rozdělení na tři dílčí části:

- **UI komponenty**: První část aplikace budou tvořit UI komponenty frameworku Angular, které budou zobrazovat data uživateli, reagovat na uživatelské vstupy apod. Jejich vzhled bude standardizován frameworkem **Angular Material**.
- **Zdroje**: Zdroje jsou data zobrazovaná uživateli. Uživatel je bude moci

prostřednictvím webového klienta mazat či modifikovat. Definice zdrojů budou součástí modulu `server-api.ts`, viz část 6.1.5.

- **Služby**: Vzhledem k tomu, že webový klient nemá vlastní perzistentní vrstvu. Budou veškerá data získávána a modifikována přes HTTP rozhraní serverové aplikace (viz obr 6.9). Logika samotného HTTP volání a následného asynchronního zpracování dat bude soustředěna do služeb.



Obrázek 6.9: Architektura webového klienta

7 Implementace systému

Systém pro extrakci přídavných látek z fotografií složení potravin byl implementován tak, jak bylo navrženo v částech 5 a 6. Logo systému můžeme vidět na obrázku 7.1. Cílem této kapitoly je seznámit čtenáře s detaily implementace procesu extrakce aditiv z fotografií složení potravin, počínaje předzpracováním obrazových dat a konče extrakcí aditiv z rozpoznaného textu, viz části 7.1 až 7.3.

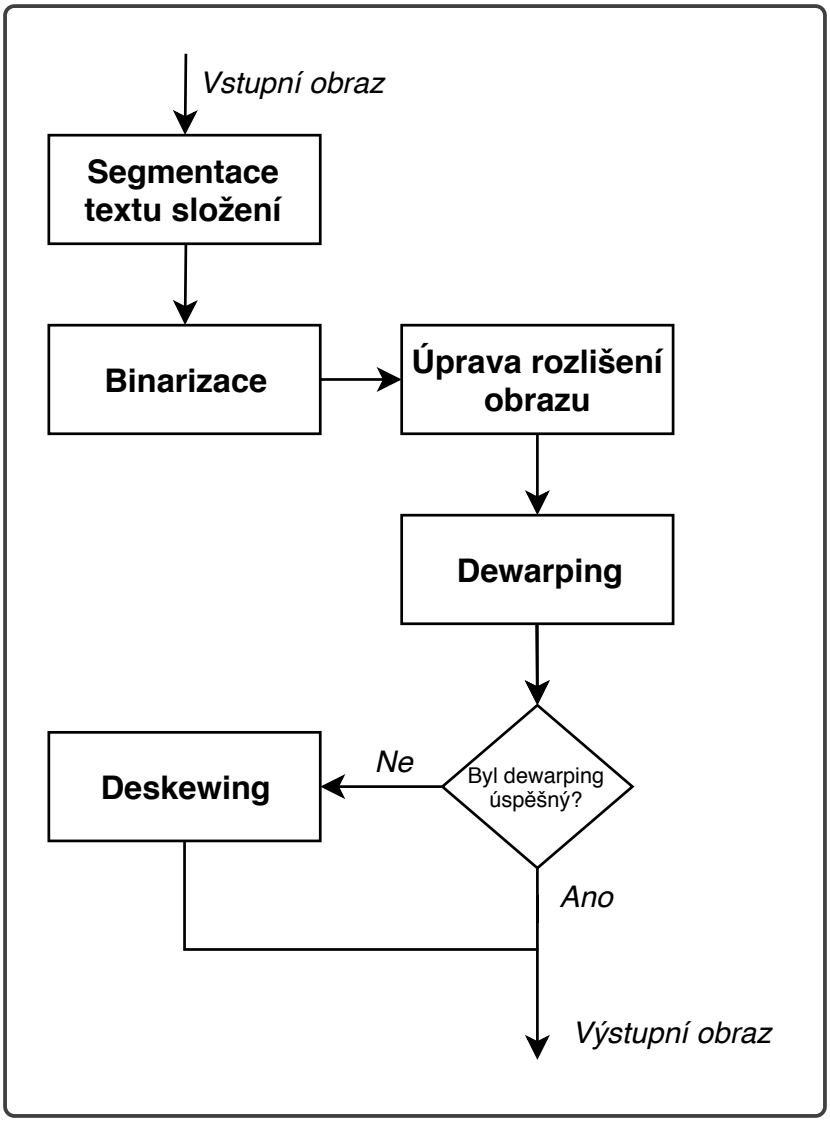
Obrázek 7.1: Logo systému

7.1 Předzpracování obrazových dat

Schématické znázornění implementovaného algoritmu pro předzpracování fotografie složení potraviny můžeme vidět na obrázku 7.2. Algoritmus je rozdělen do několika na sebe navazujících dílčích fází:

- **Segmentace textu složení**: Cílem této fáze je oříznutí vstupního snímku tak, aby obsahoval pouze českojazyčný text složení, viz část 7.1.1.
- **Binarizace**: Snímek českojazyčného textu složení je binarizován. Výstupem binarizace je obraz obsahující černý text (popředí) a bílé pozadí, viz část 7.1.2.
- **Úprava rozlišení obrazu** V této fázi dochází k úpravě rozlišení vstupního binárního obrazu, viz část 7.1.3.
- **Dewarping**: V této fázi dochází k narovnání křivých řádek textu, viz část 7.1.4.

- **Deskewing**: Pokud se nepodařil dewarping fotografie, tak dojde alespoň k nápravě jejího zešikmení, viz část 7.1.5.



Obrázek 7.2: Schéma předzpracování

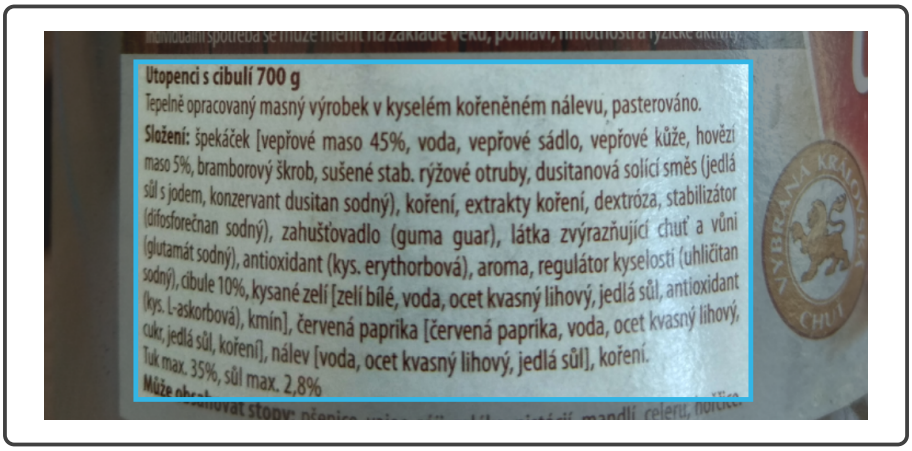
Během vývoje systému byly používány knihovny OpenCV a Leptonica, viz část 6.2.1. Konečná implementace však používá pouze knihovnu Leptonica ve verzi 1.74. Knihovna Leptonica má velmi dobrou dokumentaci [4], aktivní uživatelskou základnu a poskytuje velké množství funkcionalit specializované přímo pro potřeby OCR[1]. Ukázky kódu v částech 7.1.2 až 7.1.5 slouží zejména pro identifikaci použitých funkcí v knihovně Leptonica.

[1]Funkcionalita knihovny Leptonica je demonstrována na příkladech, jenž můžeme nalézt v *http://tpgit.github.io/UnOfficialLeptDocs/leptonica/prog-dir.html*

7.1.1 Segmentace textu složení

Cílem této fáze je oříznout vstupní obraz tak, aby výsledný obraz obsahoval pouze českojazyčný text složení potraviny. Ořezem fotografie snížíme její rozměry a tím znatelně urychlíme proces jejího dalšího předzpracování i následného rozpoznávání textu.

Výběr oblasti výskytu českojazyčeného textu složení ve fotografii je plně v režii uživatele mobilní aplikace. Uživateli je mobilní aplikací zobrazeno grafické rozhraní s fotografií složení potraviny, v němž uživatel označí českojazyčný text složení pomocí obdélníkové oblasti, viz obr. 7.3.



Obrázek 7.3: Výběr oblasti českojazyčného textu složení

7.1.2 Binarizace

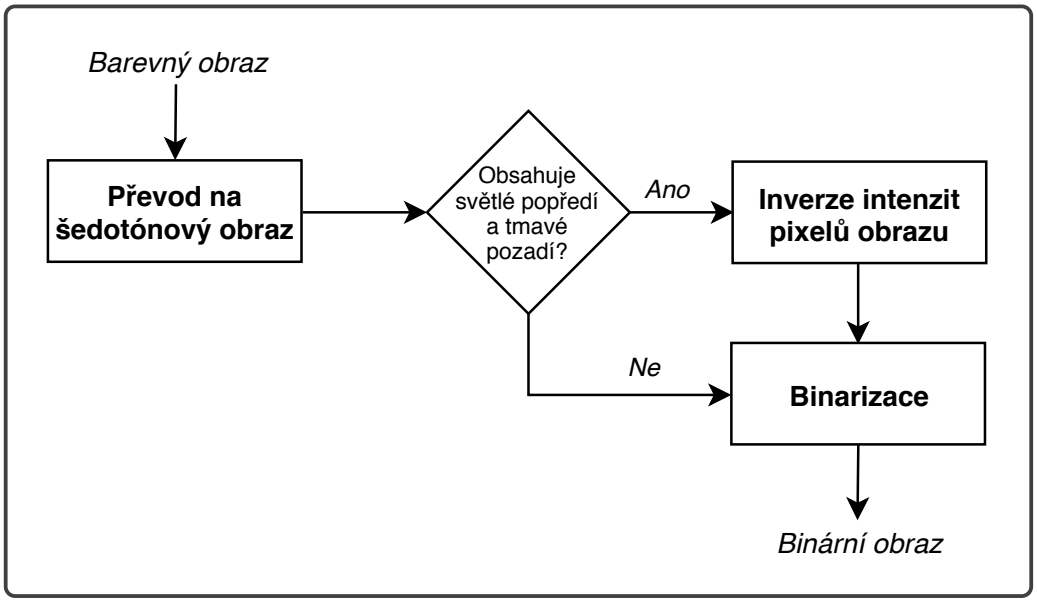
Schéma procesu binarizace je zobrazeno na obrázku 7.4. Barevný obraz je nejprve transformován na obraz šedotónový pomocí funkce z knihovny Leptonica, viz kód 7.1. Dále je provedena kontrola, zda šedotónový obraz obsahuje světlý text na tmavém pozadí nebo tmavý text na pozadí světlém. V posledním kroku je provedena samotná binarizace.

```
Pix* pix_grey = pixConvertTo8(pix_color, FALSE);
```

Kód 7.1: Převod barevného obrazu na šedotónový

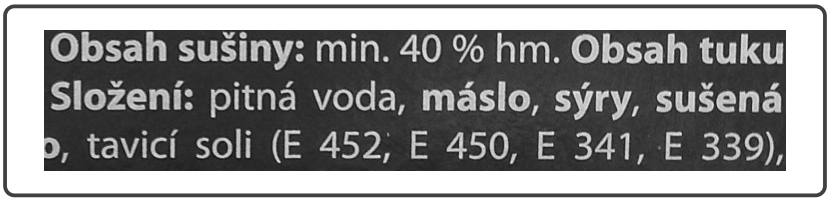
Detekce intenzity textu a pozadí

Většina implementací binarizačních algoritmů je postavena na předpokladu, že vstupní šedotónový obraz obsahuje tmavý text a světlé pozadí. Tento



Obrázek 7.4: Schéma procesu binarizace

předpoklad nemusí být pro fotografie textů složení potravin vždy splněn, viz obr. 7.5.

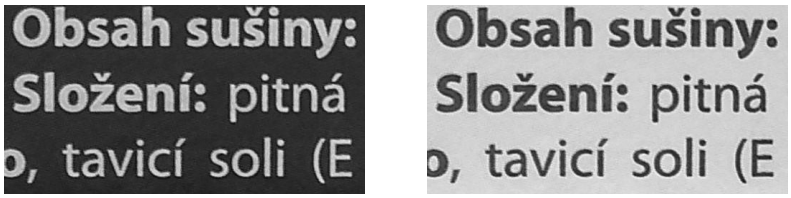


Obrázek 7.5: Světlý text složení potraviny

V rámci diplomové práce byla implementována metoda, jež dokáže detekovat fotografie se světlým textem umístěným na tmavém pozadí. Je-li taková fotografie detekována, tak je provedena inverze intenzit jejích obrazových bodů, viz obr. 7.6. Vstupem metody je tedy libovolný obraz textu složení potraviny. Jejím výstupem je obraz složení potraviny obsahující tmavý text a světlé pozadí.

Princip metody je relativně jednoduchý. Jeho cílem je ve fotografii složení určit oblast textu O_t a oblast pozadí O_p. Pokud platí, že průměrná intenzita pixelů v O_t je větší než průměrná intenzita pixelů v O_p, tak musí být provedena inverze pixelů obrazu složení.

Šedotónový obraz složení vyhladíme nejprve jednoduchým konvolučním filtrem, jenž je implementován v knihovně Leptonica, viz kód 7.2. Na vyhlazený obraz aplikujeme Sobelův filtr pro detekci hran (viz kód 7.3). Dete-



(a) Původní (b) Invertovaný

Obrázek 7.6: Inverze intenzit pixelů

kované hrany propojíme pomocí operace morfologického uzavření, viz část 3.2.3. Takto získáme binární masku, jež přibližně odpovídá oblasti textu O_t v původním obrazu. Masku pozadí získáme tak, že provedeme inverzi intenzit obrazových bodů masky textu, viz obr. 7.7.

```
Pix* pix_smooth = pixBlockconvGray(pix, NULL, 5, 5);
```

Kód 7.2: Vyhlazení obrazu

```
Pix* pix_edges = pixSobelEdgeFilter(pix, L_ALL_EDGES);
```

Kód 7.3: Detekce hran

```
pixSetMasked(pix, pix_foreground_mask, 255);
```

Kód 7.4: Aplikace masky popředí

(a) Popředí (b) Pozadí

Obrázek 7.7: Masky texu a pozadí

Masky aplikujeme nad původním obrazem složení (viz kód 7.4) a získáme tak přibližné obrazy oblastí textu O_t a pozadí O_p (viz obr. 7.8). Ty následně použijeme pro výpočet průměrné intenzity textu a průměrné intenzity pozadí. Jejich porovnáním zjistíme, zda má být původní obraz invertován, viz kód 7.5.

(a) Text (b) Pozadí

Obrázek 7.8: Oblasti textu a pozadí

```
pixInvert(pix, pix_inverted);
```

Kód 7.5: Inverze intenzit pixelů obrazu

Binarizační algoritmus

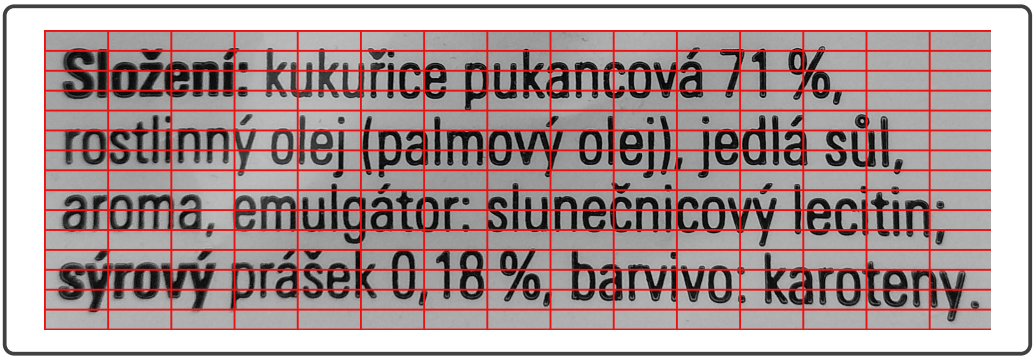
V rámci diplomové práce byla vyzkoušena řada binarizačních algoritmů, včetně všech algoritmů popsaných v části 3.1. Globální prahovací metody se dle očekávání prokázaly pro úlohu binarizace obrazu složení potraviny jako naprosto nedostatečné. Obrazy textů složení jsou ve většině případů nerovnoměrně osvětleny nebo poškozeny světelnými odlesky, a výstupem globálních metod tak byl často binární obraz obsahující defekty, jež čitelnosti výstupního textu bránily nebo ji přímo znemožňovaly.

Adaptivní Sauvulova a Niblackova metoda, viz části 3.1.3 a 3.1.2, dosahovaly při binarizaci textu složení obecně velmi dobrých výsledků, kvalita jejich výstupu však nebyla konzistentní pro všechny testovací fotografie. Tento fakt připisuji jejich obtížné parametrizaci pro danou úlohu, protože fotografie textů složení se vzájemně velmi odlišují. V závislosti na vzdálenosti fotoaparátu od foceného textu složení, použitém fontu, ohniskové vzdálenosti a rozlišení fotoaparátu mobilního telefonu se velikost vyfotografovaného textu pohybuje v řádu desítek až stovek pixelů. Tento fakt komplikoval volbu potřebných parametrů, jako je velikost plovoucího okénka apod. Manuální volba jedné hodnoty parametru zlepšila výsledek binarizace pro jednu skupinu fotografií a pro skupinu druhou jej naopak zhoršila.

Pro úlohu binarizace textů složení potravin byla použita jednoduchá lokální prahovací metoda, implementoval Renard Wellnitz v open-source programu Text Fairy[2]. Metoda dosahuje při binarizaci textů složení dobrých a konzistentních výsledků. Princip metody je relativně jednoduchý. Vstupní šedotónový obraz je nejprve rozdělen do několika navzájem se nepřekrý-

[2] *https://github.com/renard314/textfairy*

vajících podobrazů. Pro binarizaci obrazů složení je použito rozdělení na 15×15 podobrazů, viz obr 7.9. V každém z nich je následně stanovena hodnota prahu, která je poté použita při prahování pixelů tohoto podobrazu. Stanovení hodnoty prahu podobrazu je velmi podobné myšlence, na které je postaven i výpočet prahu Niblackovy metody, viz část 3.1.2.



Obrázek 7.9: Rozdělení obrazu složení

V podobrazu je nejprve vypočten průměr m a rozptyl σ^2 intenzit obrazových bodů, viz vztah 7.1:

$$m = \frac{1}{wh}\sum_{i=1}^{w}\sum_{j=1}^{h} B_{ij} \qquad \sigma^2 = \frac{1}{wh}\sum_{i=1}^{w}\sum_{j=1}^{h}(B_{ij} - m)^2 \tag{7.1}$$

kde w a h je šířka a výška podobrazu v pixelech a B_{ij} je intenzita obrazového bodu. Práh θ je následně určen vztahem 7.2:

$$\theta = \begin{cases} m & \text{když } \frac{m}{2} < \sigma^2 \\ m - 3\sqrt{\sigma^2} & \text{když } \frac{m}{2} \geq \sigma^2 \end{cases} \tag{7.2}$$

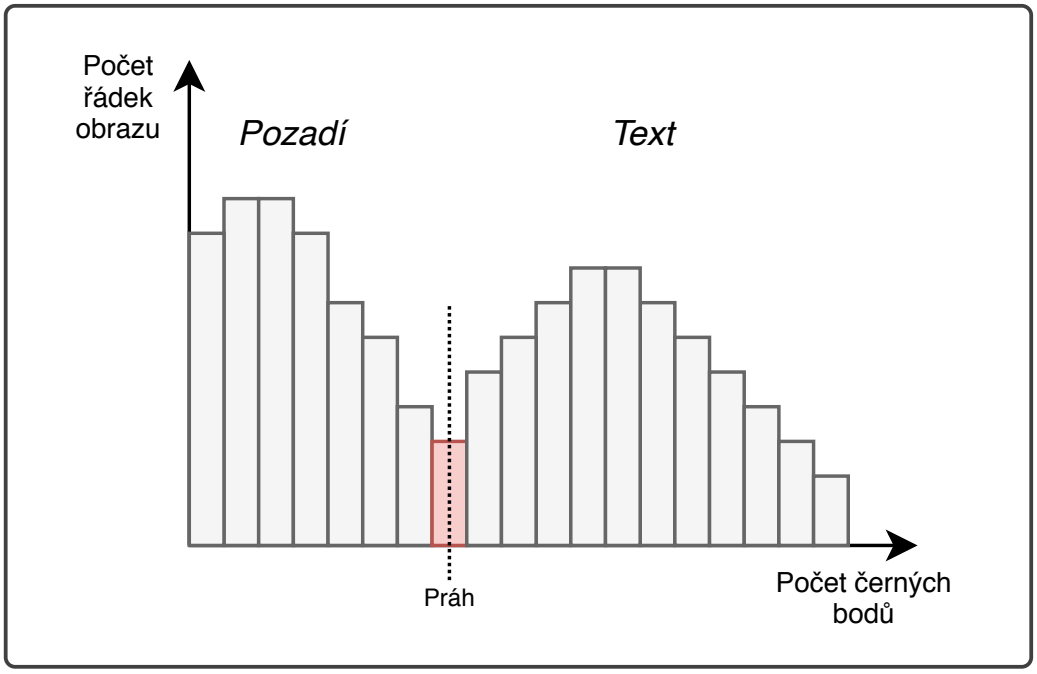
Velký rozptyl znamená dobrou separaci intenzit textu a pozadí a za hodnotu prahu θ je zvolen průměr m.

7.1.3 Úprava rozlišení obrazu

Vzhledem k tomu, že se velikost vyfotografovaného textu složení může pohybovat v řádu desítek až stovek pixelů, viz část 7.1.2, je potřeba provést normalizaci velikosti textu za účelem zvýšení úspěšnosti následného optického rozpoznávání znaků, viz část 4.2. Cílem této fáze je ve vstupním binárním obrazu složení zjistit průměrnou výšku obsažených řádek textu V_p. Pokud je průměrná výška řádek textu složení V_p příliš malá, tak je rozlišení vstupního obrazu škálováno směrem nahoru. Pokud je nalezená průměrná výška řádek

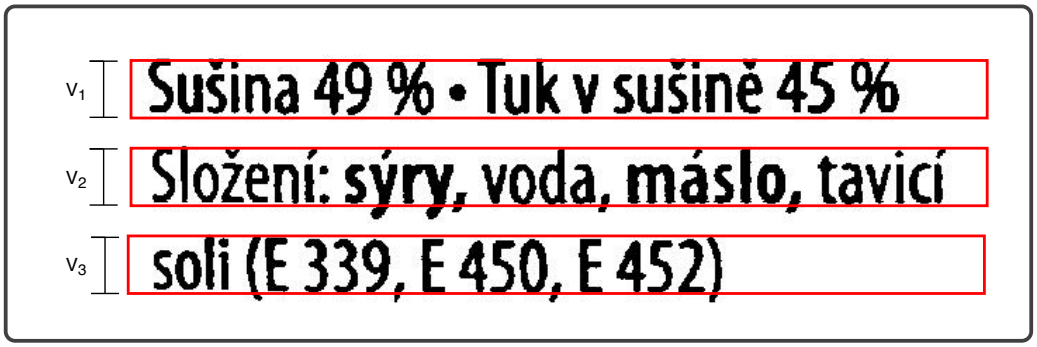
textu V_p příliš velká, tak je rozlišení vstupního obrazu škálováno směrem dolů. Při implementaci metody pro normalizaci velikosti textu složení jsem vycházel zejména z algoritmu, jež je součástí open-source programu Text Fairy, viz část 7.1.2. Průběh implementované metody lze rozdělit do tří fází.

Metoda prochází binární obraz textu po řádcích (jedná se o řádky obrazu nikoliv textu) a počítá počty černých bodů, které na nich leží. Ze získaných údajů je následně vytvořen histogram počtu řádek obrazu v závislosti na počtu černých pixelů, viz obr. 7.10. Z histogramu je patrné rozdělení řádek obrazu na dvě skupiny. První skupinu tvoří řádky obrazu, na kterých leží velký počet černých bodů. Jedná se zřejmě o řádky obrazu, jež **prochází textem**. Druhá část je tvořena řádkami obrazu, na kterých leží malý počet černých bodů. Tyto řádky zřejmě tvoří pozadí obrazu a textem neprochází. Na základě Otsuovy metody (viz část 3.1.1) určíme práh θ, který tyto dvě skupiny optimálně rozděluje.



Obrázek 7.10: Histogram počtu řádek obrazu v závislosti na počtu černých bodů

Práh θ je následně využit pro detekci počtu řádek textu a jejich výšek ve vstupním obrazu. Výška řádky textu je definována jako počet po sobě jdoucích řádek vstupního obrazu, které obsahují více černý bodů, než je hranice stanovená prahem θ, viz obr. 7.11. Na základě počtu detekovaných řádek a jejich výšek poté určíme průměrnou výšku jedné řádky textu V_p.



Obrázek 7.11: Detekce řádek textu a jejich výšek

V poslední fázi proběhne samotné škálování rozlišení binárního obrazu, viz vztah 7.3:

$$w_n = w \times \frac{50}{V_p} \qquad h_n = h \times \frac{50}{V_p} \tag{7.3}$$

kde w_n a h_h je nová šířka a výška obrazu v pixelech, w a h je původní výška a šířka obrazu a $\frac{50}{V_p}$ je škálovací koeficient. Výstupem škálování je tedy obraz, který obsahuje řádky textu o průměrné výšce právě 50 pixelů. Volba 50 pixelů není náhodná a odpovídá výšce znaku velikosti 12pt[3], jenž byl naskenován skenerem s rozlišením 300DPI (viz vztah 7.4), neboť rozlišení 300DPI je optimální pro OCR analýzu prováděnou nástrojem Tesseract OCR [27].

$$12 \times \frac{1}{72} \times 300 \approx 50\text{pixelů} \tag{7.4}$$

Implementace algoritmu je rozšířena o další mechanismy, které vznikly v rámci diplomové práce a mají za cíl zvýšit jeho robustnost. Konkrétní implementaci algoritmu nalezneme v souboru *libs/preprocessing/descale.cpp*.

7.1.4 Dewarping

Pro dewarping textu byla použita Bloombergova metoda [3], jejíž princip byl stručně vysvětlen v části 3.3. Implementace této metody je rovněž součástí knihovny Leptonica. Její použití je omezeno pouze na binární obraz textu, proto vstupní obraz složení musí být nejprve binarizován, viz část 7.1.2. Průběh metody je rozdělen do dvou fází. Nejprve je vytvořen matematický model zakřivení textu vstupní fotografie složení, viz kód 7.6. Matematický model zakřivení textu je úspěšně vytvořen, jsou-li splněny následující podmínky:

- Ve vstupním obrazu jsou detekovány, alespoň 4 „validní“ řádky textu.

[3] 1pt odpovídá 1/72 palce

- Detekovaná řádka je považována za „validní“, pokud její délka dosahuje alespoň 80% nejdelší detekované řádky.

Pokud byl model zakřivení textu úspěšně vytvořen, tak můžeme provést nápravu jeho zakřivení binárního obrazu, viz kód 7.7.

```
l_int32 vsuccess,
L_DEWARPA *dewa;
dewarpBuildPageModel(dew, NULL);
dewarpaModelStatus(dewa, 1, &vsuccess, NULL);
```

Kód 7.6: Vytvoření modelu zakřivení

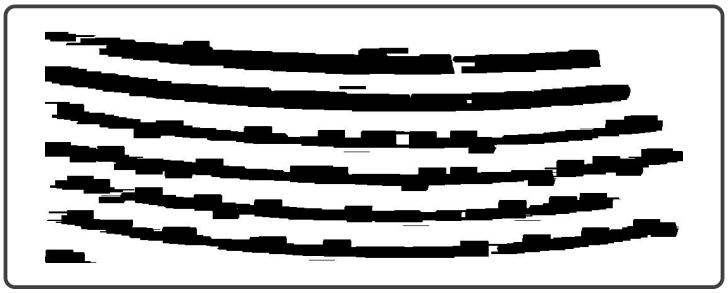
Implementace Bloombergovy metody v knihovně Leptonica dosahovala nad binárními obrazy textu složení obecně velmi špatný výsledků. Ve většině případů selhal proces detekce řádek, a nedošlo tak k vytvoření matematického modelu zakřivení. Text složení potraviny je často velmi kompaktní, řádkování textu je nepatrné a jednotlivé řádky jeho textu jsou těžce rozlišitelné. V rámci diplomové práce byl v reakci na tento problém navržen a implementován postup, jenž úspěšnost dewarpingu znatelně zvýšil.

```
// check model status
if (vsuccess) {
    dewarpaApplyDisparity(dewa, 1, pixb, pixd);
}
```

Kód 7.7: Aplikace modelu a oprava zakřivení

Řádky binárního obrazu složení jsou morfologickými operacemi transformovány na vzájemně dobře rozlišitelné linie, viz obr. 7.12. Jmenovitě je nad původním obrazem použita morfologická eroze strukturním elementem 1×15 (viz část 3.2.2). Ta zvětší vertikální mezery mezi řádky. Následně je použita morfologická dilatace strukturním elementem 40×1 (viz část 3.2.1), která zaplní mezery mezi slovy, jimiž jsou řádky tvořeny. Nad takto transformovaným obrazem je vytvořen model zakřivení, který je poté aplikován při nápravě zakřivení obrazu původního.

S ohledem na fakt, že ne všechny zakřivené texty složení obsahují 4 a více řádek textu, jež jsou Bloombergovou metodou vyžadovány pro úspěšné vytvoření modelu zakřivení, byly v rámci práce prováděny i experimenty s modifikací tohoto požadavku přímo v kódu knihovní funkce. Po snížení požadovaného počtu detekovaných řádek (na 2 a 3) se však metoda začala v



Obrázek 7.12: Transformace řádek textu

mnoha případech chovat nestabilně a deformovat vstupní obraz místo očekáváné nápravy jeho zakřivení. Požadavek 4 řádek byl tedy ponechán.

7.1.5 Deskewing

Deskewing binárního obrazu složení potraviny je opět postaven na funkcionalitě, jež poskytuje knihovna Leptonica. Nejprve je knihovní funkcí detekován úhel zešikmení fotografie, viz kód 7.8, a binární obraz je následně o daný úhel otočen, viz kód 7.9.

```
l_float32 angle = 0;
pixFindSkewSweep(pix, &angle, 2, 47., 1.);
```

Kód 7.8: Detekce úhlu zešikmení

```
pixRotate(pix, DEG_RAD_RATIO * angle);
```

Kód 7.9: Rotace obrazu

Deskewing je prováděň pouze nad obrazy, jejichž dewarping selhal, viz obr. 7.2. Implementace Bloombergovy metody v knihovně Leptonica (viz část 7.1.4) dokáže úspěšně napravovat i zešikmení obrazu. Pokud se tedy pro binární obraz textu složení podařilo vytvořit matematický model zakřivení, tak deskewing obrazu již postrádá významu.

7.2 Optické rozpoznávání znaků

Pro optické rozpoznávání znaků byla použita knihovna Tesseract OCR. Konkrétně byla zvolena její modifikace **tess-two** vytvořená Robertem Theisem[4],

[4]https://github.com/rmtheis/tess-two

která umožňuje použití Tesseractu i na mobilní platformě Android, viz část 6.2.1. Pro analýzu textu složení potraviny byla použita jednodušší varianta OCR analýzy, kterou nástroj Tesseract OCR nabízí, viz kód 7.10. Pro rozpoznání textu složení z fotografie je potřeba pouze:

1. vytvořit objekt `TessBaseAPI`
2. nastavit Tesseractu cestu k trénovacím datům a specifikovat používaný jazyk
3. nastavit obraz, jehož text chceme rozpoznat
4. nastavit mód segmentace, v rámci diplomové práce byly vyzkoušeny všechny segmentační módy, nejlepších výsledků pro danou úlohu dosahoval mód `PSM_SINGLE_BLOCK`
5. zavolat blokující metodu `getUTF8Text()`, která nám posléze vrátí rozpoznaný text

```
final String TESS_LANG = "ces";
TessBaseAPI mTessApi = new TessBaseAPI(this);
mTessApi.init(this.mDictionaryPath, TESS_LANG);
mTessApi.setImage(labelBitmap);
mTessApi.setPageSegMode(PSM_SINGLE_BLOCK);
String result = mTessApi.getUTF8Text();
```

Kód 7.10: Ukázka práce s knihovnou tess-two

Knihovna Tesseract OCR umožňuje provádět i pokročilejší analýzu textu, jejímž výstupem je kromě rozpoznaného textu i míra jistoty rozpoznaných slov (angl. *confidence*) a pravděpodobností mřížka, viz část 4.3.1. Výstup této pokročilejší analýzy lze následně použít při implementaci vlastních algoritmů pro detekci a korekci chyb rozpoznaného textu. Práce na toto téma byla úspěšně řešena Jiřím Martínkem na Západočeské univerzitě v Plzni [19].

Při implementaci procesu extrakce aditiv jsem se soustředil zejména na předzpracování obrazových dat takovým způsobem, aby byl výstup knihovny Tesseract OCR co možná nejspolehlivější. S ohledem na charakter vstupních obrazových dat, bylo jejich předzpracování nutné. Algoritmus pro extrakci aditiv z rozpoznaného textu, viz část 7.3, je postaven na podobnostním vyhledávání, a je tak částečně odolný vůči poruchám rozpoznaného textu. Výsledný proces tak dosahuje relativně dobrých výsledků i bez implementace

vlastního algoritmu pro detekci a korekci chyb, viz část 8. Využití detailního výstupu OCR analýzy a implementace vlastního korekčního algoritmu, podobně jako v [19], bude předmětem další práce.

7.3 Extrakce aditiv z textu složení

Vstupem algoritmu pro extrakci aditiv z textu složení je text složení potraviny rozpoznaný OCR knihovnou Tesseract a jeho výstupem je seznam přídavných látek nalezených v rozpoznaném textu. Algoritmus je schopen extrahovat z rozpoznaného textu přídavné látky na základě jejich názvu i E kódu.

Při návrhu a následné implementaci algoritmu musela být vzata v potaz omezení vyplývající z charakteru textu rozpoznaného OCR knihovnou. Rozpoznaný text je nezřídka zatížen chybami, a algoritmus proto musí počítat s následujícími defekty textu:

- **Absence oddělovačů**: Nelze se spolehnout na přítomnost oddělovačů (větných čárek, středníků apod.), které by oddělovaly jednotlivé složky složení.
- **Výpadek slov**: V rozpoznaném textu může dojít k výpadku slov. Jeli ve složení potraviny uvedeno aditivum s víceslovným názvem, tak algoritmus obecně nemůže vycházet z předpokladu, že všechna slova tvořící název tohoto aditiva budou přítomna i v rozpoznaném textu.
- **Poškození slov**: Rozpoznaná slova mohou být poškozena. OCR algoritmus může například rozpoznat slovo „citronové“ ačkoliv v textu složení je uvedeno slovo „citronová“.

Výsledný algoritmus byl navržen tak, aby byl vůči výše zmíněným poruchám rozpoznaného textu odolný. Jeho implementace je postavena na knihovně **Hibernate Search** ve verzi 5.6.0, viz část 7.3.1.

7.3.1 Hibernate Search

Hibernate Search je knihovna jazyka Java, jež integruje framework **Hibernate ORM** s knihovnou pro full-textové vyhledávání[5] a analýzu textu **Lu-**

[5]Fulltextové vyhledávání, (z ang. *full* – celý, plný a *text*) je speciální způsob vyhledávání informací v relačních databázích nebo v textových souborech, které jsou obvykle předem připraveny, tj. indexovány, aby bylo možno nalézt libovolné slovo (řetězec znaků) v nejkratším možném čase.

cene [34]. Jejím cílem je umožnit uživateli snadné a rychlé full-textové vyhledávání instancí entit uložených v databázi. Zjednodušeně řečeno, umožňuje nalézt instance entit, jejichž vybraný textový atribut obsahuje uživatelem zadaný řetězec.

Analýza textu atributů

Aby entitní třída jazyka Java byla full-textově vyhledatelná knihovnou, tak musí definovat primární klíč a obsahovat alespoň jeden atribut typu `String`, ve kterém bude uživatelem zadaný řetězec vyhledáván. Textový atribut, nad nímž chceme provádět full-textové vyhledávání musíme označit anotací `@Field`, viz kód 7.11.

```
@Entity
@Indexed
public class Additive {
  @Id
  @GeneratedValue
  private Integer id;

  @Field
  private String additiveCode;

  @Field
  private String name;

  // getters and setters
}
```

Kód 7.11: Hibernate Search anotace

Anotované atributy instancí entit jsou knihovou Hibernate Search analyzovány. Proces analýzy textu atributů obstarává tzv. analyzér, jenž se sestává z několika filtrů a jednoho tokenizéru:

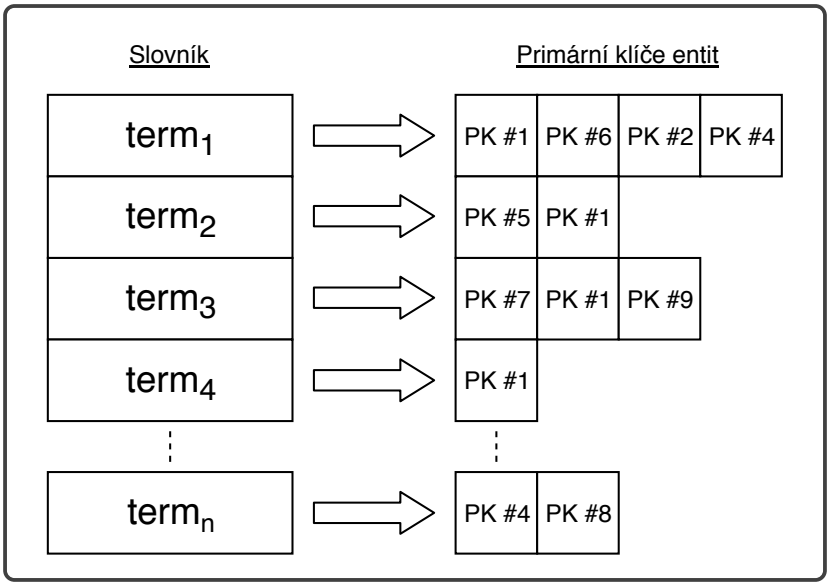
- **Char filter**: Přídává, odebírá nebo mění znaky ve vstupním textu. Analyzér může používat několik různých typů char filterů.
- **Tokenizér**: Je zodpovědný za rozdělení vstupního řetězce na jednotlivá slova. Tato „slova“ označuje Hibernate Search jako **tokeny**. Výstupem činnosti tokenizéru je seznam tokenů. Analyzér může obsahovat vždy pouze 1 typ tokenizéru.

- **Token filter**: Modifikuje nebo odebírá tokeny. Analyzátor může obsahovat několik různých typů filterů.

Výstupem analýzy je tedy různě modifikovaný seznam tokenů. Hibernate Search umožňuje pro každý textový atribut definovat vlastní analyzér o libovolné konfiguraci filterů a tokenizéru. K tomuto účelu slouží anotace `@AnalyzerDef`.

Indexace entit

Po analýze textu atributů instancí entitní třídy následuje proces indexace. Pro každý atribut entitní třídy označený anotací `@Field` vytvoří Hibernate Search tzv. full-textový index. Jeho zjednodušené schéma můžeme vidět na obrázku 7.13. Jde o invertovaný seznam, kde jsou klíčem unikátní tokeny (tzv. *termy*) získané analýzou textu v daném atributu entity napříč všemi jejími instancemi [34]. Ke každému termu je přiřazen seznam primárních klíčů, které identifikují instance entit, v jejichž atributu se daný term vyskytuje.



Obrázek 7.13: Hibernate Search index

Jedná se datovou strukturu, která znatelně urychluje full-textové vyhledávání. V případě změny textu indexovaného dokumentu nebo textového atributu instance entity musí být však index aktualizován. Výhodou použití knihovny Hibernate Search je, že na sebe přebírá zodpovědnost za aktualizaci indexu. S každou změnou hodnoty indexovaného atributu instance entitní

třídy jsou provedené změny asynchronně propagovány i do full-textového indexu bez nutnosti explicitního zásahu programátora.

Vyhledávání instancí entit

Indexované instance entity může můžeme následně full-textově vyhledávat pomocí vyhledávacích dotazů (angl. *query*) . K tomuto účelu slouží třída **Query** z knihovny Lucene. Query pro vyhledání instance entitní třídy **Additive** (viz kód 7.11), jenž má kódové označení „e412“, můžeme vidět v kódu 7.12. Při vytváření Query musíme specifikovat vyhledávaný text a atribut entity, ve kterém chceme zadaný text hledat.

```
org.apache.lucene.search.Query query = queryBuilder
  .keyword()
  .onField("additiveCode")
  .matching("e412")
  .createQuery();
```

Kód 7.12: Query pro vyhledání aditiva

Knihovna Hibernate Search poskytuje několik typů vyhledávacích dotazů. Základními typy dotazů jsou:

- **Keyword**: Slouží pro vyhledání instancí entity, jejichž vybraný atribut obsahuje zadané slovo (token).
- **Fuzzy**: Slouží pro vyhledání instancí entity, jejichž vybraný atribut obsahuje zadané slovo nebo slova jemu podobná. Podobnost slov je určována pomocí tzv. Levenshteinovi vzdálenosti[6].
- **Bool**: Umožňuje spojovat předchozí dva typy dotazů pomocí logických operátorů disjunkce (`OR`), konjunkce (`AND`) a negace (`NOT`).

Máme-li sestavený dotaz reprezentovaný třídou Query z knihovny Lucene, tak vyhledávané instance získáme z relační databáze voláním metody `query.getResultList()`. Pokud nás zajímá pouze počet instancí entity, které splňují požadavky definované vyhledávácím dotazem, tak můžeme použít metodu `query.getResultSize()`. Tato metoda pracuje pouze s vytvořeným indexem a je tedy znatelně rychlejší než metoda předchozí.

[6]Levenshteinova vzdálenost je vzdálenost dvou řetězců definovaná jako minimální počet operací vkládání, mazání a substituce takových, aby po jejich provedení byly zadané řetězce totožné.

7.3.2 Implementace algoritmu

V této části je vysvětlena implementace algoritmu pro extrakci přídavných látek z textu složení rozpoznaného OCR knihovnou.

Indexace entity Additive

Entitní třída **Additive**, jež v systému reprezentuje přídavnou látku (viz část 6.1.1), byla rozšířena o anotace z knihovny Hibernate Search, viz ukázka kódu 7.11. Její instance jsou indexovány na základě textu obsaženého v jejím jménu (atribut **name**) a textu v jejím E kódu (atribut **additiveCode**).

Pro analýzu textu obou atributů byl nakonfigurován vlastní analyzátor, který obsahuje:

- **StandardTokenizer**: Jedná se o implementaci standardního a jazykově nezávislého tokenizéru v knihovně Lucene. Rozděluje analyzovaný text do tokenů pomocí standardizovaného algoritmu s názvem *Unicode Text Segmentation Algorithm* [10].
- **LowerCaseFilter**: Jedná se o filtr, který nahrazuje velká písmena písmeny malými.
- **StopFilter**: Tento filtr odstraňuje tokeny, které odpovídají českým stop-slovům[7].

Analyzovaný text atributů je před procesem indexace rozdělen na jednotlivá slova (tokeny), jež obsahují pouze malá písmena. S ohledem na fakt, že velká část názvů přídavných látek vychází z chemického názvosloví, **nebyl použit filtr**, jenž by prováděl *stemming*[8] nebo *lematizaci*[9] tokenů.

Průběh algoritmu

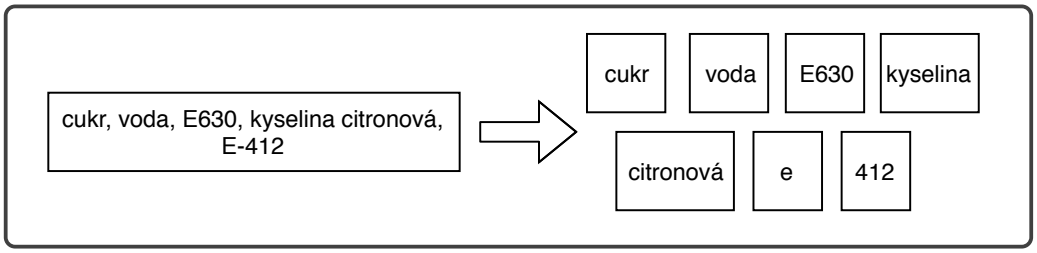
Vstupem algoritmu je text složení rozpoznaný OCR knihovnou. Tento text je nejprve předzpracován. Průběh předzpracování je stejný jako průběh analýzy textu atributů entity Additive. Výstupem předzpracování vstupního textu je tedy seznam tokenů, jejichž pořadí odpovídá pořadí jejich výskytu v rozpoznaném textu, viz obr. 7.14.

Ze získaného seznamu tokenů složení dokáže algoritmus extrahovat přídavné látky zapsané jak pomocí E kódů, tak i pomocí jejich celých názvů.

[7]Jde o slova, která nenesou sama o sobě žádný význam. V češtině jde především o předložky, spojky atd.

[8]Stemming je proces nalezení kmene slova (nepřesně a úžeji lze říci kořene). Algoritmus, který jej vykonává, se nazývá stemmer.

[9]Lematizace je operace, jenž vrací základní tvar slova (tzv. lemma).



Obrázek 7.14: Předzpracování rozpoznaného textu

Postup extrakce aditiva na základě E kódu je relativně jednoduchý a vychází z předpokladu, že jeho E kód je reprezentován jedním až dvěma tokeny:

- Jedním tokenem bude reprezentována E kód aditiva, jenž byl v textu složení potraviny zapsán bez mezer a oddělovačů např. „E420“ nebo „e630“.
- Dvěma tokeny bude reprezentován E kód aditiva, jenž byl v textu složení potraviny zapsán s využitím mezer nebo oddělovačů např. „E-420“ nebo „e 630“.

Algoritmus prochází seznam tokenů složení potraviny a kontroluje výskyt tokenu, jehož formát odpovídá validnímu E kódu, nebo výskyt tokenu „e“, který je následován takovým tokenem, po jehož připojení vznikne platný E kód. Validita nalezeného E kódu je ověřována regulárním výrazem. Pokud je při procházení seznamu tokenů detekován validní E kód, tak pomocí **Keyword Query** (viz kód 7.12) zjistíme, zda je daná přídavná látka přítomna v indexu a pokud ano, dojde k jejímu načtení z databáze a uložení do seznamu extrahovaných aditiv.

Postup při extrakci aditiva na základě jeho názvu je mnohem komplikovanější než postup předchozí. Názvy různých aditiv jsou reprezentovány obecně různým počtem tokenů a nelze tedy vycházet z předpokladu jejich fixního počtu. Čím delší je text aditiva, tím větší je rovněž pravděpodobnost, že se OCR systém dopustil chyb při jeho rozpoznávání. Tesseract OCR má při rozpoznávání českých slov tendenci zaměňovat jejich koncovky, např. místo slova „jablečná“ může rozpoznat slovo „jablečné“ apod.

Pokud algoritmus při průchodu seznamem tokenů narazí na token, který není platným E kódem nebo znakem „e“, tak se stále může jednat o token tvořící název aditiva. Pomocí **Fuzzy Query** je zjištěn počet přídavných látek, jež ve svém názvu obsahují slova podobná aktuálnímu tokenu. Míra podobnosti je omezena nastavením maximální Levenshteinovy vzdálenosti na hodnotu 1, viz kód 7.13. Vyhledávání na základě podobnosti je použito

jako nutná kompenzace možných defektů rozpoznaného textu složení. V závislosti na počtu výsledků je další postup následující:

1. Pokud nejsou v indexu nalezena žádná aditiva, tak algoritmus začne analyzovat další token v seznamu a celý postup se opakuje.

2. Pokud je v indexu nalezena pouze jedna přídavná látka, tak je následně načtena z databáze a přidána do seznamu nalezených aditiv. Algoritmus začne analyzovat další token v seznamu, celý postup se opakuje.

3. Je nalezena více než jedna přídavná látka, jež ve svém názvu obsahuje slovo podobné aktuálnímu tokenu. Například token „kyselina“, je obsažen v názvech několika přídavných látek a nelze tak z něj jednoznačně určit o jakou přídavnou látku se jedná. Algoritmus se následně pokusí přídavnou látku jednoznačně identifikovat postupnou analýzou dalších tokenů. Princip dalšího postupu je jednoduchý. Nalezlo-li Fuzzy Query pro token T_i více než jednu přídavnou látku, tak vytvoříme Fuzzy Query pro token T_{i+1} a obě Query spojíme do konjunktivního Bool Query. Ten hledá aditiva, která ve svém názvu obsahují slovo podobné tokenu T_i a zároveň obsahují i slovo podobné tokenu T_{i+1}. Proces přidávání tokenu do Bool Query je opakován (viz obr 7.15). Po každém přidání tokenů je zkontrolován počet nalezených aditiv. Proces přidávání tokenů je ukončen pokud:

   (i) Aktuální Bool Query nalezlo přesně jednu přídavnou látku a přidáním dalšího tokenu do Bool Query by klesl počet nalezených aditiv na 0. V tomto případě se aditivum podařilo jednoznačně určit, je načteno z databáze a uloženo do seznamu extrahovaných aditiv.

   (ii) Aktuální Bool Query již obsahuje nově přidávaný token. Pokud je počet výsledků aktuálního Bool Query roven 1, pak je aditivum jednoznačně určeno a je přidáno do seznamu extrahovaných aditiv, v opačném případě se aditivum nepodařilo jednoznačně určit.

   (iii) Počet aditiv, která konjunktivní Bool Query nalezlo, je rovný nule. Aditivum se tedy nepodařilo jednoznačně určit.

Výše popsané řešení mělo jednu zásadní slabinu, a to sice častý výskyt kolizních tokenů v rozpoznaném textu složení. Kolizní token je takové slovo, které může být obsaženo jak v názvu přídavné látky, tak i v názvu potravinové složky, která přídavnou látkou není. Jedná se například o slova jako

jsou „citronová“ nebo „mléčná“. Citronová může být kyselina (E330) ale i šťáva. Stejně tak může být mléčná kyselina (E270) ale i kultura.

Abychom předešli falešným detekcím přídavných látek v rozpoznaném textu, tak byl algoritmus rozšířen o seznam potenciálně kolizních slov. Pokud je v rozpoznaném textu složení detekována přídavná látka, tak proběhne kontrola, zda toto aditivum nebylo extrahováno pouze na základě kolizních tokenů a žádných jiných. V kladném případě je rozpoznané aditivum odstraněno ze seznamu nalezených přídavných látek, neboť nejsme schopni určit, zda se nejedná o falešně pozitivní výsledek.

Seznam kolizních slov byl vytvořen manuálně a je ho možné modifikovat prostřednictvím webového klienta. Jeho vytvoření bylo otázkou několika málo minut, kdy byly zkontrolovány názvy uložených aditiv a potenciální zdroje kolizí byly přidány na seznam. Jedná se o slova typu „mléčná“, „citronová“, „rostlinná“ apod.

$$\text{BoolQuery} = \text{Fuzzy}(\text{token}_i) \textbf{ AND } \text{Fuzzy}(\text{token}_{i+1}) \textbf{ AND } \ldots \textbf{ AND } \text{Fuzzy}(\text{token}_{i+n})$$

Obrázek 7.15: Konjunktivní spojování dotazů

```
org.apache.lucene.search.Query query = queryBuilder
                        .keyword()
                        .fuzzy()
                        .withEditDistanceUpTo(1)
                        .onField("name").matching(token)
                        .createQuery();
```

Kód 7.13: Vyhledávání na základě podobnosti

8 Ověření kvality extrakce aditiv

V této kapitole je zhodnocena dosažená úspěšnost extrakce přídavných látek z testovacích fotografií složení potravin. Metodika testování, včetně popisu použité kolekce testovacích fotografií, je předmětem části 8.1. Výsledky testování a jejich následnou diskuzi nalezneme v části 8.2.

8.1 Metodika testování

Ověření kvality extrakce aditiv bylo provedeno pomocí mechanismu, jenž byl navržen v části 5.2. Kvalita extrakce aditiv je vyhodnocena pomocí makro a mikro průměru přesnosti P, úplnosti R a F-míry, viz část 5.2.3. Konkrétní obsah testovací kolekce fotografií složení, na jejímž základě byla kvalita extrakce vyhodnocena, je uveden v části 8.1.1.

Testování bylo provedeno na mobilním telefonu „Redmi Note 4“ od značky Xiaomi. Jeho specifikaci nalezneme v tabulce 8.1. Všechny výsledky uvedené v části 8.2 se vztahují k tomuto zařízení. Po celou dobu testování bylo zařízení připojeno k nabíjecí stanici.

Xiaomi Redmi Note 4	
CPU	Snapdragon 625 (8x 2000MHz)
RAM	3 GB
OS	Android 7.1
Rozlišení fotoaparátu	13 Mpix

Tabulka 8.1: Specifikace testovacího zařízení

8.1.1 Kolekce testovacích fotografií

Všechny testovací fotografie složení potravin byly vyfoceny mobilním telefonem, jehož specifikace je uvedena v tabulce 8.1. Získané fotografie byly následně nahrány na server a anotovány, viz části 5.2.1 a 5.2.2.

Finální testovací kolekce se skládala ze 105 fotografií složení potravin. Kolekce neobsahovala žádné duplicity, tj. v ní obsažené fotografie zobrazovaly složení navzájem různých potravin. Do kolekce nebyly zahrnuty fotografie složení, jež byly rozmazané nebo poškozené světelnými odlesky takovým

způsobem, který bránil čitelnosti textu nebo jeho části. V kolekci 105 fotografií bylo 40 fotografií se světlým textem na tmavém pozadí a 65 fotografií s tmavým textem na pozadí světlém. Dále 52 fotografií obsahovalo složení se šikmým textem a 37 fotografií obsahovalo složení s textem viditelně zakřiveným.

8.2 Výsledky

Tabulka 8.2 zobrazuje výsledky testování kvality extrakce aditiv z fotografií složení v závislosti na použité metodě předzpracování obrazových dat. Výsledky uvedené v posledním sloupci tabulky náleží finální implementaci, která využívá předzpracování obrazových dat, viz část 7.1. V řádcích tabulky 8.2 nalezneme mikro a makro průměry dosažených přesností P, úplností R a F-mír. V poslední řádce je uveden průměrný čas extrakce aditiv z 1 testovací fotografie.

	binarizace	binarizace & úprava rozlišení	binarizace & úprava rozlišení & deskewing	binarizace & úprava rozlišení & deskewing & dewarping
F_{macro}	0,82	0,84	0,85	0,87
F_{micro}	0,83	0,85	0,85	0,88
R_{macro}	0,78	0,80	0,81	0,84
R_{micro}	0,74	0,76	0,78	0,82
P_{macro}	0,91	0,93	0,94	0,94
P_{micro}	0,94	0,96	0,95	0,95
Čas extrakce aditiv z 1 fotografie (průměr)	10 457 ms	9 142 ms	8 648 ms	8 142 ms

Tabulka 8.2: Výsledky v závislosti na předzpracování fotografií složení

Tabulka neobsahuje výsledky pro ten případ, kdy testovací fotografie nebyly nijak předzpracovány, tj. OCR analýza byla prováděna na původních nepředzpracovaných testovacích fotografiích složení, neboť v takovém případě byla průměrná doba zpracování 1 fotografie delší než 2 minuty, a testování tak z praktických důvodů nebylo dokončeno.

Správnost výpočtu hodnotících metrik (přesnosti, úplnosti, apod.) byla v rámci diplomové práce důsledně otestována a logika jejich výpočtu je 100% pokryta jednotkovými testy napsanými pomocí frameworku JUnit a knihovny Mockito.

8.2.1 Diskuze výsledků

Testování na základě testovací kolekce fotografií složení potravin (viz tabulka 8.2) prokázalo, že implementovaný systém dosahuje při rozpoznávání přídavných látek z fotografií složení potravin dobrých výsledků, viz tabulka 8.3. Vysoká přesnost rozpoznání přídavných látek z fotografií složení potravin je dána zejména implementací algoritmu pro extrakci aditiv z rozpoznaného textu, který filtruje přídavné látky detekované na základě kolizních slov, viz část 7.3, a předchází tak falešně pozitivním nálezům. Uspokojivá je rovněž úplnost rozpoznaných aditiv, která mírně přesahuje 80%, tj. systému se podaří ve fotografii složení potraviny průměrně rozpoznat 4 z 5 obsažených přídavných látek.

F_{macro}	F_{micro}	R_{macro}	R_{micro}	P_{macro}	P_{micro}	Čas extrakce aditiv z 1 fotografie (průměr)
0,87	0,88	0,84	0,82	0,94	0,95	8 142 ms

Tabulka 8.3: Shrnutí konečných výsledků

Výsledky uváděné v tabulkách 8.2 a 8.3 mohou být částečně zkresleny relativně malou kolekcí testovacích fotografií. Z výsledků testování je však patrné, že předzpracování obrazových dat, tj. jejich binarizace, deskewing, dewarping apod., má pozitivní vliv jak na kvalitu dosažených výsledků, tak na průměrnou dobu extrakce aditiv z jedné fotografie. Předzpracování obrazových dat zvyšuje úspěšnost OCR analýzy a výrazně snižuje dobu jejího výpočtu, a tudíž i dobu celého procesu extrakce aditiv z fotografie složení potraviny.

Dosažené výsledky dokazují, že je možné využít knihovnu Tesseract OCR v úloze extrakce označení přídavných látek z fotografií složení potravin, ale fotografie složení potravin je nutné nejprve předzpracovat, tj. alespoň provést jejich binarizaci.

9 Závěr

Výstupem diplomové práce je systém pro automatickou extrakci přídavných látek z fotografií složení potravin. Výsledný systém se skládá ze serverové aplikace, webového klienta a mobilní aplikace pro platformu Android. Mobilní aplikace umožňuje jejímu uživateli pořídit fotografii složení potraviny skrze fotoaparát integrovaný v mobilním zařízení, pořízený snímek předzpracuje a rozpozná obsažený text. Rozpoznaný text je odeslán serverové aplikaci. Ta z rozpoznaného textu vyextrahuje označení přídavných látek, výsledek extrakce je následně předán zpět mobilní aplikaci, která nalezené přídavné látky prezentuje uživateli a uživatel si může zobrazit jejich detailní popis.

V rámci diplomové práce byla vyzkoušena řada technik a metod předzpracování obrazových dat za účelem zvýšení úspěšnosti optického rozpoznávání znaků. Byly vyzkoušeny knihovny pro manipulaci s obrazem OpenCV a Leptonica. Proces optického rozpoznávání znaků je prováděn knihovnou Tesseract OCR. Pro extrakci aditiv z rozpoznaného textu byl navržen vlastní algoritmus, jehož implementace je postavena na knihovně Hibernate Search.

Dále byl navržen a implementován mechanismus pro automatické ověření kvality extrahovaní aditiv z fotografií složení potravin, jenž byl využit k vyhodnocování výsledků extrakce aditiv během vývoje systému a rovněž ke konečnému zhodnocení úspěšnosti procesu extrakce aditiv z fotografií složení.

Úspěšnost extrakce aditiv byla vyhodnocena pomocí testovací kolekce fotografií. Proces extrakce aditiv z testovací kolekce fotografií dosahuje relativně dobrý výsledků. Průměrná úplnost (*recall*) aditiv extrahovaných z testovací kolekce fotografií přesahuje 80% a přesnost (*precision*) extrakce přesahuje 90%, viz část 8. Dosažené výsledky a uživatelská přívětivost systému prokazují, že je možné využít knihovnu Tesseract OCR v úloze extrakce označení přídavných látek z obalů potravin. Všechny body zadání diplomové práce byly splněny.

Budoucí práce by se měly zaměřit zejména na možnost implementace vlastního algoritmu pro korekci a detekci chyb OCR analýzy jako tomu bylo v [19]. Přínosné by rovněž bylo rozšíření stávajícího uživatelského systému nebo možnost personifikace vzhledu aplikace a zobrazovaného obsahu.

Literatura

[1] Bangare, S. L. – Bangare, P. – Patl, S. Reviewing otsu's method for image thresholding. *International Journal of Applied Engineering Research.* 2015, 10, 9, s. 777–783. ISSN 0973-4562.

[2] Bloomberg, D. *Analysis of Document Skew* [online]. 2002. [cit. 2019/04/03]. Leptonica Documentation. Dostupné z: `http://www.leptonica.com/papers/docskew.pdf`.

[3] Bloomberg, D. *Dewarping text pages* [online]. 2010. [cit. 2019-03-03]. Leptonica Documentation. Dostupné z: `http://www.leptonica.com/dewarping.html`.

[4] Bloomberg, D. *Leptonica Documentation* [online]. 2018. [cit. 2019-02-05]. Leptonica Documentation. Dostupné z: `http://www.leptonica.com`.

[5] Bloomberg, D. S. – Kopec, G. E. – Dasari, L. Measuring document image skew and orientation. In *Document Recognition II*, 2422, s. 302–317. International Society for Optics and Photonics, 1995. doi: 10.1117/12.205832.

[6] Boyd, R. *Getting started with OAuth 2.0.* O'Reilly Media, Inc., 2012. ISBN 1449311601.

[7] Bukhari, S. S. – Shafait, F. – Breuel, T. M. Dewarping of document images using coupled-snakes. In *Proceedings of Third International Workshop on Camera-Based Document Analysis and Recognition, Barcelona, Spain*, s. 34–41. Springer India, 2009. ISBN 978-81-322-1906-4.

[8] Chaki, N. – Shaikh, S. H. – Saeed, K. A comprehensive survey on image binarization techniques. In *Exploring Image Binarization Techniques.* Springer, 2014. s. 5–15. ISBN 978-81-322-1907-1.

[9] Chaudhuri, A. et al. *Optical Character Recognition Systems for Different Languages with Soft Computing.* Springer International Publishing, 2017. ISBN 978-3-319-50252-6.

[10] Davis, M. *Unicode Text Segmentation* [online]. 2019. [cit. 2019-02-05]. Dostupné z: `http://hibernate.org/search/`.

[11] Duda, R. O. – Hart, P. E. Use of the Hough transformation to detect lines and curves in pictures. Technical report, Menlo Park CA, 1971.

[12] Eaton, B. *Food Additives: Definition, History, and Debate* [online]. 2018. [cit. 2019-04-05]. Dostupné z: `https://bruceeatonphd.wordpress.com/2016/04/08/food-additives-definition-history-and-debate/`.

[13] Garg, N. K. Binarization Techniques used for grey scale images. *International Journal of Computer Applications*. 2013, 71, 1. doi: 10.5120/12320-8533.

[14] Hadjadj, Z. et al. ISauvola: Improved Sauvola's algorithm for document image binarization. In *International Conference on Image Analysis and Recognition*, s. 737–745. Springer, 2016. doi: 10.1007/978-3-319-41501-7_82. ISBN 978-3-319-41500-0.

[15] He, J. et al. A comparison of binarization methods for historical archive documents. In *Eighth International Conference on Document Analysis and Recognition*, s. 538–542. IEEE, 2005. doi: 10.1109/ICDAR.2005.3.

[16] Hlaváč, V. *Matematická morfologie* [online]. České vysoké učení technické v Praze, 2018. [cit. 2019-03-09]. Dostupné z: `http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/71-3MatMorpholBinCz.pdf`.

[17] Hlaváč, V. – Sedláček, M. *Zpracování signálů a obrazů*. Vydavatelství ČVUT, 2000. ISBN 80-01-02114-9.

[18] Hough, P. V. Machine analysis of bubble chamber pictures. In *Conf. Proc.*, 590914, s. 554–558, 1959.

[19] Jiří, M. Inteligentní vyhledávání dokumentů. Master's thesis, Západočeská univerzita, Fakulta aplikovaných věd, 2017.

[20] Klescht, V. – Hrnčiříková, I. – Mandelová, L. *Éčka v potravinách*. Computer press, 2006. ISBN 8025114834.

[21] Kolouchová, M. Morfologické operace ve zpracování obrazu. Master's thesis, Vysoké učení technické v Brně, Fakulta informačních technologií, 2008.

[22] Ma, K. et al. DocUNET: Document image unwarping via a stacked U-Net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, s. 4700–4709, 2018. doi: 10.1109/CVPR.2018.00494.

[23] Maglie, A. *Reactive Java Programming*. Springer, 2016. ISBN 978-1-4842-1429-9.

[24] Miller, F. P. – Vandome, A. F. – McBrewster, J. *Apache Maven*. Alpha Press, 2010. ISBN 9786130652197.

[25] Morris, T. – Mellor, E. *Improve OCR Quality* [online]. 2019. [cit. 2018-12-10]. Dostupné z: `https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality`.

[26] Narasimhan, H. et al. Optimizing the Multiclass F-Measure via Biconcave Programming. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, s. 1101–1106. IEEE, 2016.

[27] Ohlsson, V. Optical Character and Symbol Recognition using Tesseract. Master's thesis, Luleå University of Technology Department of Computer Science, 2016.

[28] Powers, D. M. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *J. Mach. Learn. Technol.* 2011, 2. doi: 10.9735/2229-3981.

[29] Ravi, S. – Khan, A. Morphological operations for image processing: understanding and its applications. In *Proc. 2nd National Conference on VLSI, Signal processing & Communications NCVSComs*, 2013.

[30] Rogowska, J. Digital image processing techniques for speckle reduction, enhancement, and segmentation of optical coherence tomography (OCT) images. *Optical Coherence Tomography: Principles and Application*. 2006. doi: 10.1088/0031-9155/47/4/307.

[31] Saxena, L. P. Niblack's binarization method and its modifications to real-time applications: a review. *Artificial Intelligence Review*. 2017, s. 1–33. doi: 10.1007/s10462-017-9574-2.

[32] Shukla, B. K. – Kumar, G. – Kumar, A. An approach for Skew Detection using Hough Transform. *International Journal of Computer Applications*. 2016, 136, 9, s. 20–23.

[33] Singh, T. R. et al. A new local adaptive thresholding technique in binarization. *CoRR*. 2012.

[34] Software, R. *Hibernate Search Documentation* [online]. 2019. [cit. 2019-02-05]. Hibernate Search Documentation. Dostupné z: `http://hibernate.org/search/documentation/`.

[35] Soille, P. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013. ISBN 978-3-662-05088-0.

[36] Son, K.-C. – Lee, J.-Y. The method of android application speed up by using NDK. In *2011 3rd International Conference on Awareness Science and Technology (iCAST)*, s. 382–385. IEEE, 2011. doi: 10.1109/ICAwST.2011.6163104.

[37] TYLOVÁ, K. Aditivní (přídatné) látky v potravinách a jejich vlastnosti. Master's thesis, University of South Bohemia in České Budějovice, Faculty of Education, České Budějovice, 2008.

[38] Tížková, J. Bezpečnost vybraných doplňků stravy z pohledu přítomnosti přídavných látek. Master's thesis, Univerzita Karlova, Farmaceutická fakulta v Hradci Králové, 2014.

[39] Ulges, A. – Lampert, C. H. – Breuel, T. M. Document image dewarping using robust estimation of curled text lines. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, s. 1001–1005. IEEE, 2005.

[40] Walls, C. *Spring Boot in action.* Manning Publications, 2016. ISBN 9781617292545.

[41] ČESKO. *Vyhláška č. 4/2008 Sb.* [online]. 2008. [cit. 2018-11-05]. Dostupné z: `https://www.zakonyprolidi.cz/cs/2008-4/`.

[42] Špelina, V. – Ostrý, V. Potravinová přecitlivělost: alergie a intolerance. *Vědecký výbor pro potraviny.* 2003. Dostupné z: `http://czvp.szu.cz/vedvybor/dokumenty/studie/alerg_2003_3_deklas.pdf`.

A Sestavení a spuštění aplikací

V této příloze jsou uvedeny příručky pro sestavení a následné spuštění jednotlivých komponent systému. Nejprve musí být přeložena serverová aplikace, viz část A.1, a poté webový klient nebo mobilní aplikace, jenž závisí na artefaktech serverové aplikace, viz části A.2 až A.3.

A.1 Serverová aplikace

Pro sestavení serverové aplikace je nutné mít nainstalovaný nástroj **Maven** verze minimálně 3.0. Maven musí být správně nakonfigurovaný, tj. proměnné prostředí systému musí obsahovat proměnou `JAVA_HOME`, jež obsahuje cestu k používané Javě 1.8, a proměnou `M2_HOME` obsahující cestu k nástroji Maven. Jsou-li tyto prerekvizity splněny, tak se stačí přepnout do kořenového adresáře serverové aplikace a použít příkaz `install`, viz kód A.1. Po úspěšném sestavení nalezneme ve složce *server-application/target* spustitelný soubor aplikace `server-application.jar`.

```
mvn clean install
```

Kód A.1: Překlad serverové aplikace

Před spuštěním aplikace je potřeba nakonfigurovat databázi, jež aplikace potřebuje ke své činnosti, viz tabulka A.1. Rovněž musíme nastavit JDBC konektor databáze v *BOOT-INF/classes/application.properties* (cesta v JAR archivu).

Jméno databáze	db_fotecka
Porovnání	utf8mb4_unicode_ci
Uživatel	fotecka_user
Heslo	password

Tabulka A.1: Výsledky v závislosti na předzpracování fotografií složení

V souboru *application.properties* rovněž musíme nastavit cestu k adresáři, kde budou ukládány testovací fotografie složení, a cestu k adresáři, kde bude uložen index knihovny Spring Search, viz kód A.2. Jsou-li splněny všechny prerekvizity, tak spuštění serverové aplikace provedeme příkazem v kódu A.3.

```
# Where the uploaded photots are stored
file.upload-dir=c:/fotecka/labels
# Where the search indexes are stored
file.upload-dir=c:/fotecka/indexes
```

Kód A.2: Konfigurace serverové aplikace

```
java -jar server-application.jar
```

Kód A.3: Spuštění serverové aplikace

A.2 Webový klient

Pro překlad webové klienta je nutné mít nainstalovaný nástroj **npm** verze 5.6.0 nebo vyšší. Pomocí nainstalovaného **npm**, je následně třeba stáhnout a nainstalovat sadu nástrojů **Angular CLI**, viz kód A.4.

```
npm install -g @angular/cli
```

Kód A.4: Instalace Angular CLI

Jsou-li splněny výše uvedené prerekvizity, tak se přepneme do kořenového adresáře webového klienta. Překlad a následné spuštění provedeme v kořenovém adresáři pomocí příkazů v kódu A.5.

```
npm install
ng serve --open
```

Kód A.5: Překlad a spuštění webového klienta

A.3 Mobilní aplikace

Pro překlad mobilní aplikace pro platformu Android je nutné mít nainstalovaný nástroj Android Studio verze 3.1.3 nebo vyšší. Rovněž musí být stažena a nainstalována sada nástrojů Android NDK (*Native Development Kit*)[1]. Jsou-li splněny všechny prerekvizity, tak samotný překlad provedeme skrze

[1]Detailní popis instalace Android NDK a jeho následnou konfiguraci lze nalézt v *https://developer.android.com/ndk/guides*

grafické uživatelské rozhraní nástroje Android Studio nebo pomocí nástroje Gradle v příkazové řádce[2] v kořenovém adresáři mobilního klienta.

Přeloženou aplikace můžeme nainstalovat do mobilního telefonu pomocí grafického uživatelského rozhraní nástroje Android Studio (skrze tzv. ADB). Na cílovém zařízení musíme nejprve povolit tzv. „Vývojářský mód“ a povolit debugování přes USB. Případně můžeme vzít instalační APK soubor přeložené aplikace, jenž bude po překladu umístěn v adresáři *android-client/app/build/outputs/apk*, uložit jej na souborový systém mobilního zařízení a kliknutím na něj spustit instalaci. Aplikaci lze nainstalovat pouze na zařízení s operačním systémem Android verze 5.0 (Lolipop) nebo vyšší.

[2]Postup pro překladu Android aplikace z příkazové řádky nalezneme na *https://developer.android.com/studio/build/building-cmdline*

B Uživatelské příručky

Součástí této přílohy jsou uživatelské příručky pro mobilní aplikaci a webového klienta, viz části B.1 a B.2.

B.1 Mobilní aplikace

V této části nalezneme uživatelskou příručku mobilního klienta. Tato příručka vysvětluje základní principy práce s mobilním klientem od registrace, viz část B.1.1, až po spuštění automatického testování kvality extrakce aditiv, viz část B.1.5.

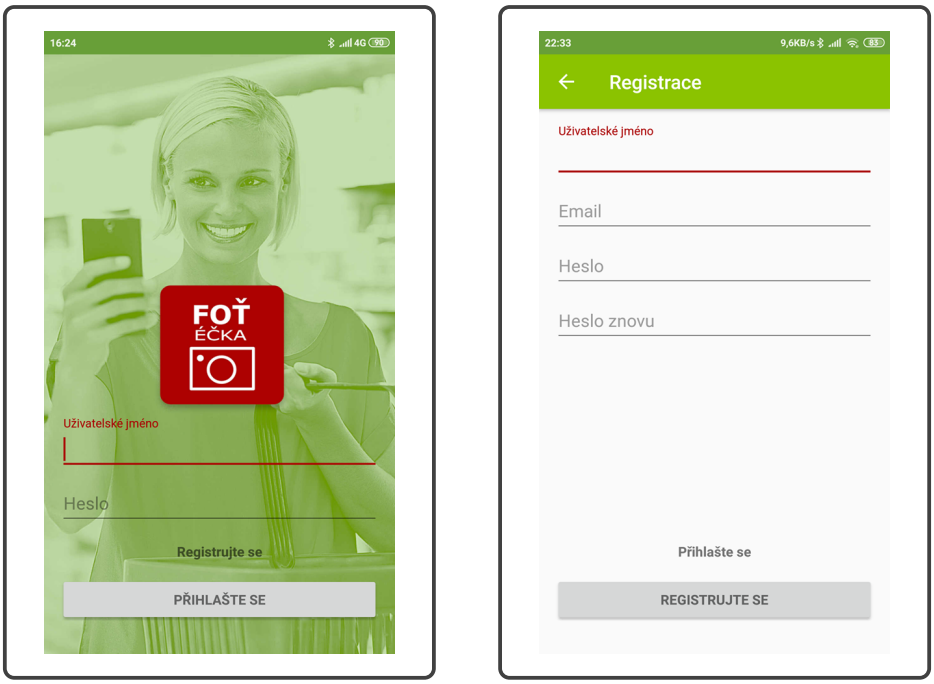
B.1.1 Přihlášení a registrace

Po spuštění aplikace je nepřihlášenému uživateli zobrazena přihlašovací obrazovka, viz obr. B.1. Pokud se již uživatel registroval do systému, tak se může přihlásit zadáním svého uživatelského jména a hesla. Neregistrovaný uživatel může provést registraci kliknutím na tlačítko „Registrujte se“ a následně vyplnit registrační formulářf.

B.1.2 Navigace

Po úspěšném přihlášení nebo registraci je uživateli zobrazena hlavní obrazovka s bočním vysouvacím menu, viz obr. B.2. Přihlášený uživatel může provést následující úkony:

- **Vyhledat aditivum v katalogu**: Kliknutím na tlačítko „Vyhledat éčka“ dojde k zobrazení grafického rozhraní, které uživateli umožní vyhledávání přídavných látek, viz část B.1.3.
- **Extrahovat aditiva z fotografie složení**: Kliknutím na tlačítko „Vyfotit složení“ dojde ke spuštění procesu extrakce aditiv z fotografie složení, viz část B.1.4.
- **Spustit testování**: Stiskem tlačítka „Testování“ spustí uživatel proces automatického ověřování kvality extrakce aditiv na základě testovacích fotografií složení, viz část B.1.5.
- **Zobrazit informace o aplikaci**: Kliknutím na tlačítko „O aplikaci“ se uživateli zobrazí základní informace o aplikaci, jejím cíli a původu.



(a) Přihlášení (b) Registrace

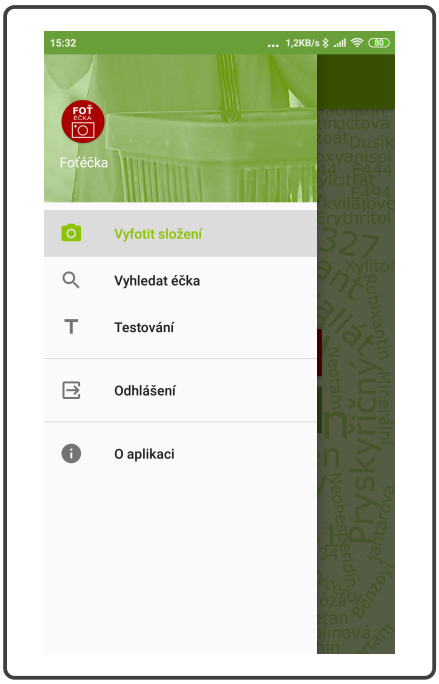
Obrázek B.1: Uživatelské rozhraní pro nepřihlášeného uživatele

- **Odhlásit se**: Stisknutím tlačítka „Odhlášení“ se uživatel odhlásí z aplikace.

B.1.3 Katalog přídavných látek

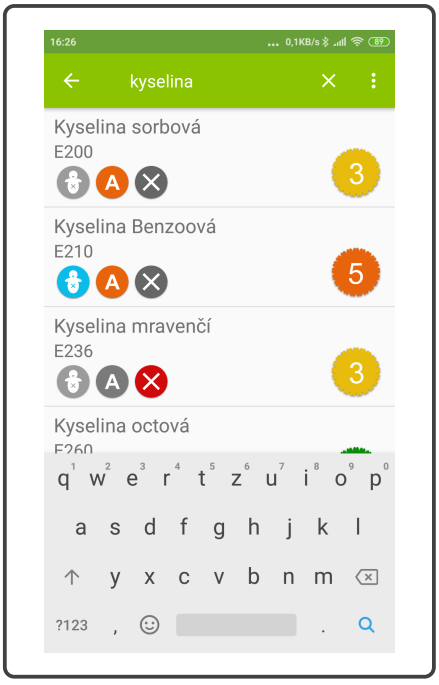
Aplikace umožňuje uživateli vyhledávat v katalogu přídavných látek. Přídavné látky lze vyhledávat pomocí jejich názvu nebo E kódu, viz obr. B.3. Aplikace uživateli průběžně našeptává aditiva na základě aktuálně zadaného vstupu. Kliknutím na položku v seznamu nalezených přídavných látek zobrazíme její detailní popis. Součástí popisu přídavné látky je její charakteristika, výrobní proces, možné nežádoucí účinky a doplňující informace. Součástí popisu jsou rovněž ikony a grafické ukazatele, které indikují škodlivost dané přídavné látky a upozorňují uživatele na přídavné látky, jež mají alergenní účinky, jsou nevhodné pro děti nebo je jejich používání v České republice zakázáno.

(a) Hlavní obrazovka

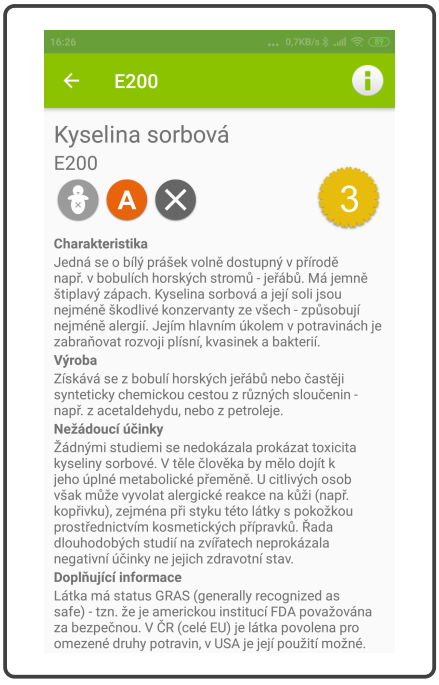


(b) Vysouvací menu

Obrázek B.2: Prvky hlavní obrazovky



(a) Vyhledávání



(b) Detail aditiva

Obrázek B.3: Katalog přídavných látek

Význam jednotlivých ikon a ukazatelů v detailu aditiva je vysvětlen v legendě, viz obr. B.4. Legendu může uživatel zobrazit kliknutím na ikonu písmena „i“ v pravém horním rohu detailu aditiva.

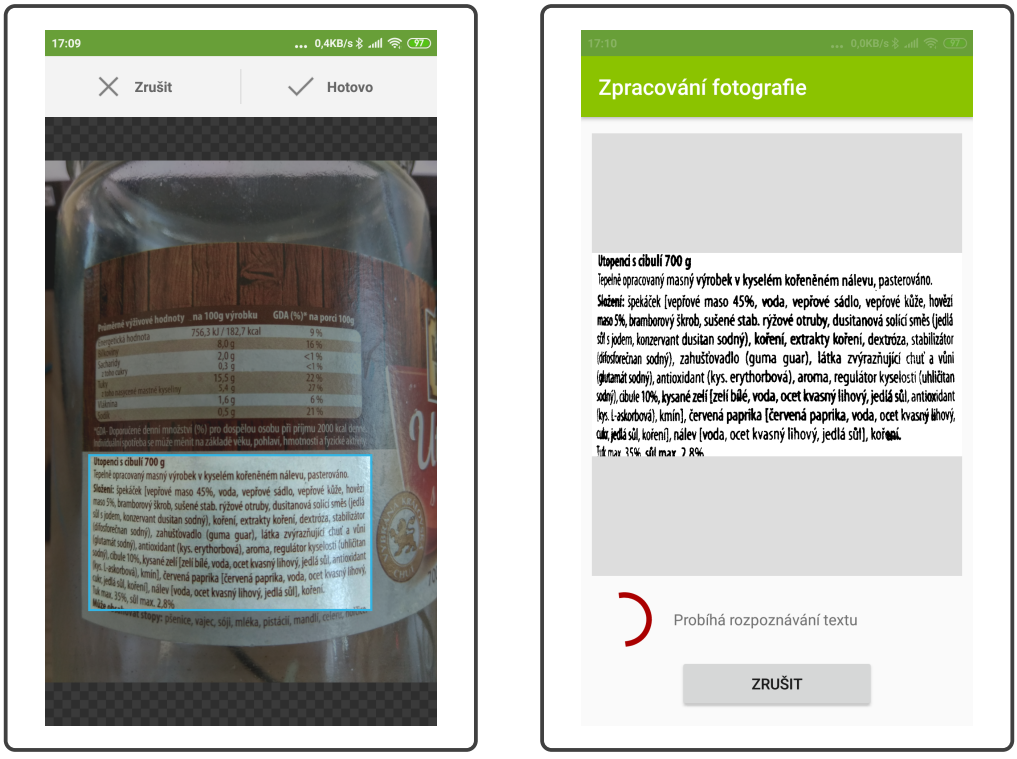
Obrázek B.4: Legenda s vysvětlivkami

B.1.4 Extrakce aditiv z fotografie složení

Proces extrakce aditiv z fotografie složení spustí uživatel stiskem tlačítka „Vyfotit složení“, které nalezne v hlavní obrazovce aplikace, viz část B.1.2. Uživateli se nejprve zobrazí náhled fotoaparátu a je vyzván k pořízení fotografie složení. Je-li spokojen s kvalitou pořízené fotografie složení, tak fotografii potvrdí. Následně je uživatel vyzván, aby na fotografii označil výskyt českojazyčného textu složení, viz obr. B.5. Cílem je označit jen a pouze text složení, znatelně tak urychlíme celý proces a dosáhneme obecně lepších výsledků extrakce přídavných látek.

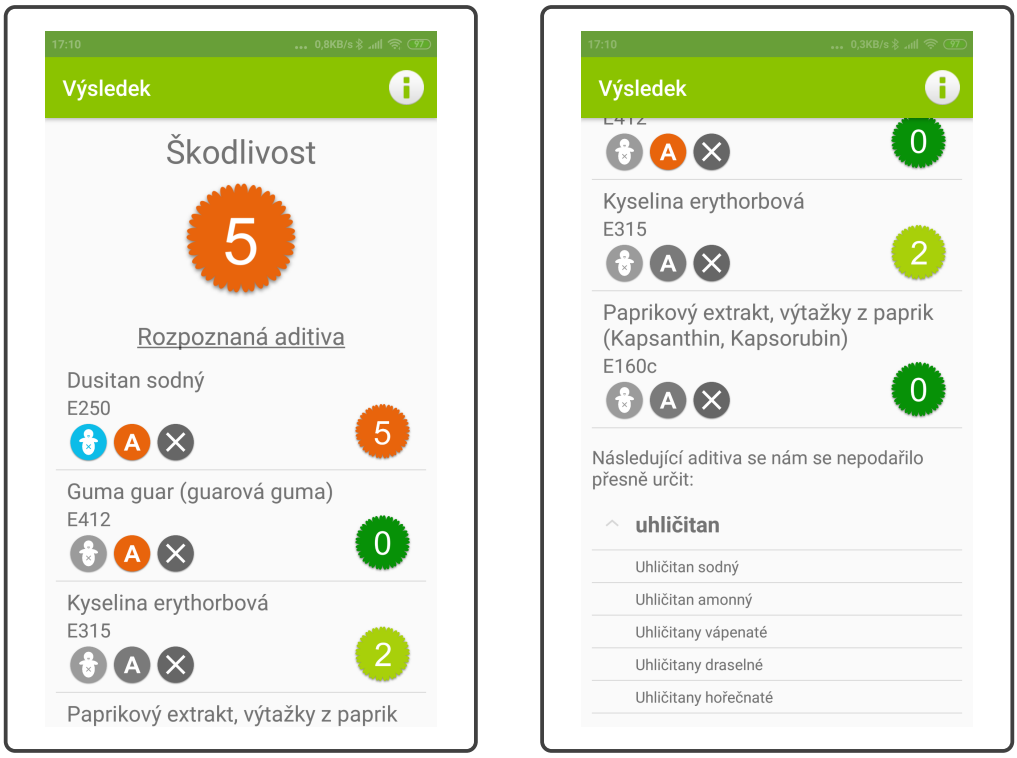
Po výběru oblasti textu složení je zahájen samotný proces zpracování fotografie a extrakce přídavných látek. Celková doba procesu extrakce aditiv v průměru nepřesahuje 10s. Výsledek extrakce je následně uživateli prezentován, viz obr. B.6. Výsledek se skládá ze tří částí:

- **Jednoznačně určená aditiva**: Jedná se o přídavné látky, jenž se podařilo z fotografie složení potraviny jednoznačně určit. Kliknutím na libovolné z nich zobrazíme jeho detail.
- **Nejednoznačně určená aditiva**: Jedná se o aditiva, která se systému nepodařilo jednoznačně určit. Například nalezne-li systém v rozpoznaném textu pouze slovo „uhličitan“, tak není schopen určit zda se jedná o uhličitan sodný, amonný nebo draselný. Uživateli jsou prezentovány všechny možné varianty uhličitanů. Pro každou z nich může uživatel zobrazit její detail.
- **Skóre škodlivosti potraviny**: Celkové skóre škodlivosti potraviny je určeno jako maximální skóre škodlivosti v seznamu jednoznačně určených přídavných látek.



(a) Označení textu (b) Zpracování fotografie

Obrázek B.5: Extrakce aditiv z fotografie

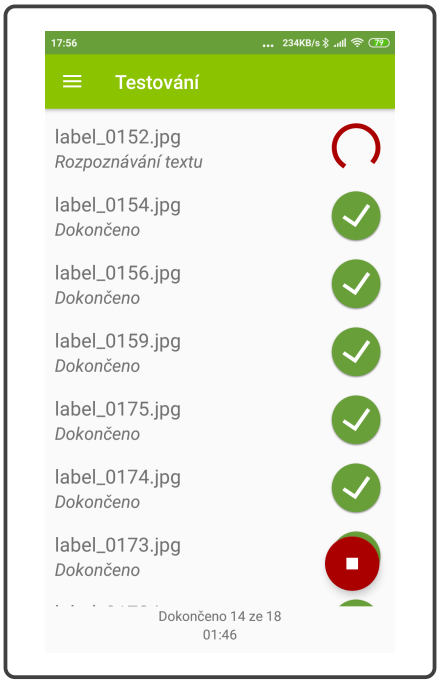


(a) Jednoznačné (b) Nejednoznačné

Obrázek B.6: Výsledky extrakce přídavných látek

B.1.5 Testování

Stisknutím tlačítka „Testování“, jenž se nachází ve vysouvacím menu hlavní obrazovky, zahájí uživatel proces automatického ověřování úspěšnosti extrakce přídavných látek, viz část 5.2. Průběh testování může sledovat pomocí grafického rozhraní, jež je zobrazeno na obrázku B.7. Nalezne v něm celkový počet testovaných fotografií, informaci o tom, která fotografie je aktuálně zpracovávána, a které fotografie již byly zpracovány. Po ukončení testování si uživatel může zobrazit detailní výsledky testování prostřednictvím webového klienta, viz část B.2.6.



Obrázek B.7: Průběh testování

B.2 Webový klient

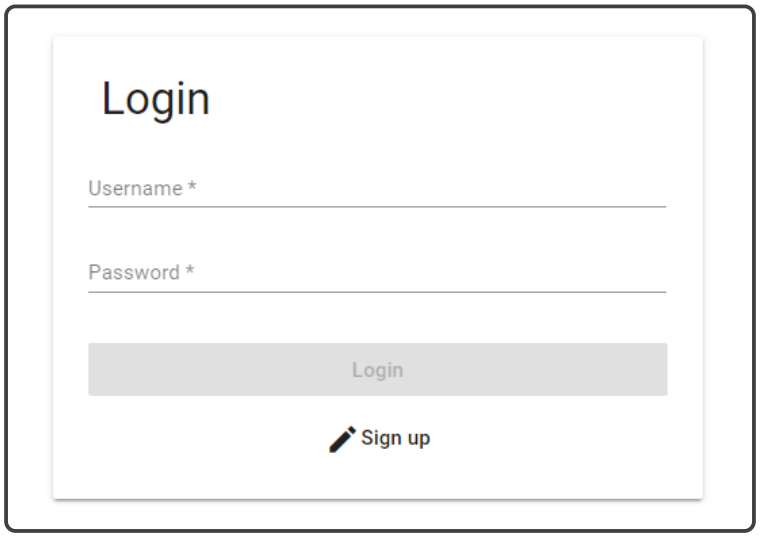
V této části nalezneme uživatelskou příručku webového klienta. Tato příručka vysvětluje základní principy práce s webovým klientem od registrace, viz část B.2.1, až po správu výsledků automatického testování kvality extrakce aditiv, viz část B.2.6.

B.2.1 Přihlášení a registrace

Nepřihlášenému uživateli je při přístupu k webovému klientu nejprve zobrazen přihlašovací formulář, viz obr B.8. Přihlášení uživatel provede zadáním uživatelského jména a hesla a následným kliknutím na tlačítko „Login“. Pokud by se chtěl uživatel registrovat jako nový administrátor systému, může tak učinit kliknutím na tlačítko „Sign up“ a vyplněním zobrazeného registračního formuláře.

B.2.2 Navigace

Pro úspěšném přihlášení do webového klienta je uživateli zobrazen katalog aditiv, viz část B.2.4. V levém horním rohu obrazovky nalezneme logo sys-



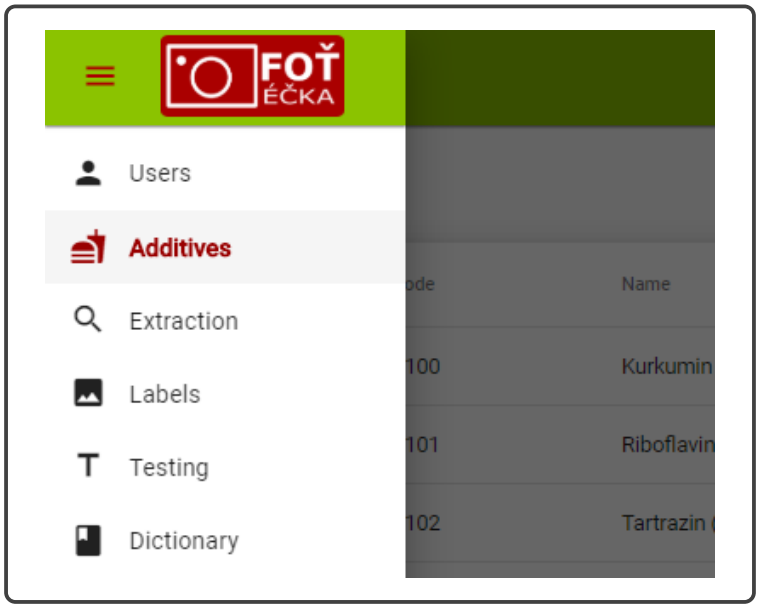
Obrázek B.8: Přihlašovací formulář

tému a tlačítko pro zobrazení navigačního menu, viz obr. B.9. Navigační menu webového klienta můžeme vidět na obrázku B.10.

Obrázek B.9: Tlačítko pro zobrazení navigačního menu

V navigačním menu nalezneme následující položky:

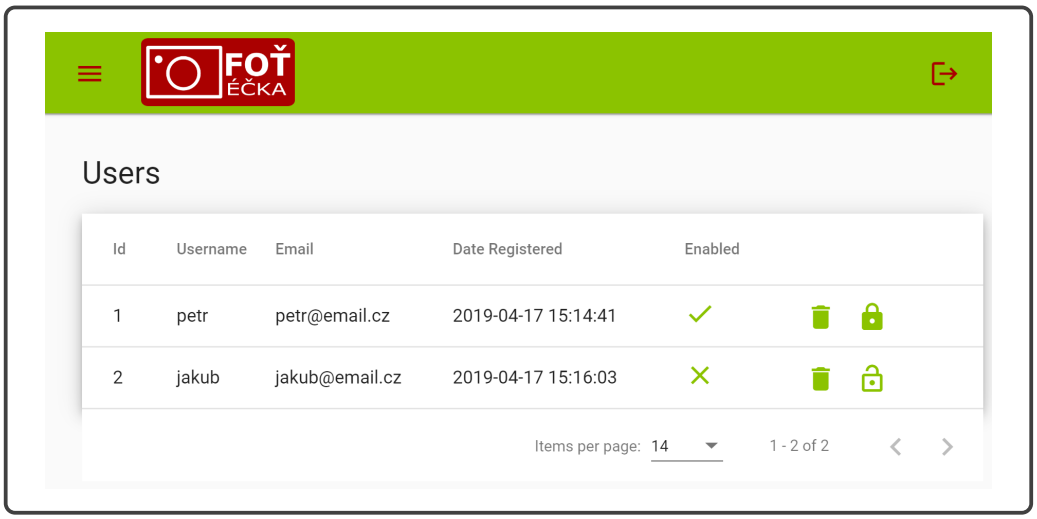
- **Users**: Kliknutím na položku zobrazíme uživatelské rozhraní pro správu uživatelů, viz část B.2.3.
- **Additives**: Kliknutím na položku zobrazíme uživatelské rozhraní pro správu přídavných látek, viz část B.2.4.
- **Labels**: Kliknutím na položku zobrazíme uživatelské rozhraní pro správu testovacích fotografií, viz část B.2.5.
- **Testing**: Kliknutím na položku zobrazíme uživatelské rozhraní pro správu výsledků testování úspěšnosti extrakce aditiv, viz část B.2.6.
- **Dictionary**: Kliknutím na položku zobrazíme uživatelské rozhraní pro správu seznamu kolizních slov.



Obrázek B.10: Navigační menu

B.2.3 Správa uživatelů

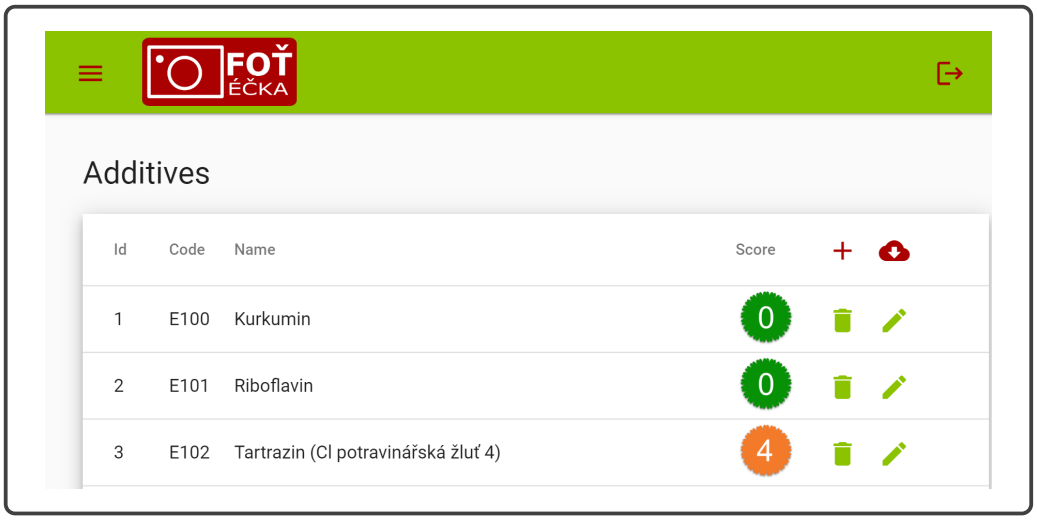
Pro jednoduchou správu uživatelů registrovaných do systému slouží tabulka viditelná na obrázku B.11. Registrovaný uživatel je reprezentovaný 1 řádkou tabulky. Kliknutím na ikonu odpadkového koše daného uživatele odstraníme, kliknutím na ikonu zamčeného či odemčeného zámku můžeme uživatelův účet zablokovat nebo odblokovat.



Obrázek B.11: Tabulka registrovaných uživatelů

B.2.4 Správa aditiv

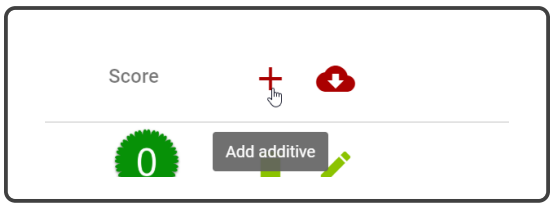
Přídavné látky uložené v systému můžeme zobrazovat a modifikovat prostřednictvím tabulky, kterou můžeme vidět na obrázku B.12. Řádka tabulky reprezentuje jednu přídavnou látku. Řádky tabulky můžeme řadit kliknutím na text hlavičky sloupce.



Obrázek B.12: Tabulka přídavných látek

Přidání nových a modifikace stávajících aditiv

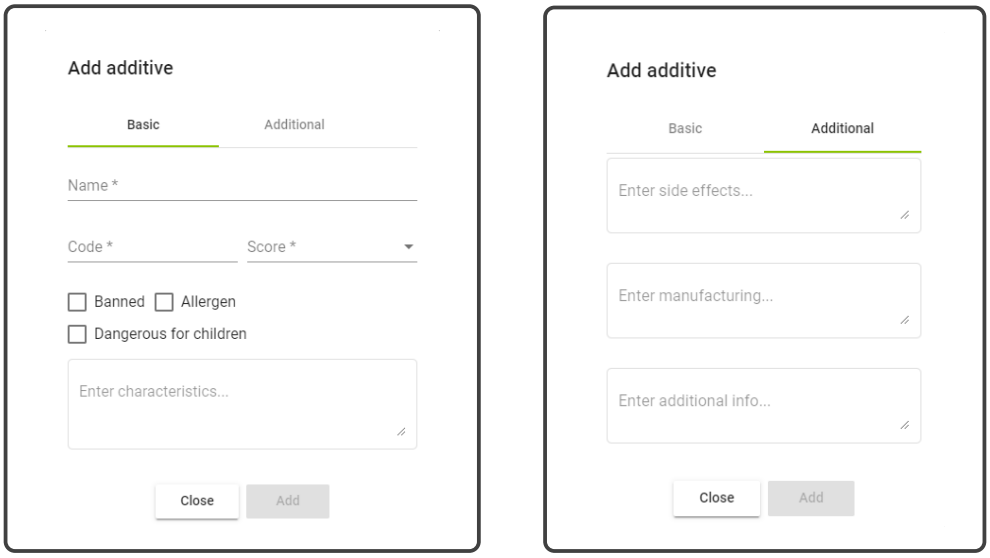
Kliknutím na tlačítko „+“ v hlavičce tabulky (viz obr. B.13) dojde k zobrazení dialogu sloužícího pro přidání nového aditiva do systému, viz obr. B.14. Dialog obsahuje dvě záložky „Basic“ a „Additional“.



Obrázek B.13: Tlačítko pro přidání aditiva

Kliknutím na záložku „Basic“ dojde k zobrazení vstupních polí pro vyplnění základních informací o daném aditivu. Jmenovitě se jedná o jeho jméno, E kód, skóre škodlivosti, základní popis a informace o tom, zda je tato přídavná látka zakázaná v EU, má alergenní charakter nebo je nevhodná pro děti. Pro úspěšné přidání aditiva musí být všechny tyto položky vyplněny.

Pod záložkou „Additional“ nalezneme vstupní pole pro vyplnění dodatečných informací o daném aditivu. Konkrétně se jedná o popis jeho výrobního postupu a případných vedlejších na lidský organismus. Tyto položky nemusí být uživatelem vyplněny.



(a) Záložka „Basic“ (b) Záložka „Additional“

Obrázek B.14: Dialog pro přidání aditiva

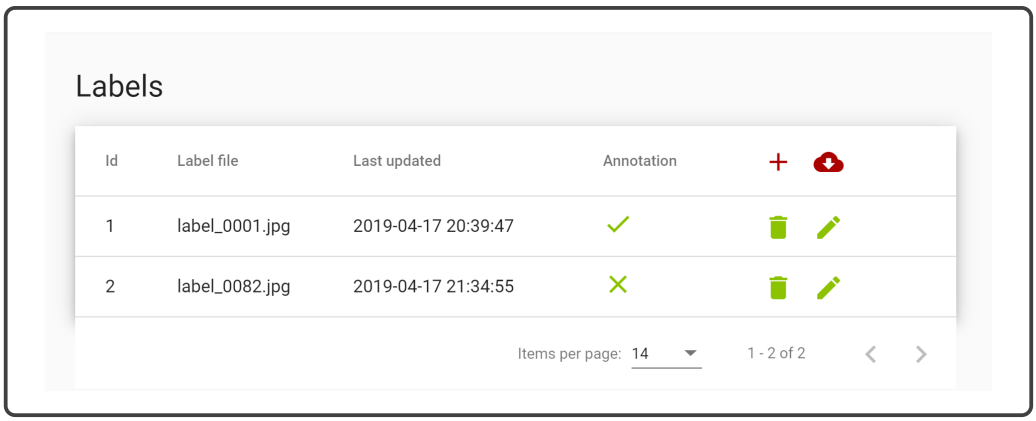
Kromě přidání nového aditiva můžeme modifikovat nebo odstranit záznamy stávajících aditiv. Odstranění provedeme kliknutím na ikonu odpadkového koše, pro modifikaci záznamu musíme kliknout na ikonu tužky, viz obr B.15.

Obrázek B.15: Tlačítka pro odstranění a modifikaci záznamu aditiva

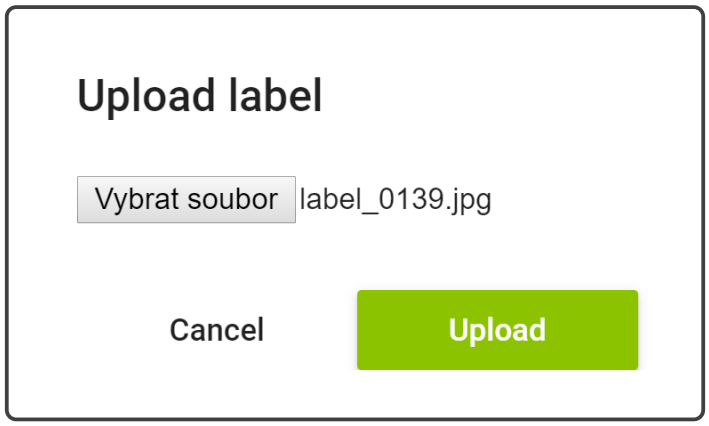
B.2.5 Správa testovacích fotografií

Pro správu testovacích fotografií složení potravin slouží tabulka na obrázku B.16. Řádky tabulky odpovídají testovacím fotografiím, jež uživatelé nahráli do systému. Testovací fotografie můžeme skrze toto uživatelské rozhraní odstranit nebo provést jejich anotaci.

Nahrání nové fotografie provedeme kliknutím na tlačítko „+“ v hlavičce tabulky a následným výběrem souboru nové testovací fotografie v zobrazeném dialogu, viz obr. B.17.



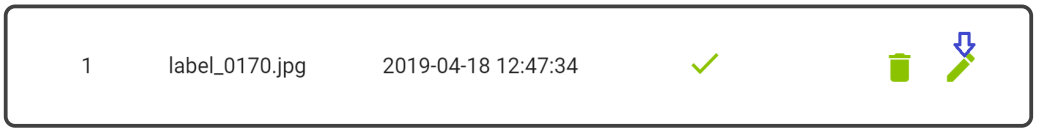
Obrázek B.16: Tabulka testovacích fotografií



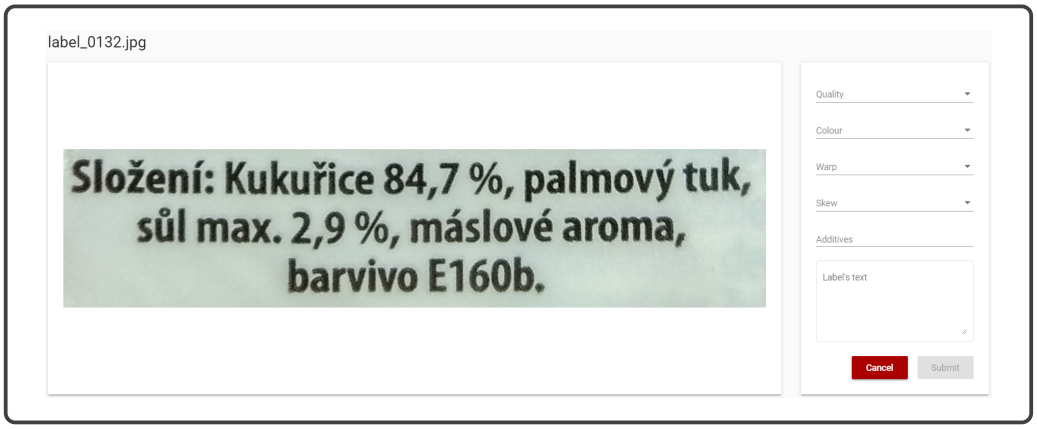
Obrázek B.17: Dialog pro nahrání testovací fotografie složení

Anotace testovací fotografie

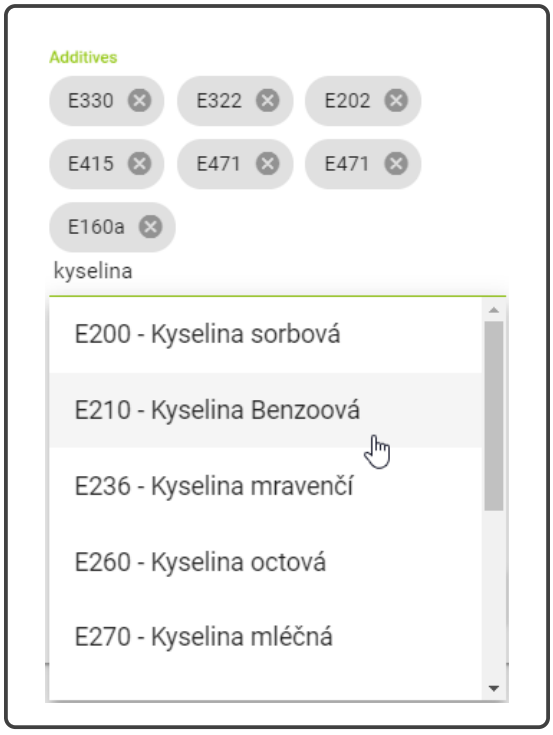
Anotaci testovací fotografie zahájíme kliknutím na ikonu tužky v odpovídajícím řádku tabulky, viz obr. B.18. Uživatelské rozhraní pro anotaci etikety se skládá ze dvou panelů, viz obr. B.19. V levém panelu je zobrazena anotovaná fotografie složení potraviny, v pravém panelu nalezneme vstupní pole pro vyplnění anotovaných položek, viz část 5.2.2. Pro manuální anotaci obsažených aditiv slouží speciální textové pole se schopností „našeptávat“ uživateli aditiva na základě aktuálně vyplněného textu, viz obr. B.20. Obsažená aditiva lze zadávat pomocí jména nebo E kódu.



Obrázek B.18: Tlačítko pro anotaci etikety



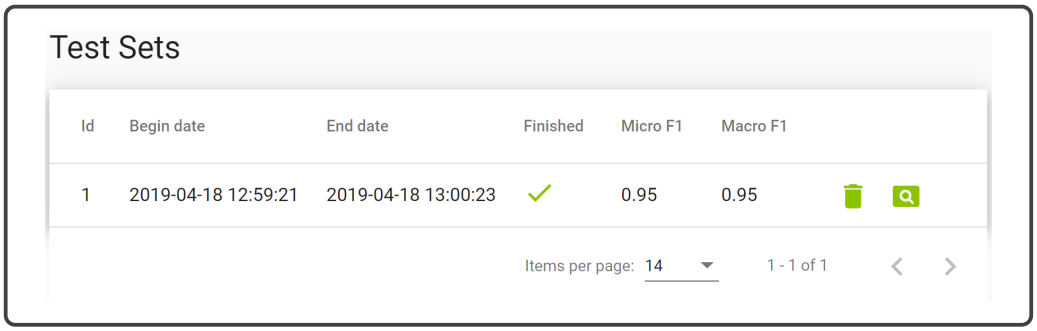
Obrázek B.19: Uživatelské rozhraní pro anotaci



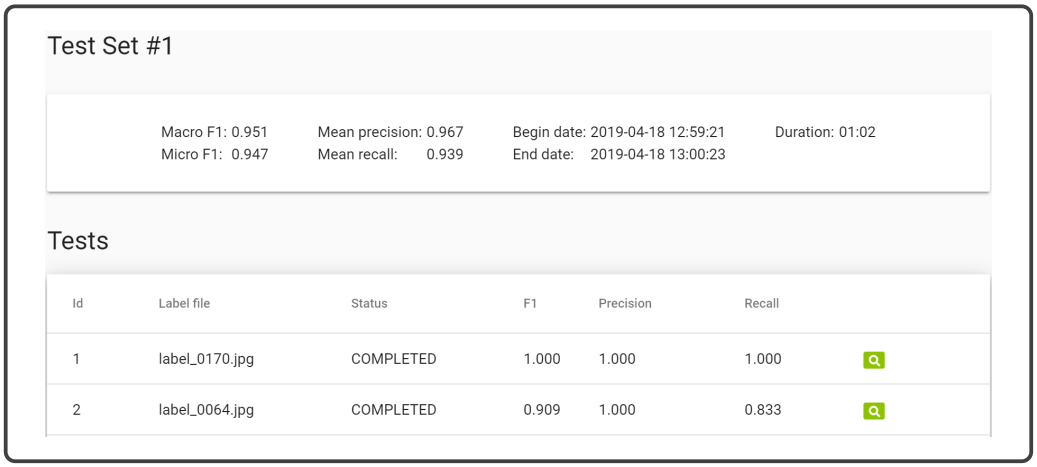
Obrázek B.20: Pole pro zadávání obsažených aditiv

B.2.6 Správa výsledků testování

Webový klient umožňuje správu a zobrazení výsledků testování úspěšnosti extrakce přídavných látek na základě testovací kolekce fotografií. Výsledky testování můžeme procházet pomocí tabulky, která je zachycena na obrázku B.21. Kliknutím na ikonu lupy v příslušné řádce tabulky zobrazíme detailní pohled na výsledek ověřování úspěšnosti extrakce aditiv, viz obr. B.22. V horním části jsou zobrazeny souhrnné metriky úspěšnosti extrakce pro celou testovací sadu fotografií. Dolní část je tvořena tabulkou dílcích výsledků dosažených pro konkrétní testovací fotografie. Kliknutím na ikonu lupy v této tabulce zobrazíme dialog obsahující detail výsledku testování pro danou testovací fotografii, viz obr. B.23. Zeleně jsou označena aditiva, které systém správně extrahoval. Červená barva je přiřazena aditivům, která jsou součástí složení, ale systém je nevyextrahoval, nebo aditivům, která systém vyextrahoval ačkoliv součástí složení potraviny nejsou.



Obrázek B.21: Tabulka výsledků testování



Obrázek B.22: Detail výsledku testování

Obrázek B.23: Detail výsledku extrakce aditiv z jedné testovací fotografie