VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

MOBILNÍ APLIKACE PRO NASKENOVÁNÍ HRY KAKURO Z NOVIN A JEJÍ DOHRÁNÍ

MOBILE APPLICATION FOR SCANNING KAKURO FROM NEWSPAPERS AND FINISHING IT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL REIN**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ DYK**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Rein Michal**
Program: Informační technologie
Název: **Mobilní aplikace pro naskenování hry Kakuro z novin a její dohrání**
Mobile Application for Scanning Kakuro from Newspapers and Finishing It
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku detekce a rozpoznání tištěných i ručně psaných číslic.
2. Prostudujte problematiku návrhu a tvorby mobilních aplikací pro iOS.
3. Navrhněte mobilní aplikaci, která nasnímá vytištěnou křížovku Kakuro a navrhne kroky pro její vyřešení.
4. Implementujte minimalistickou použitelnou verzi řešené aplikace.
5. Otestujte aplikaci na testovacím vzorku uživatelů.
6. Analyzujte zpětnou vazbu uživatelů a realizujte iterativní úpravy řešené aplikace.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání,
- rozpracovaný bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Dyk Tomáš, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Cílem této práce je vytvoření mobilní aplikace, která umožňuje naskenovat hrací plochu hry Kakuro z jakéhokoliv tištěného média a pomoci uživateli s jejím dohráním. Řešení využívá poznatky a metody z oblasti počítačového vidění a strojového učení, na základě kterých řeší problematiku detekce mřížky v obraze nebo rozpoznávání ručně psaných číslic a hracích polí za pomoci konvolučních neuronových sítí. Aplikace se celkově skládá ze serverové a klientské části. Klientské řešení zahrnuje mobilní aplikaci, vyvinutou pomocí technologie Flutter, která s pomocí serverové části aplikace, implementované v programovacím jazyce Python, zkonstruuje virtuální hrací plochu a poskytne uživateli pomoc s řešením hry. Aplikace je dostupná na zařízení s operačním systémem Android a iOS.

Abstract

The purpose of this thesis is to create a mobile application, which allows to scan a Kakuro game from any printed media and helps it's user solving it. The solution is composed of client and server sides, where client is the mobile application implemented in the Flutter framework, which collaborates with server side to construct a virtual playground and offer solution of the game to the user. This thesis also includes studies of computer vision and machine learning techniques, software engineering and algorithmic solving of the Kakuro game, which are necessary for constructing such applications. The outcome of this work is a fully functional mobile application which allows it's user to scan the game and offers help with solution. The app is available for devices with Android and iOS operating systems.

Klíčová slova

Flutter, Kakuro, konvoluční neuronové sítě, detekce mřížky, mobilní aplikace

Keywords

Flutter, Kakuro, convolutional neural networks, grid detection, mobile application

Citace

REIN, Michal. *Mobilní aplikace pro naskenování hry Kakuro z novin a její dohrání*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Dyk

Mobilní aplikace pro naskenování hry Kakuro z novin a její dohrání

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Dyka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Rein
6. května 2021

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Tomáši Dykovi za jeho veškerý čas strávený velmi přínosnými konzultacemi a poskytnutí mnoha užitečných rad. Dále bych chtěl poděkovat své rodině a přátelům, kteří se aktivně podíleli na testování výsledné aplikace.

Obsah

1 Teorie **3**
- 1.1 Kakuro . . . 3
- 1.2 Převod obrazu na stupně šedi . . . 4
- 1.3 Detekce hran . . . 5
- 1.4 Detekce přímek pomocí Houghovy transformace . . . 8
- 1.5 Rozpoznávání ručně psaných číslic . . . 9
- 1.6 Technologie pro vývoj multiplatformních aplikací . . . 10

2 Průzkum dostupných aplikací **12**
- 2.1 Distribuční služby mobilních aplikací . . . 12
- 2.2 Aplikace dostupné pro Android/iOS . . . 13
- 2.3 Shrnutí průzkumu . . . 16

3 Návrh aplikace **17**
- 3.1 Nároky na výslednou aplikaci . . . 17
- 3.2 Architektura klient-server . . . 17
- 3.3 Zpracování naskenovaných her . . . 19
- 3.4 Algoritmus pro řešení hry Kakuro . . . 21
- 3.5 Návrh GUI aplikace . . . 21
- 3.6 Revize návrhu a nové funkce . . . 24

4 Implementace aplikace **29**
- 4.1 Server . . . 29
- 4.2 Mobilní aplikace . . . 33

5 Uživatelské testování a výsledky **36**
- 5.1 Průběh testování . . . 36
- 5.2 Zpětná vazba uživatelů a úpravy aplikace . . . 37

6 Závěr **38**

A Formulář pro extrakci ručně psaných číslic **39**

B Dotazník k testování aplikace **40**

Literatura **45**

Úvod

Tato práce pojednává o vývoji aplikace nesoucí název Kakuro Solver pro mobilní zařízení s operačním systémem Android a iOS. Jedná se o aplikaci, která umožňuje uživateli naskenovat své současné řešení hry Kakuro umístěné buď v novinách, časopise nebo jakémkoliv jiném tištěném médiu a využít ji k validaci, nalezení chyby, nebo dokončení svého řešení.

Aplikace je rozdělena do dvou samostatných celků. Prvním je serverová aplikace, zaměřující se na zpracovávání snímků, ve kterých se snaží najít hrací plochu a převést všechny její části do formátu JSON. Tato reprezentace pak slouží jako vstup pro část mobilní aplikace, která poskytuje uživatelsky přívětivé virtuální prostředí pro vykreslování a řešení těchto hracích ploch.

K dosažení této funkcionality jsou využívány metody z oblasti počítačového vidění a strojového učení. V obou případech se jedná o poměrně mladé a perspektivní oblasti informatiky, které v době 21. století zažívají velký rozmach a posun kupředu. Právě díky poznatkům z těchto odvětví, můžeme v dnešní době pozorovat velké množství vynálezů, které ještě před nedávnou dobou byly pouze součástí sci-fi literatury. Mezi takové pokrokové technologie patří například autonomní vozidla, přesné rozpoznávání osob z videozáznamů nebo řada dnes již dostupných aplikací pro rozšířenou realitu.

Práce je obsahově strukturována do pěti kapitol. V první kapitole je obsažen teoretický základ technologií a metod důležitých pro realizaci řešení. Druhá kapitola se věnuje průzkumu již existujících řešení a analýze jejich uživatelských hodnocení na distribučních službách. Následuje samotný návrh aplikace, který prošel prvním uživatelským testováním a revizí na základě zpětné vazby. Čtvrtá kapitola je věnována podrobnějšímu popisu implementačních detailů a poslední pak druhému uživatelskému testování s pomocí uživatelského dotazníku.

Kapitola 1

Teorie

Tato kapitola poskytuje nezbytný teoretický základ, na jehož principech dále staví postupy a poznatky obsažené v této práci. Nejdříve je nutné přiblížit samotnou hru, kolem které vzniká výsledná aplikace. Následuje seznámení se základními technikami zpracování obrazu a stručný úvod do konvolučních neuronových sítí. V závěru kapitoly je pak uveden přehled technologií, které lze využít pro vývoj multiplatformních aplikací.

1.1 Kakuro

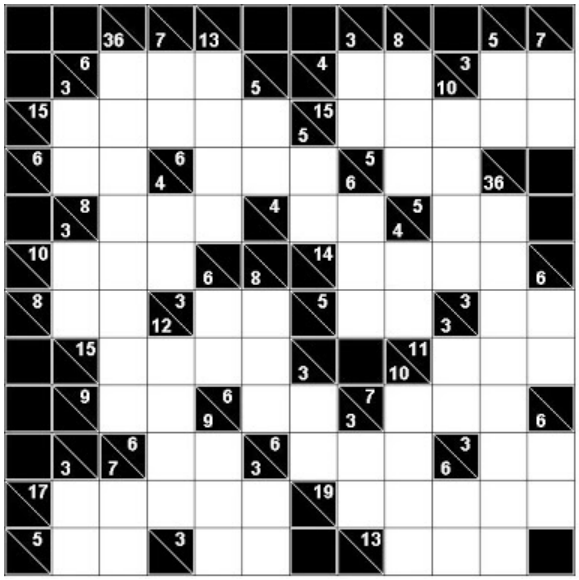
Zásadní věcí, kterou je nutné ujasnit v kontextu této práce, je definice hry, pro kterou bude výsledná aplikace určena. Tato hra nese název Kakuro a je velmi podobná světově proslulé logické hře Sudoku, avšak princip hry je odlišný.

Jako základ slouží hrací plocha v podobě mřížky, která může mít takřka libovolnou velikost ($N*M$), kde N nechť je počet řádků a M počet sloupců mřížky. Na hracích plochách existují 3 typy políček H, P a V. H je **herní pole**, do kterého lze doplnit vždy jednu číslici v rozmezí 1 až 9. P je políčko, obsahující omezení, označované jako **kontrolní součet**, platící pro posloupnost polí typu H, nacházejících se buď ve sloupci pod tímto políčkem, nebo vpravo na řádku tohoto políčka. Pole typu V pak slouží pouze jako výplň herní plochy. Obecně jsou pole typu H značena bílou, či světlou barvou, P a V pak tmavou, nejčastěji černou. Omezení kontrolního součtu platí pouze na pole, která jsou vždy bezprostředně za sebou a to buď po nejbližší konec hrací plochy nebo libovolné tmavě značené políčko. Taková posloupnost políček typu H se v herní terminologii označuje jako **segment**. Kontrolní součet pak může nabývat hodnot mezi $3 \ldots 45$ a je vždy zapsán zvlášť v podmínkovém poli pro omezení sloupcového nebo řádkového segmentu. Součet čísel v segmentu pak musí odpovídat právě hodnotě kontrolního součtu, přičemž každá číslice může být využita pouze jednou, avšak v libovolném pořadí. Na obrázku 1.1 lze vidět ukázku hrací plochy [3].

Hra Kakuro je častokrát označována jako numerická křížovka, neboť princip je téměř shodný. Cílem je vyplnit všechna políčka H, přičemž je nutné brát ohled na kontrolní součty řádku i sloupce, jichž je ono hrací pole součástí. Výzvou v této hře je projít všechny různé kombinace čísel, dávajících v součtu hodnotu omezení a nalézt ideálně vždy jeden společný možný prvek, který bude nezaměnitelný a zasazený na dané herní políčko. Hra končí úspěšným vyplněním všech herních polí.

Z hlediska algoritmické složitosti řešení se jedná o takzvaný **NP-complete**[1] problém. Společnou vlastností těchto problémů je typicky exponenciální náročnost hledání řešení,

[1] Nondeterministic polynomial-time complete



Obrázek 1.1: Ukázka hry Kakuro

které lze však následně lehce a rychle verifikovat. Dalším příkladem NP-complete problémů jsou například hry Sudoku a Heyawake [14].

Tabulka jedinečných součtů

Velmi užitečnou pomůckou při řešení Kakura je tabulka jedinečných součtů [5]. Jedná se o tabulku, ve které jsou vypsány všechny součty, které pro konkrétní velikost segmentu (počet polí, které jsou součástí daného segmentu) mají pouze jedinou možnou kombinaci čísel, ze kterých se daný segment může skládat. Takovou jedinečnou kombinaci má například součet 24 pro segment o velikosti 3, pro který existuje pouze jediná kombinace složená z čísel $7, 8$ a 9. Jestliže bychom však hledali kombinace součtu 20 pro stejnou velikost segmentu jako v předchozím případě, zjistili bychom, že zde existuje množina 4 různých kombinací, která vypadá následovně: $(3 + 8 + 9), (4 + 7 + 9), (5 + 6 + 9), (5 + 7 + 8)$. Je tedy zřejmé, že dokud nezískáme alespoň částečné řešení segmentu, nemůžeme si být jisti, kterou z kombinací uplatnit. V případě jedinečné kombinace však přesně víme, která čísla musí být využita. V kontextu celé hry pak procházíme všechna políčka a hledáme jediného společného kandidáta, kterého lze na dané pole dosadit.

1.2 Převod obrazu na stupně šedi

Velké množství technik zabývajících se zpracováním obrazu vyžaduje, aby byl vstupní obraz nejdříve předzpracován. Velmi často se jedná právě o převod obrazu na stupně šedi (angl. *grayscale*), nejčastěji z důvodu redukce barevného kanálu RGB, složeného z různé intenzity barev červené, zelené a modré, na jediný kanál stupně šedi. Konverze většinou dbá na různou vlnovou délku jednotlivých barevných složek, které různým dílem přispívají k jasu celého obrazu. V úvahu může být zahrnuta také samotná citlivost lidského oka, které je nejcitlivější na spektrum vlnové délky zelené barvy, a naopak nejméně citlivé na barvu modrou. Existuje a běžně se používá mnoho konverzních poměrů, velmi často se však setkáme s poměrem

$(0.299 * R + 0.587 * G + 0.114 * B)$, který můžeme nalézt například v knihovně OpenCV[2] [12]. Po dosazení hodnot jednotlivých barevných složek do výše uvedeného poměru pak vzniká jediná hodnota charakterizující barvu na šedotónové stupnici.

Pro oblast počítačového vidění a především rozpoznávání objektů v obraze, má převod obrazu na stupně šedi význam převážně optimalizační. Každý pixel takto převedeného obrazu představuje jeden vstup nesoucí určitou hodnotu. Při opomenutí tohoto kroku by musel program počítat s trojnásobným množstvím dat, přičemž v případě detekce hran nebo i samotné klasifikace číslic, které se bude věnovat kapitola 1.5, nemají barevné kanály téměř žádnou významnou informační hodnotu. Jak vypadá obraz převedený do šedotónové stupnice lze vidět na obrázku 1.2b.

(a) Obrázek s kanály RGB

(b) Monochromatický šedotónový obrázek

Obrázek 1.2: Porovnání standardního RGB obrázku s šedotónovým

1.3 Detekce hran

Tato podkapitola se snaží přiblížit, jakým způsobem je možné detekovat hrany objektů v obraze. Značná část poznatků vychází z prací [6] a [15], ve kterých lze nalézt celou řadu doplňujících informací nad rámec této kapitoly.

Předpokládejme, že obraz byl převeden pouze na monochromatický barevný kanál odstínů šedi a v obraze se vyskytují objekty, u kterých lze detekovat hrany (obraz tedy není například pouze vyplněn jednou barvou). Tento krok je velmi důležitý, neboť postupy, které jsou zde popsány, vycházejí z hledání náhlých odchylek v jasu, jehož hodnota je reprezentována každým pixelem šedotónového obrazu. Čím větší je tato odchylka, tím zřetelnější je nalezená hrana. K tomuto účelu pak slouží algoritmy, běžně označované jako tzv. **hranové detektory**, které pro svoji činnost využívají diferenciální operátory. Nejvýznamnějšími a nejpoužívanějšími jsou diferenciální operátory prvního a druhého řádu. Vhodná volba detektoru hran značně ovlivňuje kvalitu detekce přímek, které je věnována kapitola 1.4.

Segmentace obrazu

Důležitý pojem, který zde ještě nebyl uveden, nese název **segmentace**. Právě hranová detekce je často spojována se segmentací, jakožto jednou z jejích základních technik. Celý

[2] https://www.opencv.org/

proces segmentace je velmi složitý a většinou kombinuje dohromady více různých metod, které jako celek mohou vést k pozoruhodným výsledkům. Více informací o dalších segmentačních technikách lze nalézt v [15].

Smyslem segmentace je pak rozdělit obraz na části (segmenty), kde soustava jednotlivých pixelů, spadajících pod daný segment, koresponduje s reálnými objekty, které se vyskytují přímo v obraze. Příkladem je možné uvést jednu z problematik této celé práce, kdy za segment lze označit všechny části obrazu, které jako celek tvoří hrací plochu hry Kakuro. V tomto segmentu mohou existovat i další menší segmenty, kde každá taková část označuje jedno konkrétní hrací pole. Analogicky lze pak tento příklad převést i na segmentaci lidské tváře a částí, ze kterých se skládá (oči, ústa, nos ...).

Samotné hrany však umožňují pouze omezenou formu segmentace. Větší význam budou mít detekované hrany v kontextu této práce až ve chvíli, kdy s jejich pomocí nalezneme přímky, které budou svým logickým uspořádáním tvořit mřížku. Této fázi se bude ještě podrobněji věnovat kapitola 1.4.

Hranové detektory

Jak již bylo zmíněno dříve, nejpoužívanější metody pro detekci hran využívají první, případně druhou derivaci. Při použití první derivace se výsledný hranový gradient porovnává s prahem, pomocí kterého se určí, zda byla v daném úseku detekována hrana. Metody pracující s druhou derivací detekují hrany v případě, že je prostorová změna v polaritě druhé derivace dostatečně významná [15]. V této práci se však zaměříme především na metody využívající první derivaci, neboť právě ty budou figurovat ve výsledném řešení.

Výpočet gradientu u prvních derivací je prováděn konvolucí obrazu s jádrem konvolučního filtru, přičemž jednotlivé hranové detektory se liší právě v konfiguraci filtru. Existuje mnoho různých operátorů, mezi nejvýznamnější však řadíme především Sobelův, Prewittův, Robertsův nebo Kirschův. Každý z operátorů existuje minimálně ve dvou variantách, přičemž jedna detekuje horizontálně a druhá zpravidla vertikálně orientované hrany. Příklady významných operátorů lze vidět na obrázku 1.3.

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

(a) Robertsův řádkový operátor

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

(b) Robertsův sloupcový operátor

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

(c) Sobelův řádkový operátor

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

(d) Sobelův sloupcový operátor

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

(e) Prewittův řádkový operátor

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

(f) Prewittův sloupcový operátor

Obrázek 1.3: Ukázky nejvýznamnějších hranových operátorů

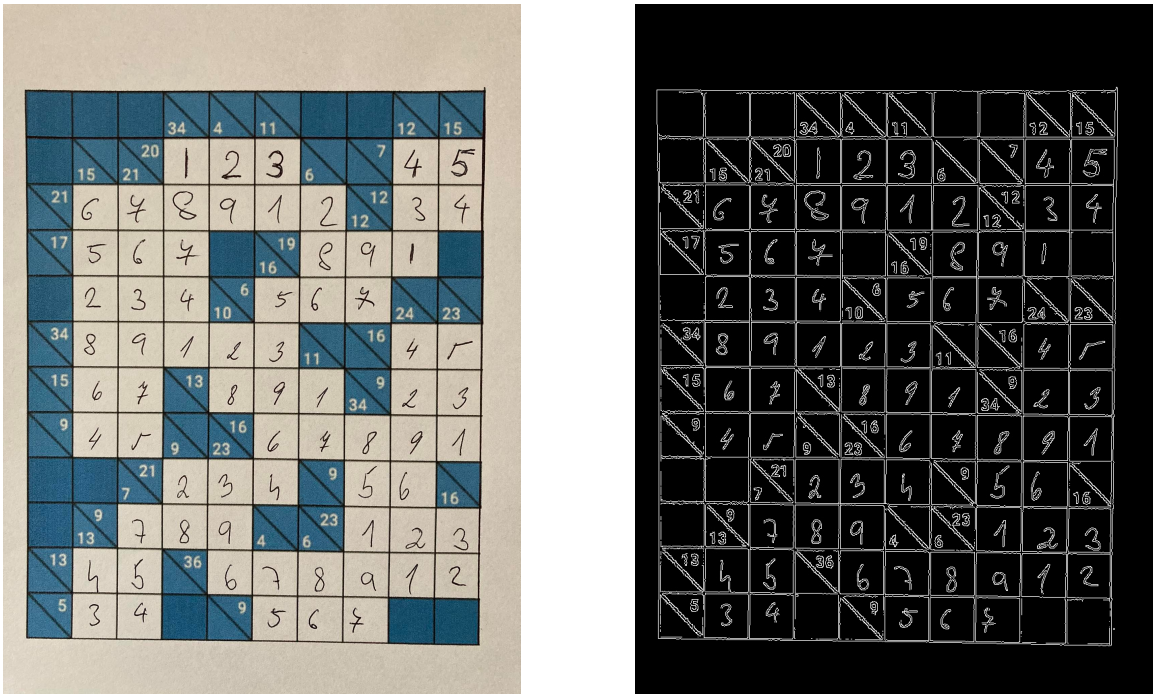
Cannyho hranový detektor

Velmi efektivní a obecně používanou metodou je detekce hran pomocí Cannyho detektoru. Tento algoritmus využívá ke své činnosti právě gradientní operátory, avšak celý proces je složen navíc ještě z několika dílčích částí.

Nejdříve je celý obraz vyhlazen aplikací Gaussova filtru, díky kterému dochází k potlačení šumu ve vstupním obraze. Dále se aplikuje (nejčastěji Sobelův) gradientní operátor jak pro horizontální, tak i vertikální směr a vypočítá se úhel gradientu vztahem:

$$Angle(\theta) = tan^{-1}\left(\frac{Gy}{Gx}\right)$$

kde Gx je první derivace horizontálního a Gy vertikálního směru. Pro konkrétní implementaci tohoto algoritmu v knihovně OpenCV se vypočtený směr dále zaokrouhlí na jeden ze 4 směrů, jež jsou horizontální, vertikální, nebo diagonální (v obou směrech) [11]. Nakonec je aplikován algoritmus pro ztenčení hran (tzv. non-maximal suppression) a dochází k prahování s hysterezí, ze kterého je následně vytvořen výsledný obraz s detekovanými hranami [6]. Příklad takového výstupu lze vidět na obrázku 1.4b



(a) Vstupní obrázek

(b) Výstup Cannyho detektoru

Obrázek 1.4: Ukázka výstupu Cannyho hranového detektoru

1.4 Detekce přímek pomocí Houghovy transformace

Houghova transformace je velmi oblíbenou a účinnou metodou pro detekci přímek v binárním obraze. Algoritmus byl navržen především pro detekci přímek, v upravených verzích je však schopen nalézt i jednoduché objekty, jako jsou například kružnice [9].

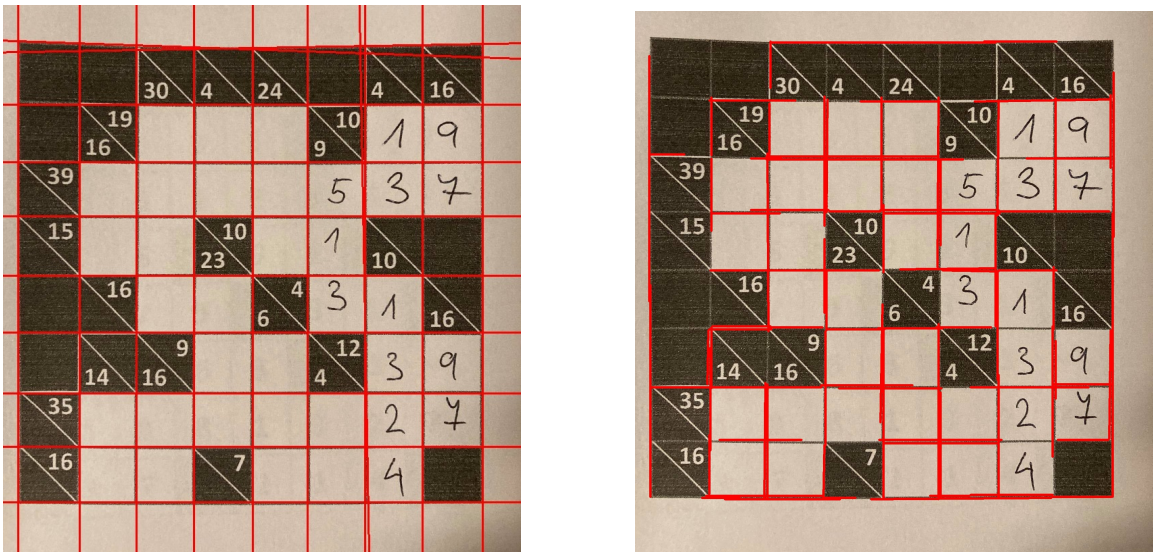
Výstupem této metody jsou přímky vyjádřené v parametrickém tvaru. Obecně lze přímku ve 2D prostoru vyjádřit pomocí rovnice:

$$y = a * x + b$$

Tento tvar má však problém s popisem vertikálních přímek, a proto je nutné použít jiné vhodnější parametrické vyjádření. Řešením problému je vyjádření přímky ze vztahu:

$$\rho = x * \cos(\theta) + y * \sin(\theta)$$

kde ρ reprezentuje délku kolmice ze středu souřadnicového systému vůči popisované přímce a θ úhel, který tato kolmice svírá s osou x [9, 10].



(a) Klasická transformace

(b) Pravděpodobnostní transformace

Obrázek 1.5: Ukázka detekce přímek pomocí Houghovy transformace

Jelikož je výstupem klasické Houghovy transformace parametrické vyjádření přímky, nelze přesně určit konkrétní množinu bodů, které spadají pod detekovanou přímku na nějakém omezeném intervalu hodnot. V úvahu jsou tak brány všechny obrazové body vyhovující výslednému vyjádření, včetně těch, které nemusí přímo ležet na detekované hraně. Tuto skutečnost lze pozorovat na obrázku 1.5a, kde červené linie představují detekované přímky, které však přesahují i mimo skutečné hrany herní plochy. Rovněž je možné pozorovat vznik chyb ve formě několikanásobné detekce téže přímky, konkrétně u předposledního sloupce hracího pole na obrázku 1.5a. Pro řešení těchto problémů se využívá tzv. pravděpodobnostní Houghova transformace [8], která umožňuje detekci linií pouze na spojitém intervalu hrany. Porovnání obou metod lze vidět na obrázku 1.5.

1.5 Rozpoznávání ručně psaných číslic

Problematika klasifikace ručně psaných číslic je jednou z často využívaných příkladů pro využití konvolučních neuronových sítí. Na internetu je k dispozici referenční dataset MNIST[3], který obsahuje celkově 70 000 snímků ručně psaných číslic v rozlišení 28x28 pixelů, na kterém lze případně síť natrénovat. Základní principy neuronových sítí jsou přehledně zpracovány v knize [2].

Konvoluční neuronové sítě

Konvoluční neuronové sítě, zkr. CNN[4], jsou speciálním typem neuronových sítí, které zpracovávají data typicky obsahující mřížkovou topologii. Může se tedy jednat například o sérii dat zachycovaných postupně v časových úsecích nějakým senzorem nebo obrázek, jež se skládá z 2D pole pixelů [4].

Samotné slovo **konvoluce** představuje speciální druh lineární operace, kterou využívá alespoň jedna z vrstev sítě, namísto klasického násobení matic, jako je tomu v případě tradičních neuronových sítí. Operace konvoluce se zapisuje pomocí znaku asterisku následovně:

$$s(t) \quad = \quad (x * w)(t) \tag{1.1}$$

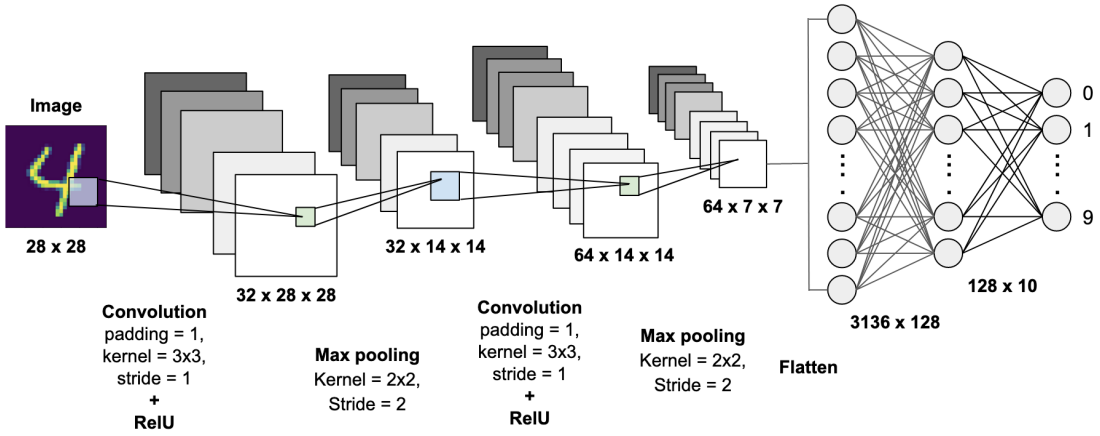
V terminologii konvolučních neuronových sítí je argument x vstupním parametrem a w představuje **konvoluční jádro** (angl. *kernel*). Výstup konvoluce $(x * w)$ bývá označován jako **mapa příznaků** (angl. *feature map*). Vstupem pak často bývají právě multidimenzionální pole dat, označované jako tenzory. Samotné konvoluční jádro je závislé právě na dimenzích vstupu a učícím algoritmu. Konvolucí různých typů jader se vstupem pak paralelně probíhá několik a vznikají tak celé série map příznaků, kde každá mapa obsahuje informace o různých charakteristikách nalezených v obraze [4]. Pro šetření paměti se využívá tzv. **pooling**, při kterém dochází k redukci velikosti vstupu a extrakci pouze dominantních rysů. K výběru hodnot se využívá nejčastěji metoda *Max-Pooling*, kdy je vybrána vždy největší hodnota daného segmentu a *Average-Pooling*, při které je spočítána průměrná hodnota v redukované oblasti.

Konvolučních a poolingových vrstev se v rámci jedné sítě může vyskytovat i více. Před samotnou klasifikací je však nutné převést příznakové mapy do plně propojených vrstev neuronů a ukončit síť výstupní vrstvou obsahující stejný počet umělých neuronů, jako je velikost množiny tříd klasifikovaných objektů. Ukázku schématu kompletní konvoluční neuronové sítě lze vidět na obrázku 1.6.

[3] http://yann.lecun.com/exdb/mnist/

[4] Convolutional Neural Networks

[5] Obrázek byl převzat z internetu: https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9



Obrázek 1.6: Příklad konvoluční neuronové sítě[5]

1.6 Technologie pro vývoj multiplatformních aplikací

Aby se aplikace mohla rozšířit k co největšímu množství uživatelů, je potřeba zvolit vhodnou platformu pro její distribuci. Na telekomunikačním trhu jednoznačně dominují zařízení s operačním systémem Android a iOS. Zastoupení uživatelů jednotlivých platforem je velmi silně ovlivněno především demograficky, avšak značnou roli hrají i jiné faktory, například pohlaví nebo věk uživatele [13].

Samotný vývoj aplikací na obě platformy zároveň bývá velmi náročný, neboť každá využívá zcela jiné technologie a je zapotřebí najít vývojáře pro obě mobilní platformy zvlášť. V poslední době však vznikají řešení, která se snaží umožnit vývojářům tvořit aplikace pod záštitou jedné společné báze kódu. Jako příklady lze uvést například Angular[6], React Native[7], Xamarin[8] nebo Flutter[9]. Samotný vývoj aplikací se tak velmi zrychluje a dochází ke snížení nákladů a času potřebného pro uvedení na trh.

Flutter

Flutter je multiplatformní open-source framwork pro vývoj aplikací s uživatelským rozhraním na mobilní zařízení, web a desktop, s podporou operačních systémů Android, iOS, macOS, Windows i Linux. Vydán byl v květnu roku 2017 společností Google, která jej dále rozvíjí a podporuje. Flutter jako celek obsahuje SDK[10] a framework, obsahující velké množství připravených elementů grafického uživatelského rozhraní, které lze dále libovolně upravovat [7].

Pro vývoj na všechny zmíněné platformy je využíván jazyk Dart[11]. Jedná se o strukturovaný, objektově orientovaný, staticky typovaný programovací jazyk, který vychází ze syntaxe jazyků C a Java. Paměť je spravována za pomocí nástroje pro automatickou správu paměti (angl. *garbage collector*). Samotný jazyk byl vyvinut rovněž společností Google a lze jej

[6] https://angular.io
[7] https://reactnative.dev
[8] https://dotnet.microsoft.com/apps/xamarin
[9] https://flutter.dev
[10] Software Development Kit - sada nástrojů pro profilování a překlad do nativních jazyků
[11] https://dart.dev

překládat do nativního kódu pro architektury ARM a x64, případně jazyka JavaScript pro nasazení na webu. Kompilace do nativního produkčního kódu probíhá v režimu AOT[12], umožňující produkovat vysoce optimalizovaný strojový kód [1]. Dle samotných vývojářů je jazyk optimalizovaný především pro tvorbu klientských aplikací zaměřených na uživatelské rozhraní.

Každá Flutter aplikace je soustava částí, kterým se říká **widgety**, dělící se na stavové (angl. Statefull) a bezstavové (angl. Stateless). Tyto komponenty jsou navzájem provázány do stromové struktury. Při změně stavu některého stavového widgetu dochází k jeho překreslení, a to včetně celého podstromu widgetů, jehož je kořenem. Vzniká zde tedy velmi široká problematika týkající se stavového managementu (angl. State management) a jeho využití pro co nejefektivnější chod aplikací [1]. Existuje tak celá řada architektur a přístupů ke správě stavu aplikace, mezi které patří například Provider[13], Redux[14] nebo BLoC[15].

[12] ahead-of-time kompilace
[13] https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple
[14] https://blog.novoda.com/introduction-to-redux-in-flutter/
[15] https://medium.com/codechai/architecting-your-flutter-project-bd04e144a8f1

Kapitola 2

Průzkum dostupných aplikací

Základem úspěchu každého produktu je pečlivý průzkum již existujících konkurenčních řešení. Smyslem je pak analýza a identifikace prvků, které se uživatelům líbí, případně nelíbí a využít tuto znalost v konkurenční výhodu. Hra Kakuro se ve světě zatím netěší tak velké pozornosti, jako například logická hra Sudoku, o to větší je však potenciál a prostor pro nová a zajímavá řešení.

Pro samotný průzkum bude využito klíčové slovo ***Kakuro*** a ***Kakuro scan***. Výsledky vyhledávání jsou rovněž porovnány se slovy ***Sudoku*** a ***Sudoku scan***, aby došlo k validaci záměru vyhledávání.

2.1 Distribuční služby mobilních aplikací

App Store

Uživatelé se zařízením iPhone vyhledávají a stahují aplikace ze služby App Store. Tato služba rovněž umožňuje uživatelům zadávat uživatelské recenze, avšak velkou nevýhodou z hlediska průzkumu je zatajování počtu stažení u konkrétních aplikací. Z výsledků vyhledávání lze tedy pouze předpokládat, že o nabízené aplikace na předních příčkách je největší zájem.

Google Play

Příznivci operačního systému Android využívají ke stahování aplikací službu Google Play. Oproti službě App Store je zde možné získat informace o počtu stažení dané aplikace, díky čemuž lze lépe analyzovat zájem o dané produkty. Pro jednotlivé aplikace nechybí rovněž možnost zadávat uživatelské recenze.

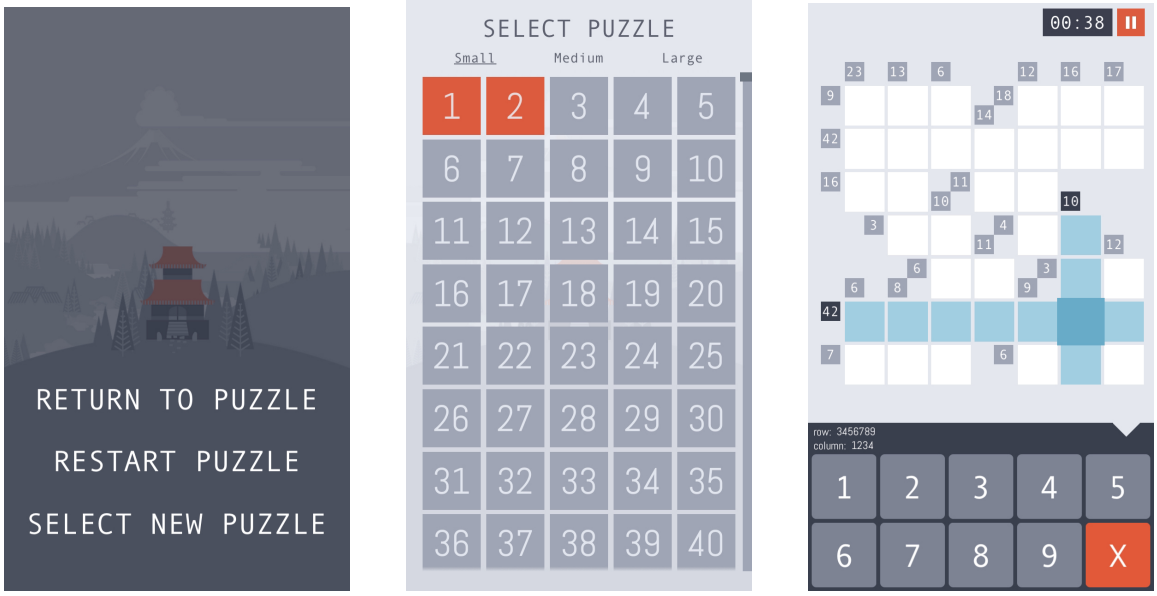
2.2 Aplikace dostupné pro Android/iOS

Klíčové slovo ***Kakuro scan*** pro platformu iOS nenabízí žádné výsledky, zkusil jsem tedy vyhledávání podle slova ***Kakuro***, přičemž jsem hledal taková řešení, ve kterých lze hru skenovat podobně jako v případě aplikací pro hru Sudoku, kterých lze na službě App Store najít hned několik. Ani v tomto případě jsem však nenalezl žádné podobné produkty.

Výsledná aplikace má však poskytovat také možnost dohrání naskenované hry uživatelem, je tedy nutné se zaměřit rovněž na ovládací prvky uživatelského rozhraní. Vyzkoušel jsem tedy službou doporučované aplikace, které jsou zaměřené na poskytnutí velkého množství již připravených her, které uživatel může řešit. Ani z těchto aplikací není možné analyzovat zpětnou vazbu od uživatelů, neboť počet uživatelských hodnocení se pohybuje pouze v jednotkách.

Kakuro Endless (iOS)

První doporučená aplikace nese název Kakuro Endless[1]. Jedná se o velmi pěkně graficky zpracovanou minimalistickou aplikaci dostupnou na iOS, která působí čistým a přehledným dojmem. Ovládací prvky poskytují příjemnou zpětnou vazbu ve formě jednoduchých animací, uživatelsky přívětivá jsou i dostatečně velká tlačítka pro zadávání číslic do jednotlivých hracích polí.



(a) Menu (b) Výběr hry (c) Herní část

Obrázek 2.1: Ukázka aplikace Kakuro Endless

Důležitým prvkem je nápověda, umístěná nad vstupními tlačítky číslic, která udává všechny číslice, které je možné doplnit na vybrané hrací pole, plynoucí z kontrolních součtů pro řádek a sloupec, na kterém se dané vstupní pole nachází. Přidružená horizontální i vertikální políčka včetně podmínek jsou rovněž po výběru zvýrazněna, díky čemuž je jednodušší se na hracím poli orientovat.

[1] https://apps.apple.com/us/app/kakuro-endless/id1327617338#?platform=iphone

Aplikace však neposkytuje žádné výrazné asistenční mechanismy nebo možnost postupovat a vracet se v krocích zpět. Absence těchto prvků je rovněž zmiňována uživateli v hodnocení na App Store, kteří by tuto možnost uvítali. Z pohledu uživatele bych rovněž uvítal alespoň minimální sledování postupu, například ve formě statistik. Rovněž není nijak využit potenciál časovače, který se na hrací ploše vyskytuje, ale informace z něj nejsou nikde zaznamenávány.

Real Kakuro (Android/iOS)

Real Kakuro[2] je aplikace dostupná jak pro platformu Android, tak i iOS. Na obou službách se ve vyhledávání vyskytuje hned na předních příčkách, přičemž na Google Play ji ohodnotilo více jak 32 000 uživatelů s průměrným hodnocením 4,7 z celkových 5ti a počet stažení sahá přes půl milionu.

Podle analýzy komentářů k hodnocení na službě Google Play, uživatelé cení především líbivé grafické zpracování, velký počet úrovní s různými obtížnostmi a možnost přepínání mezi módy dočasného zápisu „tužkou“ a samotným zápisem výsledných hodnot „perem“, které můžeme najít v dolní levé části herní obrazovky, jak lze vidět na obrázku 2.2c.

Mezi nedostatky je však často zmiňována relativně malá velikost číselníku pro zadávání vstupů a tlačítko ve tvaru šipky zpět, které uživatele přesměruje do hlavního menu. Tlačítko je umístěno přímo vedle číselníku, přičemž jej uživatelé často nechtěně stisknou při zadávání čísla 3 do hracího pole. Rovněž chybí komponenta pro návrat o krok, či několik kroků zpět.

(a) Menu (b) Výběr hry (c) Herní část

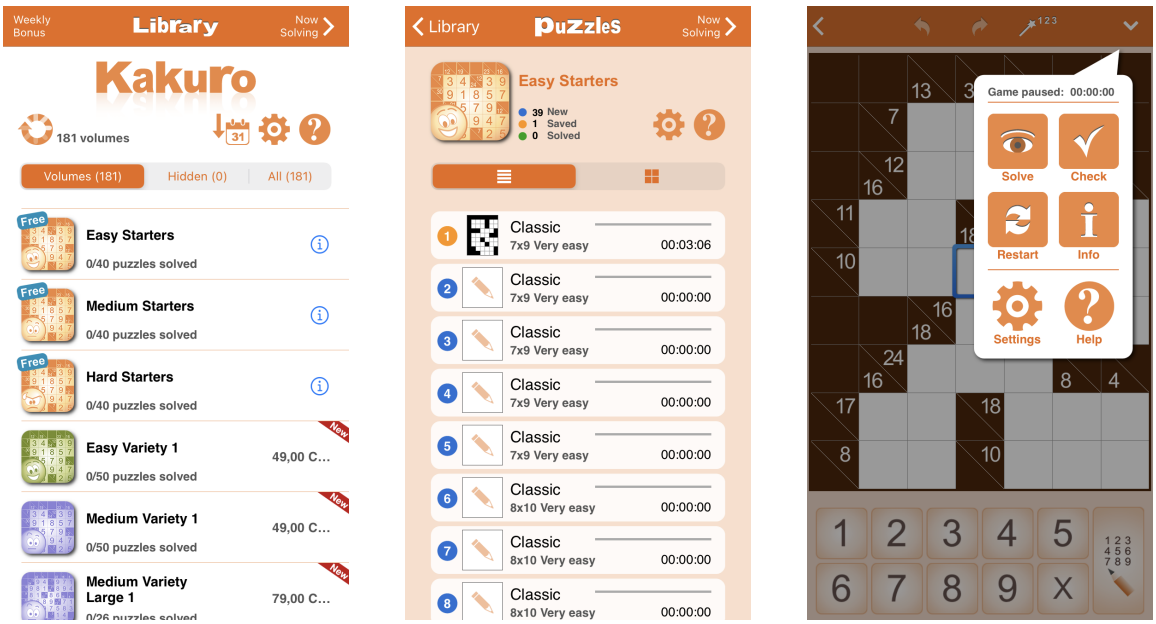
Obrázek 2.2: Ukázka aplikace Real Kakuro

[2] https://play.google.com/store/apps/details?id=co.rottz.realkakuro&hl=cs&gl=US

Conceptis Kakuro (Android/iOS)

Další možností, na kterou může uživatel narazit, je aplikace Conceptis Kakuro[3], která je dostupná rovněž na obě platformy. Na Google Play ohodnotilo aplikaci více než 1 000 uživatelů, přičemž průměrné hodnocení se pohybuje na 4,6 z celkových 5ti. Aplikace se těší více jak 100 000 celkových stažení.

Vývojáři zde zvolili způsob monetizace ve formě balíčků s hrami, rozdělených podle obtížností a velikostí. Uživateli jsou však dispozici i úvodní balíčky zdarma, které může vyzkoušet, případně bonusové úrovně, které se mění každý týden. V aplikaci nechybí základní statistiky s počtem vyřešených her a časem stráveným jejich řešením, které jsou počítány pro každý balíček zvlášť. Uživatelé kladně hodnotí možnost ověřit si, že postupují správně, a to v libovolné fázi řešení.



(a) Menu (b) Výběr hry (c) Herní část

Obrázek 2.3: Ukázka aplikace Conceptis Kakuro

[3] https://play.google.com/store/apps/details?id=com.conceptispuzzles.kakuro

2.3 Shrnutí průzkumu

Samotný průzkum poskytl velké množství užitečných informací z hlediska uživatelských nároků a priorit. Z mého pohledu je potřeba věnovat pozornost především oblasti ovládání a orientace na hrací desce. Pomoci uživateli v těchto oblastech může například zvýrazňování vybraných a jim příslušných hracích polí nebo dobře viditelná a přehledná podpora při řešení hry ve formě taháku. Důležitým prvkem je rovněž kvalitně zpracovaný, dostatečně velký číselník pro zadávání vstupu.

Zajímavým a žádaným nástrojem se jeví možnost zápisu dočasných hodnot do hracích polí, která značně napomáhá při hledání možných kombinací. Řešení této problematiky implementují všechny testované aplikace obdobným způsobem ve formě režimu zápisu „tužkou“. U této funkce jsou však k dispozici možná vylepšení, která by uživateli umožnila efektivnější přepínání mezi samotnými režimy, a to především v oblasti umístění těchto ovládacích prvků, které byly často špatně dosažitelné.

Analýza existujících řešení na úrovni platforem poukázala také na velký nepoměř v míře zájmu uživatelů o podobné aplikace. Hra Real Kakuro, implementovaná jak na platformu Android, tak iOS, byla na službě App Store hodnocena celkově 1 800 uživateli, zatímco na Google Play stejnou aplikaci hodnotilo více jak 32 000 uživatelů. Stejný jev lze pozorovat rovněž u aplikace Conceptis Kakuro. Z pohledu samotných platforem je tedy nezbytné zaměřit vývoj prioritně na zařízení s operačním systémem Android.

Kapitola 3

Návrh aplikace

Tato kapitola se věnuje popisu návrhu aplikace, kterou jako celek tvoří klientská a serverová část. Postupně tak budou rozebrány jednotlivé komponenty a jejich části, které zpřístupní uživatelům zcela unikátní funkce a zároveň umožní četná rozšíření do budoucna.

3.1 Nároky na výslednou aplikaci

Stěžejním bodem návrhu jakékoliv aplikace je jasné definování cílů, funkcionalit a případů užití daného řešení. V případě mé aplikace je cílem především poskytnout uživatelům možnost naskenování hry z fyzického média a poskytnout virtuální hrací desku, na které mohou svoji hru sami dořešit, případě nechat aplikaci, aby jim s řešením pomohla.

Pro výsledný algoritmus je stěžejní vlastností dovednost rozpoznávat rychle a spolehlivě data ve formě ručně psaných číslic, které uživatel zapisuje do jednotlivých hracích polí. Tyto vlastnosti vyžaduje rovněž rozpoznávání strojových číslic, ze kterých se skládají předem vygenerované kontrolní součty pro jednotlivé segmenty sloupců a řádků.

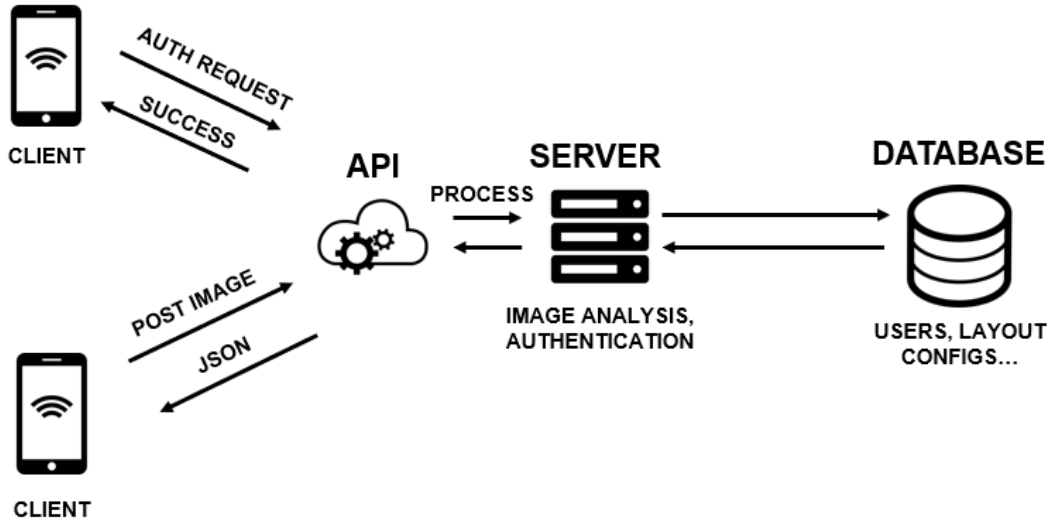
Dalším požadavkem je přesné rozpoznání rozložení hrací plochy, která může v případě hry Kakuro nabývat různých velikostí a uspořádání polí. Všechny hry však mají jednu společnou vlastnost, kterou je uspořádání polí do mřížky. Tato část je pro výslednou aplikaci taktéž kritická, neboť nesprávná detekce hrací plochy a jejich segmentů vede k celkovému znehodnocení a nemožnosti vyřešení hry.

Předpokládá se, že uživatelé nebudou pouze validovat svá řešení, ale budou aplikaci využívat také k běžnému řešení svých naskenovaných her. Z tohoto pohledu je nutné správně navrhnout uživatelské grafické rozhraní (zkr. *GUI*[1]), aby bylo jednoduché a příjemné k dlouhodobějšímu používání. Samozřejmostí je rovněž ukládání her a postupu v nich, případně možnost získat i jiné službou dostupné konfigurace hracích ploch.

3.2 Architektura klient-server

V oblasti architektury celého systému jsem se rozhodl pro přístup rozdělení aplikace na klientskou a serverovou část. Tato architektura umožňuje velmi flexibilní a pohodlný vývoj, neboť obě strany lze vyvíjet takřka nezávisle na sobě. Obecně klient pracuje s daty od serveru, který je zpracovává a transformuje, přičemž komunikace probíhá po síti, nejčastěji pomocí HTTP protokolu.

[1] angl. Graphical User Interface



Obrázek 3.1: Schéma navržené klient-server aplikace

Při návrhu systému jsem se však snažil vyvarovat přílišné závislosti klienta na serveru, aby bylo možné aplikaci využívat také v offline režimu. Typický případ užití pro tento scénář je například stažení, či naskenování her před odjezdem uživatele na místo, kde nemá přístup k internetu. Obrázek 3.1 znázorňuje schéma navržené aplikace této architektury.

Serverová část

Aplikace serverové části je koncipována jako REST[2] API[3], které pomocí metod definovaných HTTP protokolem poskytuje data a služby klientům. V aktuálním návrhu jsou modelovány tyto služby:

- Autentizace a registrace uživatelů
- Zpracování naskenovaných her
- Sdílení her mezi uživateli

Mezi možná rozšíření se však řadí například služby žebříčku nejlepších řešitelů nebo systém zkušeností a odměn. V úvahu připadá rovněž ukládání výsledků zpracovaných naskenovaných her a využití těchto dat pro budoucí vylepšování systému, například rozšíření sady dat pro trénování přesnějších neuronových sítí.

Pro uchování nezbytných informací vyžaduje server databázi, která bude vzdáleně a bezpečně spravována některým z poskytovatelů databázových služeb. Uchovávány budou údaje o uživatelích, konfigurace hracích ploch sdílených samotnými uživateli a další potřebná data v případě budoucího rozšíření.

[2] zkr. Representional State Transfer
[3] zkr. Aplication Programming Interface

Klientská část

V případě klientské části se jedná o mobilní aplikaci, která disponuje implementací metod pro řešení a vykreslování herních ploch, komunikaci se serverovou částí a práci s lokální databází. Pro pohodlnou práci a zároveň přenos informací po internetu jsou data o herních plochách ukládána do textové podoby ve formátu JSON. Návrhu uživatelského rozhraní aplikace se bude věnovat podkapitola 3.5.

3.3 Zpracování naskenovaných her

Jak již bylo nastíněno v kapitole 3.2, o zpracování obrázků se zachycenou herní plochou se stará serverová část aplikace. Samotné zpracování lze logicky rozdělit do několika částí:

- Detekce mřížkové struktury a rozdělení do segmentů
- Klasifikace typu hracího pole každého segmentu
- Detekce a klasifikace číslic u kontrolních součtů a herních polí
- Serializace dat do formátu JSON

Detekce mřížkové struktury

K zachycení mřížkové struktury v obraze je možné využít Cannyho hranový detektor a Houghovu transformaci. V rámci této práce byla využita pravděpodobnostní Houghova transformace, neboť je podstatně rychlejší na výpočet, a zároveň vykazuje lepší šance na detekci alespoň části linie mřížky oproti klasické transformaci. Právě problém možného selhání detekce některé z hran je v kontextu detekce mřížkové struktury zcela kritický. Díky nalezení alespoň malé linie dělící čáry jednotlivých segmentů hrací plochy je možné případně dopočítat body prolnutí těchto linií a složit tak celkové rozložení hrací desky.

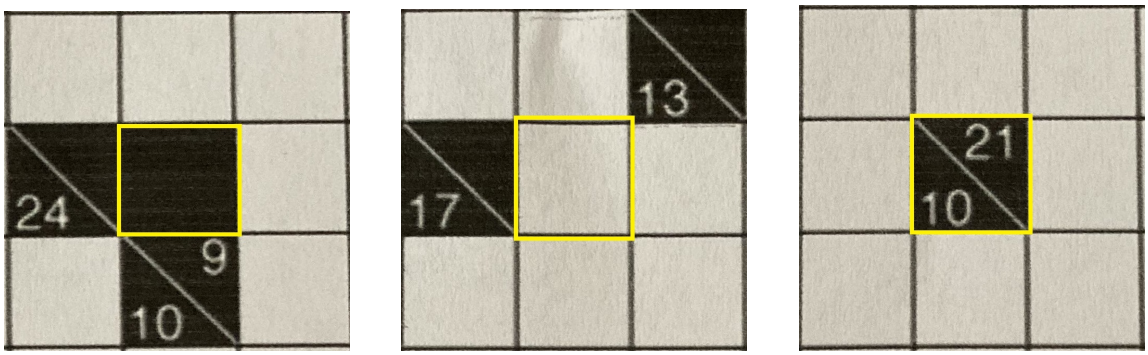
Vstupní snímek je nejdříve potřeba převést na stupně šedi a následně provést detekci hran pomocí Cannyho hranového detektoru. Nad takto zpracovaným vstupem je možné provést Houghovu transformaci, jejíž výstupem je množina nalezených přímek v obraze. Mezi liniemi pak lze dopočítat body prolnutí, které by po seřazení měly tvořit mřížku. Jestliže takové body následně uspořádáme do řádků a sloupců podle umístění v obraze, pak každá čtveřice bodů $[x_i, y_i], [x_i, y_{i+1}], [x_{i+1}, y_i], [x_{i+1}, y_{i+1}]$, kde x, y jsou obrazové body souřadnicového systému a i sloupcový/řádkový index, tvoří ohraničení segmentu hracího pole. Každý takový segment lze dále samostatně zpracovávat.

Klasifikace typu hracího pole

Pro účely následujícího kroku zpracování je nezbytné provést klasifikaci typu hracího pole. Ve hře Kakuro můžeme pozorovat celkově 3 typy polí:

- Výplňové pole
- Herní pole
- Pole kontrolního součtu

Rozhodnout o jaký typ pole se jedná lze například pomocí metody prahování obrazu nebo umělých neuronových sítí. Testováním a srovnáním obou metod jsem však došel k rozhodnutí využít umělé neuronové sítě, neboť dosahují podstatně vyšší přesnosti klasifikace i v případě různých skupin variací vstupů. Jednotlivé typy polí jsou znázorněny na obrázku 3.2, konkrétní políčka jsou zvýrazněna žlutě.



(a) Výplňové pole (b) Herní pole (c) Pole kontrolního součtu

Obrázek 3.2: Přehled typu polí vyskytujících se na herní ploše

Rozpoznávání číslic

V závislosti na typu zpracovávaného pole může být zapotřebí klasifikovat ručně, nebo strojově psané číslice. Pro tyto účely budou vytvořeny 2 různé konvoluční neuronové sítě, přičemž jedna bude trénovaná na sadě dat s ručně psanými a druhá se strojovými číslicemi.

Po extrakci všech informací z polí se data zpracují do textové podoby ve formátu JSON a následně dojde k jejich odeslání klientské aplikaci pro zpracování a vykreslení virtuálního hracího pole. Do jednotlivých záznamů je nutné zařadit rovněž informaci o hodnotě jednotlivých rozhodnutí neuronových sítí, aby mobilní aplikace mohla uživatele upozornit na možné nepřesnosti klasifikace a případně vynutit jejich kontrolu nebo opravu.

3.4 Algoritmus pro řešení hry Kakuro

Aby bylo možné uživateli s řešením pomoci, případně odhalit nesprávně naskenované hrací plochy, je nutné navrhnout postup, jakým bude výsledná aplikace hry řešit. K hledání řešení her jsem zvolil metodu backtracking podpořenou eliminačními technikami.

Pseudokód pro daný algoritmus je následující:

```
def solve(emptyFields);
    if emptyFields.length == 0 :
      return True
    else if fieldWithoutCandidatesExists(emptyFields):
      return False
    else:
      field = getFieldWithLeastCandidates(emptyFields)
      emptyFields.remove(field)
      for candidate in field.candidates:
          field.number = candidate
          if solve(emptyFields):
            return True
          field.number = None
      return False
```

Algoritmus funguje na principu rekurze, přičemž dochází k postupnému zanořování programu v rámci stavového prostoru. Úspěšnou ukončovací podmínkou je absence prázdných polí na hrací ploše, tedy stav, kdy bylo nalezeno validní řešení pro každé políčko. Jestliže některé z nevyřešených polí nemá žádného kandidáta nebo žádný z kandidátů nevede dále k možnému řešení, je předchozí instanci funkce na zásobníku volání předána informace o selhání a dochází k hledání nové cesty. Výběr prázdného pole k doplnění závisí na celkovém počtu jeho kandidátů, přičemž jsou přednostně vybírána ta pole, která mají kandidátů nejméně.

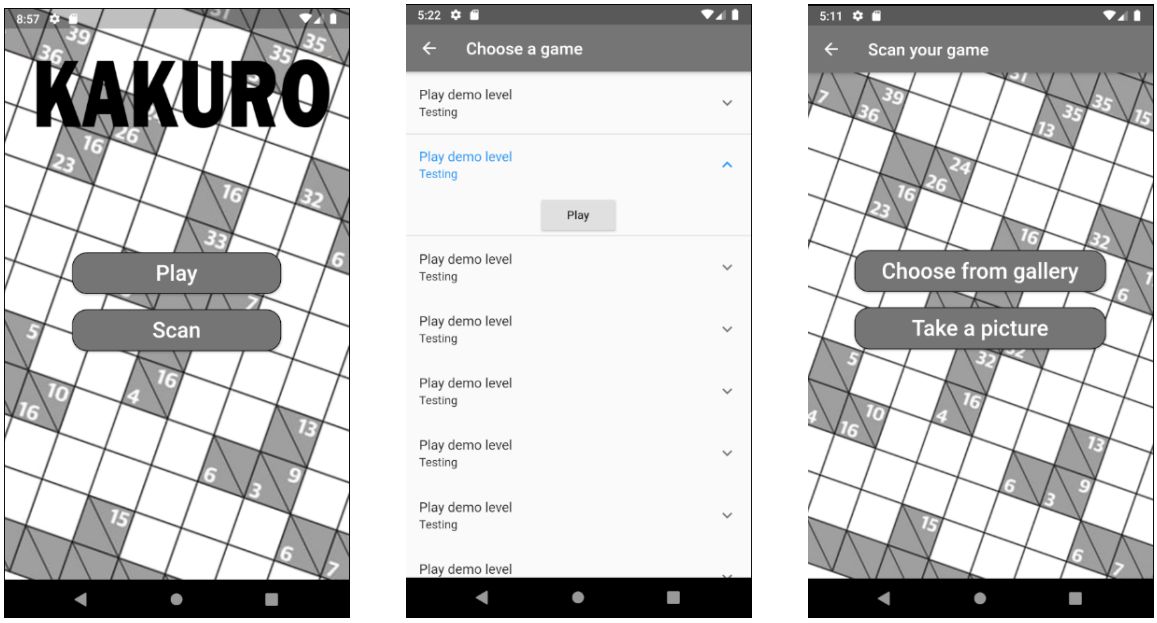
3.5 Návrh GUI aplikace

Pro návrh GUI aplikace jsem se rozhodl implementovat velmi jednoduchý prototyp, který si lze přímo prohlédnout na fyzickém zařízení a otestovat proces výběru hry a zadávání vstupu do hracích polí na předdefinované hrací ploše. Na obrázku 3.3 a 3.4 lze vidět základní pohledy navržené aplikace, konkrétně hlavního menu, nabídky pro výběr uložených her, nabídky pro možnosti skenování a samotné herní části s ovládacími prvky základních funkcí.

Navigační část

Hlavní nabídka umožňuje navigaci do dvou základních logických celků aplikace. Pomocí tlačítka „Play“ se uživatel přesune do herní části aplikace, ve které si může vybrat a pokračovat v řešení některé z již naskenovaných her. Tlačítko „Scan“ pak umožňuje přístup do nabídky pro skenování, ve které lze vybrat způsob získání snímku pro samotné zpracování. Celkově tato nabídka nabízí dvě možnosti výběru. Uživatel může snímek buď pořídit

pomocí fotoaparátu ve svém mobilním zařízení, nebo jej vybrat z galerie již pořízených snímků. Jednotlivé pohledy nabídek lze vidět na obrázku 3.3.



(a) Hlavní menu (b) Menu výběru hry (c) Menu skenování

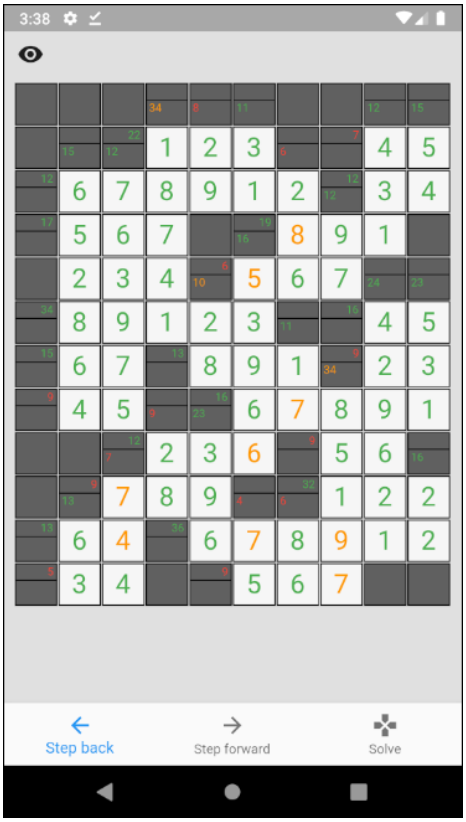
Obrázek 3.3: Prvotní návrhy navigace GUI výsledné aplikace

Herní část

Návrh herní obrazovky znázorňuje obrázek 3.4. Při návrhu jsem zohlednil různorodou velikost hracích ploch a snažil jsem se ovládací prvky uskupit takovým způsobem, aby samotná hra měla dostatek místa a vždy byla vidět v co největším možném rozsahu. Hrací plochu lze rovněž libovolně přiblížit, oddálit nebo posouvat v rámci interních omezení i za hranice obrazovky.

Mezi funkce hrací části patří především možnost vracení se o krok zpět, odhalení dalšího validního kroku nebo vyřešení celé hry. Tyto funkce jsou lokalizovány ve spodní části obrazovky, přičemž lze tuto nabídku skrýt pomocí ikonky oka na liště v horní části. V budoucnu lze aplikaci rozšiřovat o další funkce, přičemž lze znovu využít a rozšířit spodní nabídku nebo horní lištu.

Zadávání číselných vstupů je realizováno selektorem čísel, který se objeví jako vyskakovací okno po výběru některého z hracích polí. Různorodé barvy číslic viditelných na hrací ploše prototypu indikují důvěru klasifikačního algoritmu v dané rozhodnutí, neboť se předpokládá, že vstupem pro herní část aplikace bude převážně výstup klasifikátoru po přijetí a zpracování naskenované hry. Tento prvek má především za úkol informovat uživatele o možných nesprávně detekovaných segmentech a pomocí barev usnadnit hledání těchto chyb. Uživatel tak může jednotlivé chyby snadněji nalézt a případně opravit.



Obrázek 3.4: Návrh herní části

Tento návrh jsem následně nabídl na vyzkoušení úzkému okruhu osob a analyzoval jejich zpětnou vazbu. Při testování jsem se uživatelů dotazoval na přehlednost menu sekce, ovládání herní plochy a umístění jednotlivých funkcí v herní části obrazovky. Rovněž jsem sbíral odezvu ohledně celkového vzhledu aplikace a nechal uživatele navrhovat změny ohledně barevné palety a vzhledu tlačítek včetně hrací plochy.

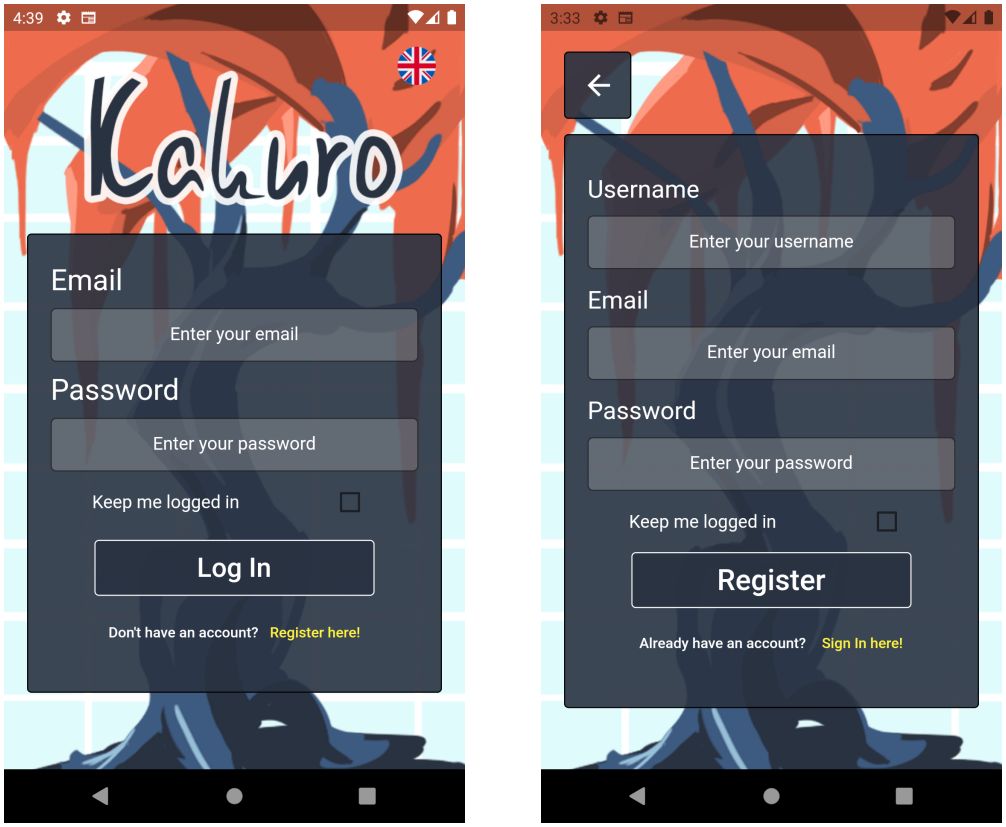
Se špatným ohlasem se setkal především způsob zadávání číslic do herních polí. Podle testovacích uživatelů byla odezva a samotný výběr velmi pomalý a spíše rušivý. Po této odezvě jsem se rozhodl nahradit selektor statickým číselníkem ve spodní části obrazovky v nadcházejícím vývoji aplikace.

3.6 Revize návrhu a nové funkce

Po konzultaci s uživateli jsem již měl ucelenější představu o tom, jak by výsledná aplikace mohla vypadat a jaké další funkce by uživatelé ocenili. Rozhodl jsem se tedy provést velkou revizi návrhu a přemýšlet o zajímavějším designu celé aplikace. Začal jsem rovněž pracovat na dalších funkcích, které však nebyly nezbytné pro první pilotní testování, jako je například registrace a přihlašovaní uživatelů nebo uživatelské nastavení účtu.

Přihlášení a registrace uživatelů

Jednou z potřebných a základních funkcí je evidence uživatelských účtů. Přihlašovací stránka, zobrazená na obrázku 3.5a, je první stránkou, se kterou se uživatel při stažení aplikace setká. Pro přihlášení je vyžadována emailová adresa a heslo, pro samotnou registraci však musí uživatel zadat i svoji přezdívku. Po registraci dojde ihned k přihlášení do samotné aplikace, kterou lze automaticky vynutit po každém jejím spuštění zakliknutím přepínače „Keep me logged in“.



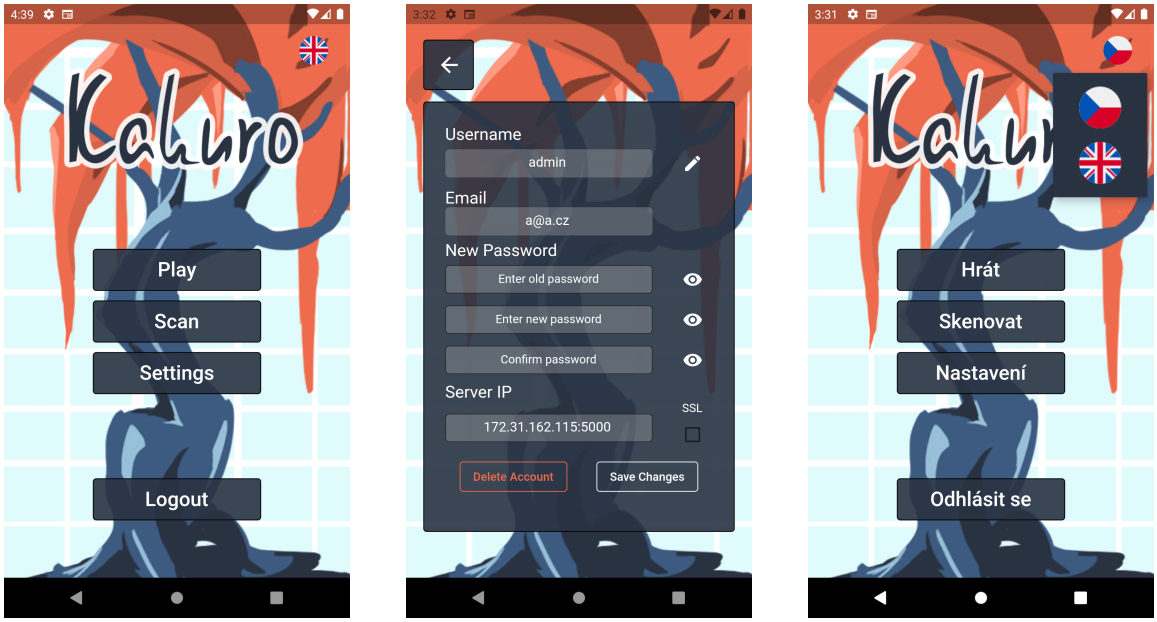
(a) Přihlašovací stránka (b) Registrační stránka

Obrázek 3.5: Přihlašovací a registrační stránka aplikace

Uživatelské nastavení a podpora více jazyků

Samozřejmostí je rovněž základní správa uživatelského účtu, ke které se lze dostat z hlavní nabídky tlačítkem „Settings“. V samotném nastavení lze provést změnu uživatelského jména a hesla, případně smazat účet z databáze. Stránku nastavení lze vidět na obrázku 3.6b.

Pro větší dosah aplikace jsem se rozhodl přidat vícejazyčnou podporu. Nutnost vícejazyčné podpory také vyžaduje samotné testování na uživatelích české národnosti v mém okolí, kteří ne vždy ovládají anglický jazyk. Samotný jazyk lze změnit v hlavní nabídce nebo na přihlašovací obrazovce kliknutím na ikonu vlajky státu, ke kterému je daný jazyk přidružen. V aktuální verzi je možné přepínat mezi češtinou a angličtinou, není však problém v budoucnu přidat i jiné jazyky. Nabídka změny lokalizace je vyobrazena na obrázku 3.6c.

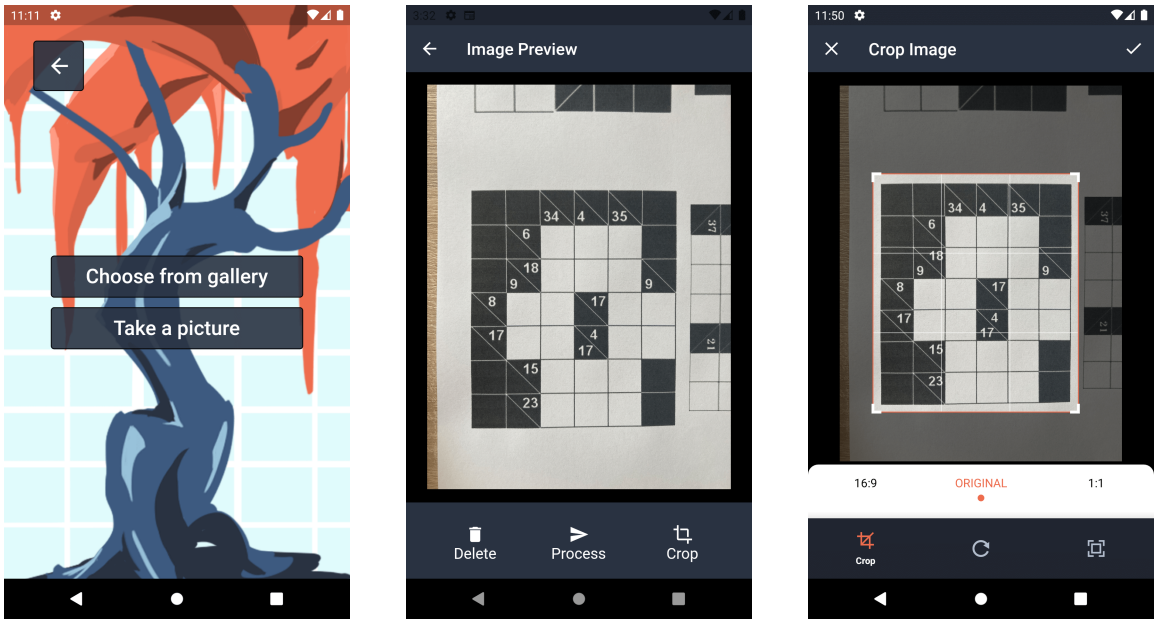


(a) Hlavní menu (b) Nastavení (c) Podporované jazyky

Obrázek 3.6: Stránky hlavního menu, nastavení a možnost výběru jazyka

Rozšíření funkce skenování her

Změnami prošla také sekce skenování her. Při náhledu vybraného či vyfotografovaného snímku je možné jej přímo v aplikaci oříznout, přičemž cílem této funkce je dosáhnout lepších výsledků skenování, eliminací okolních rušivých elementů, které by mohly způsobit chyby při detekci mřížkové struktury. Jakmile je snímek připraven, tlačítkem „Process“ dojde k jeho odeslání na server, který jej zpracuje a vrátí aplikaci data potřebná pro sestavení virtuálního hracího pole, na které uživatele následně přesměruje. Nové prostředí náhledu snímku je možné vidět na obrázku 3.7b, funkci ořezu pak na 3.7c.



(a) Menu skenování (b) Náhled obrázku (c) Ořez obrázku

Obrázek 3.7: Možnosti skenování a nové funkce náhledu obrázku

Výběr her a jejich sdílení mezi uživateli

Nabídka výběru her byla rozšířena o nové funkce, a to především o prohlížení dostupných her ve sdílené databázi, filtrování podle obtížnosti a sdílení naskenovaných her s ostatními uživateli. Jednotlivé záznamy jsou vizuálně interpretovány jako dlaždice, které lze expandovat a zpřístupnit tak operace, jenž může uživatel nad jednotlivými záznamy provádět.

Přepínači s názvy „Storage“ a „Local“ lze vybírat mezi lokálně uloženými a sdílenými hrami. Každý lokálně uložený záznam zobrazuje informace o obtížnosti, celkově odehraný čas a datum posledního spuštění. Nad jednotlivými položkami lze provádět základní operace, mezi které patří mazání, sdílení a přejmenování záznamu. Tlačítko „Play“ pak přesune uživatele do samotné herní části aplikace. Sdílení her může uživatel provést prostřednictvím tlačítka „Share“, sdílet lze však pouze uživatelem naskenované hry, pro které aplikace našla validní řešení. Všechny sdílené hry jsou následně viditelné v nabídce na obrázku 3.8b. Pro záznamy ve sdílené databázi lze vytvořit náhled hrací plochy a případně je uložit na lokální úložiště. Každá stažená položka je následně patřičně vizuálně označena, což vede k jasnému odlišení stažených her od vlastnoručně naskenovaných.

(a) Lokálně uložené hry (b) Sdílené hry

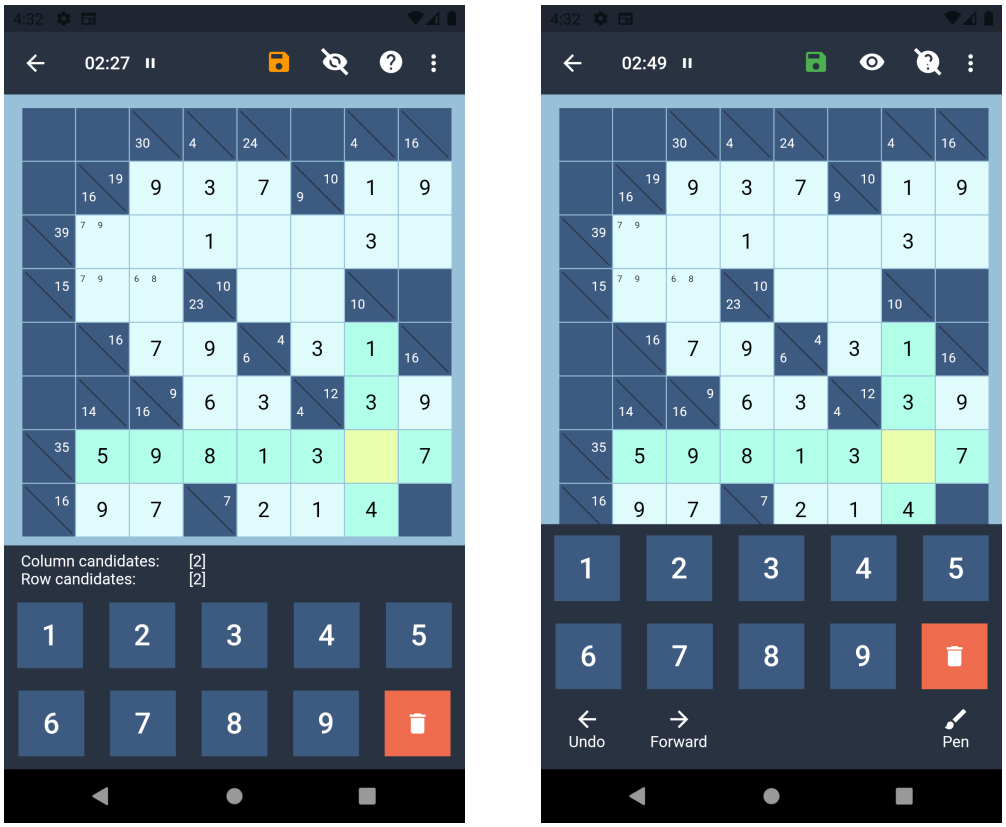
Obrázek 3.8: Rozhraní výběru her

Hrací plocha a její funkce

Samotné herní prostředí prošlo velkými změnami a vylepšeními. Kromě přepracování celkového vzhledu herní plochy byly přidány i nové funkce pro lepší uživatelkou zkušenost, jako zvýraznění hracího pole po jeho vybrání, časovač, nápověda ve formě výpisu kandidátů vybraného pole a přepínání mezi módy „tužky“ pro zapisování možných kandidátů nanečisto a „pera“ pro zadávání konečných vstupů.

Přidána byla i funkce automatického ukládání, které probíhá každých 20 vteřin, jestliže byla zaznamenána změna na herní ploše. Stav uložení je indikován ikonkou diskety na horní liště, přičemž kliknutím na tuto ikonu lze vynutit uložení také manuálně.

Lišty nápovědy a pomocných funkcí lze libovolně skrývat dle potřeby pomocí ikon v horní nabídce. Uživatelé si tak mohou rozšířit prostor pro hrací plochu, jestliže tyto funkce nevyžadují. Obrazovky herních částí, kdy každá zobrazuje právě jednu z pomocných lišt lze vidět na obrázku 3.9.



(a) Prostředí s nápovědou (b) Prostředí s lištou nástrojů

Obrázek 3.9: Ukázka herní části aplikace

Kapitola 4

Implementace aplikace

Následující kapitola se podrobněji věnuje implementační části vývoje aplikace. Jak již bylo naznačeno v kapitole 3 věnující se návrhu, tato aplikace se skládá celkově ze dvou separátních částí, kterými jsou server a samotná mobilní aplikace. V jednotlivých podkapitolách proto budou popsány obě části zvlášť, neboť se jedná o zcela odlišné a do jisté míry nezávislé celky.

4.1 Server

Úkolem serveru v kontextu výsledné aplikace je především zpracování snímků her, které si uživatelé chtějí naskenovat z fyzického média. Výstupem je pak virtuální reprezentace dané hry ve formátu vhodném pro přenos po síti, v tomto případě tedy JSON. Server je vyvíjen a testován vůči platformám Linux a macOS.

Použité technologie

Implementace serveru je realizována pomocí programovacího jazyka `Python` a frameworku `Flask`[1], který poskytuje minimální implementaci webového serveru. Pro zpracování snímků byly využity volně dostupné knihovny `OpenCV` poskytující velké množství nástrojů pro práci s obrazem, `TensorFlow`[2] umožňující návrh a trénování neuronových sítí a `numpy`[3] pro úpravu a manipulaci dat. Jako produkční WSGI[4] na platformu UNIX je využívána implementace `gunicorn`[5].

Popis zpracování snímků

Po spuštění server čeká na příchozí požadavky. Zpracování snímků obstarává koncový bod `/image/process`, který reaguje na zprávy protokolu `HTTP`, konkrétně metody `POST`. Typ obsahu `Content-Type` v hlavičce zprávy je očekáván jako `multipart/form-data`. Samotná binární data obrázku pak musí být umístěna v poli formuláře s označením `image`.

[1] https://flask.palletsprojects.com/en/1.1.x/
[2] https://www.tensorflow.org/
[3] https://numpy.org/
[4] Web Server Gateway Interface
[5] https://gunicorn.org/

Po přijetí požadavku jsou data uložena do paměti, konvertována do datové struktury typu `numpy Array` a následně dekódována do maticové struktury pomocí funkce `imdecode`, se kterou následně pracují i všechny ostatní funkce, jež jsou součástí knihovny `OpenCV`. Dekódovaný obrázek je následně nutné převést na stupně šedi a snížit jeho rozlišení takovým způsobem, aby se šířka obrázku pohybovala přibližně kolem hodnoty 500 pixelů. Tuto hodnotu jsem určil na základě experimentování s různými parametry rozlišení, přičemž právě taková hodnota je kompromisem pro co nejlepší poměr rychlosti a přesnosti detekce mřížkové struktury. Původní obrázek převedený na stupně šedi však zůstává zachovaný v paměti pro účely dalšího zpracování.

Jakmile je obrázek s upraveným rozlišením z předešlého kroku připravený, je možné v něm detekovat jednotlivé linie. Tuto detekci lze provést funkcí `detect_lines`, která nejdříve provede detekci hran pomocí Cannyho hranového detektoru a následně provede pravděpodobnostní Houghovu transformaci, jejíž výstupem je seznam všech nalezených přímek v obraze, mající formát souřadnic počátečního a koncového bodu `[x1, y1, x2, y2]`. Tyto linie jsou převedeny na objekt `Line` a dochází k přepočítání souřadnic takovým způsobem, aby odpovídaly originálnímu rozlišení snímku. Vytvořené objekty jsou dále roztříděny do dvou seznamů podle vertikální a horizontální orientace.

Dalším krokem zpracování je detekování mřížky na základě získaných linií. Pro každou horizontální a vertikální linii jsou vypočítány body průsečíků, přičemž dochází k vynechání bodů, které se nacházejí duplicitně ve své těsné blízkosti. Tento jev typicky vzniká výskytem linií, které celkově tvoří jednu větší dělící čáru, jež však byla detekována jako soustava několika různých segmentů. Takto nalezené body jsou následně seřazeny do matice dle své pozice funkcí `sort_points_to_grid` a vzniká tak struktura mřížky.

Po získání matice bodů lze sestavit objekt `Layout`, který reprezentuje virtuální podobu naskenované hry. Z matice jsou postupně vybírány čtveřice bodů, které jako celek tvoří čtverec, na základě kterého je proveden výřez v originálním šedém obraze, tedy segmentu hracího pole. Pro každý segment je spočítán jeho řádkový a sloupcový index v rámci mřížky, jež jsou dále předány jako celek klasifikátoru.

Třída `Classifier` reprezentuje klasifikátor, který dokáže nad přijatým výřezem snímku provést jeho klasifikaci a přetvořit jej na objekt `Field`. Instance třídy `Classifier` umí provádět následující klasifikace:

- Klasifikace typu pole
- Klasifikace číslic v hracích polích
- Extrakce a klasifikace kontrolních součtů

Jako první je vždy nutné provést klasifikaci typu pole. Ta se provádí pomocí natrénované neuronové sítě, která je specializována na rozlišování herních, výplňových a podmínkových polí. V návaznosti na rozhodnutí se odvíjí další kroky zpracování. V případě detekce výplňového pole nese příslušný objekt `Field` pouze informaci o tomto typu, herní pole je dále předáno neuronové síti pro rozpoznávání ručně psaných číslic.

Nejnáročnějším zpracováním prochází podmínkové pole, nesoucí informace o kontrolních součtech pro příslušný řádek nebo sloupec. Každé takové políčko může obsahovat až 2 součty, přičemž se často jedná o dvoumístné číslice, jejichž členy je nejdříve nutné od sebe oddělit a následně předat neuronové síti pro klasifikaci strojově psaných číslic. Právě v této části je žádoucí, aby zpracovávaný segment disponoval co možná největším rozlišením, neboť jednotlivé číslice pak mohou téměř splynout do sebe a znemožnit jejich separaci. Pro podobné situace je nad celým výřezem provedena konvoluce s jádrem, které je vyobrazeno

na obrázku 4.1. Tento filtr má za následek zvýraznění hran objektů v obraze a potlačení šumu v jejich těsné blízkosti. Pro následnou detekci jednotlivých číslic je využita funkce `findContours` knihovny `OpenCV`, jejíž výstupem jsou pozice ohraničujících rámečků nalezených objektů. Z původního obrazu jsou následně provedeny výřezy těchto oblastí, jež jsou předány jako vstup neuronové síti rozpoznávající strojově psané číslice.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Obrázek 4.1: Konvoluční jádro filtru pro zvýraznění hran

Nad výslednými objekty třídy `Field` lze dále provést serializaci do formátu `JSON` a zpracovaná data odeslat zpátky klientské aplikaci. Jakým způsobem se tato data zpracovávají v mobilní aplikaci bude popsáno v kapitole 4.2.

Optimalizace rychlosti zpracování snímků

Výše uvedený popis zpracování snímků dosahuje uspokojivých výsledků, avšak rychlost procesu není zcela optimální. Jako referenci zde uvádím tabulku 4.1, která popisuje rychlost zpracování u různých velikostí hracích ploch. Je zde však nutné upřesnit, že doba zpracování se výrazně odvíjí od distribuce typů jednotlivých polí a nejsou zde zahrnuty další faktory, jako například doba přenosu obrázku po síti a režie serveru. Výsledná doba se tedy může při ostrém nasazení navíc prodloužit.

Šířka	Výška	Celkem polí	Doba zpracování (s)
17	16	272	18.798723
12	10	120	13.194325
8	8	64	7.231423

Tabulka 4.1: Tabulka závislosti rychlosti zpracování na velikosti hrací plochy

Pro řešení problematiky optimalizace jsem se rozhodl rozložit zpracování segmentů do několika samostatných jednotek a využít tak paralelismu. K implementaci této myšlenky jsem využil třídy `Process` a `JoinableQueue`, které jsou součástí modulu `multiprocessing`.

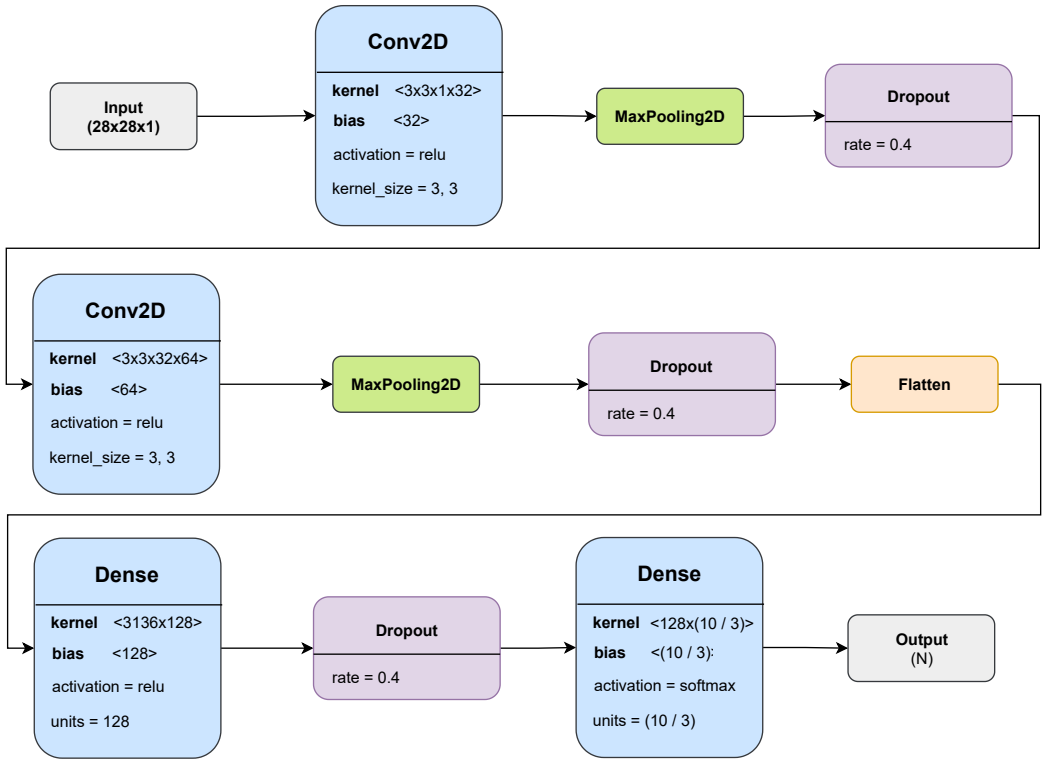
Třídu `Classifier` jsem přepracoval takovým způsobem, aby mohla využít dědičnosti třídy `Process` a být tak spuštěna jako samostatný podproces. Pro distribuci práce jsem využil `JoinableQueue` a vytvořit tak 2 fronty `processing_queue` a `results_queue`. Hlavní proces obsluhující požadavek klienta pak provede detekci mřížkové struktury a vytvoří výřezy všech segmentů přítomných na hrací ploše, kterými následně naplní frontu `processing_queue`. Procesy třídy `Classifier` bežící jako `daemon` hlavního procesu serveru pak naslouchají na této frontě a postupně výřezy polí zpracovávají. Výsledky v podobě objektů `Field` jsou průběžně vkládány do fronty `results_queue`. Jakmile dojde k zpracování všech polí, dochází k serializaci výsledků a data jsou odeslána klientovi. Výsledné časy zpracování po této optimalizaci jsou uvedeny v tabulce 4.2.

Šířka	Výška	Celkem polí	Doba zpracování (s)
17	16	272	5.8791420
12	10	120	3.3994240
8	8	64	1.8193192

Tabulka 4.2: Závislosti rychlosti paralelního zpracování na velikosti hrací plochy

Konvoluční neuronové sítě

Nedílnou součástí aplikace jsou konvoluční neuronové sítě. Jak již bylo řečeno v předchozích kapitolách, celkově byly natrénovány 3 neuronové sítě na různých datových sadách, přičemž všechny sítě sdílí společnou konfiguraci vrstev, kterou lze vidět na obrázku 4.2.



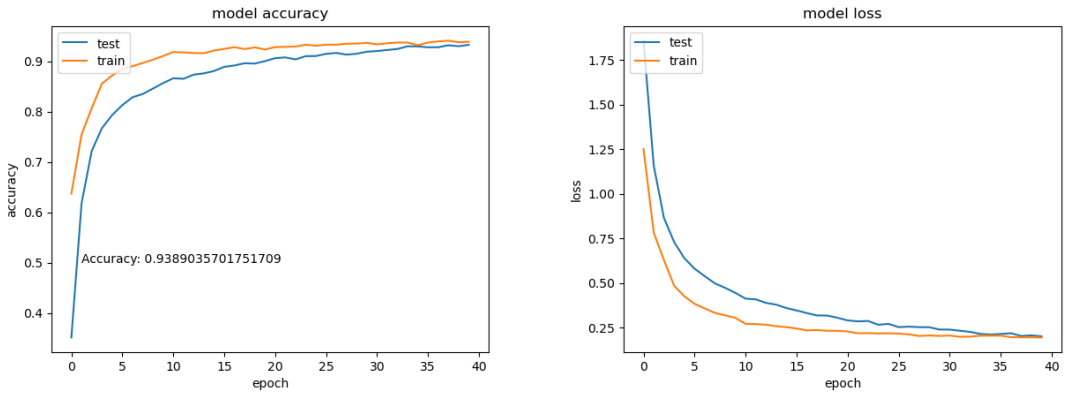
Obrázek 4.2: Konfigurace konvoluční neuronové sítě

Pro trénování sítí jsem využil vlastní datové sady, jež jsem pro účely této práce vytvořil. Jednotlivé sady jsou popsány v tabulce 4.3. Abych docílil efektivní extrakce ručně psaných číslic, vytvořil jsem jednoduchý formulář, který lze najít v příloze A. O vyplnění formuláře jsem požádal více osob a získal tak různé styly písma. Pro automatizaci celého procesu extrakce jsem využil poznatků získaných při vývoji systému pro detekci mřížkové struktury a vytvořil skript, který z naskenovaného formuláře získá políčka s číslicemi a následně je roztřídí. Kompletní datová sada ručně psaných číslic pak vznikla sloučením se vzorky, které nasbíral kolega Lazorík Juraj, zabývající se obdobnou problematikou.

Všechny 3 natrénované modely uvádí přesnost větší než 90%. Pro demonstraci zde uvádím grafy přesnosti a loss funkce z průběhu trénování modelu pro rozpoznávání ručně psaných číslic, které lze vidět na obrázku 4.3.

Datová sada	Počet tříd	Rozlišení	Celkem vzorků
Ručně psané číslice	10	28x28	15 134
Strojově psané číslice	10	28x28	520
Typy polí herní plochy	4	28x28	661

Tabulka 4.3: Popis jednotlivých datových sad



(a) Graf přesnosti

(b) Graf loss funkce

Obrázek 4.3: Grafy přesnosti a loss funkce modelu pro klasifikaci ručně psaných číslic

4.2 Mobilní aplikace

Použité technologie

Aplikace byla napsána v programovacím jazyce `Dart` a s pomocí frameworku `Flutter`, který je na tomto jazyce rovněž postaven. Samotný framework poskytuje multiplatformní podporu, díky čemuž je aplikace plně funkční a otestovaná jak na zařízení s operačním systémem Android, tak iOS. Značnou součást řešení tvoří komunitní balíčky, které jsou volně dostupné a poskytují celou škálu funkcionalit, díky kterým mohla tato aplikace vzniknout. Mezi významné použité balíčky patří například:

- provider (stavový management aplikace)
- camera (rozhraní pro přístup k fotoaparátu)
- image_picker (výběr snímků z galerie zařízení)
- image_cropper (přístup k rozhraní nativního ořezu snímků)
- sqflite (lokální databáze)
- cloud_firestore (vzdálená databáze služby firebase)
- firebase_auth (autentizační rozhraní služby firebase)

Hlavní motivací výběru technologie Flutter oproti jiným konkurenčním řešením je především již moje minulá pozitivní zkušenost s touto technologií, velmi intuitivní deklarativní přístup k popisu uživatelského rozhraní a celá řada naučných materiálů vytvořených jak širokou komunitou, tak i samotnými vývojáři.

Struktura projektu

Samotný projekt je rozvržen do částí takovým způsobem, aby byla oddělena aplikační logika od uživatelského rozhraní. Zdrojové kódy jsou tedy rozděleny do modulů nesoucí názvy `ui`, `providers`, `services` a `models`. Modul `ui` obsahuje definice všech pohledů a vlastních widgetů, přičemž tyto widgety pak získávají data pomocí tříd obsažených v modulu `providers`. Konkrétní implementace služeb autentizace a databází lze pak nalézt v `services` a datové modely v `models`. Struktura velmi připomíná návrhový vzor MVC[6], avšak nejedná se zcela o jeho tradiční implementaci.

Autentizace

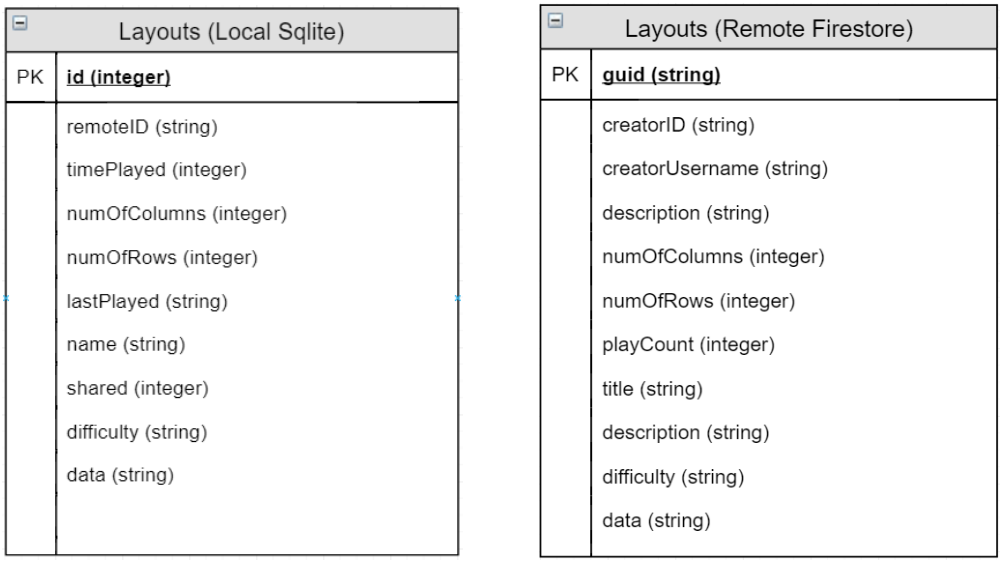
Pro autentizaci a registraci uživatelů jsem využil služby `Firebase`[7] od společnosti Google, která poskytuje velmi jednoduché a efektivní rozhraní pro tyto účely. Aby bylo možné službu použít na mobilním zařízení, je nejdříve potřeba aplikaci do služby zaregistrovat a vložit do ní potřebné konfigurační soubory. Pomocí balíčku `firebase_auth` lze pak využívat autentizační rozhraní přímo v aplikaci pomocí dostupných metod.

Databáze

Naskenované a stažené hry je nutné ukládat na perzistentní uložiště, aby se uživatelé ke svým řešením mohli kdykoliv vrátit. K tomuto účelu jsem využil balíček `sqlite` obsahující implementaci jednoduché relační databáze s obdobným názvem SQLite[8], ve které lze vytvořit tabulku a ukládat do ní data jednotlivých her. Na obrázku 4.4a lze vidět entitu hry, která slouží jako předloha pro položky lokální databáze.

Sdílené hry jsou posílány a ukládány v kolekcích na cloudové službě `Firestore`, která je součástí celku služeb Google Firebase. Každý záznam uložený v kolekci sdílených her má strukturu viditelnou na obrázku 4.4b. Pro komunikaci s databází slouží balíček `cloud_firestore`, který disponuje rozhraním pro komunikaci se vzdálenými servery.

[6] Model-View-Controller
[7] https://firebase.google.com/
[8] https://www.sqlite.org/index.html



(a) Entita hry v lokální databázi (b) Entita hry ve vzdálené databázi

Obrázek 4.4: Databázové entity popsané jazykem UML

Management stavu aplikace

Velkou otázkou každé technologie zabývající se vývojem aplikací je stavový management. Jak již bylo nastíněno v kapitole 1.6, stavebním kamenem aplikace postavené na frameworku Flutter, jsou stavové widgety, které si po celou dobu svojí existence uchovávají stavové proměnné, které lze za běhu aplikace měnit a vynutit tak překreslení widgetu s jinými daty. Tento přístup je však v kontextu větších aplikací prakticky neudržitelný, neboť představuje implementaci logiky v rámci jednotlivých widgetů a činí tak celý kód těžko čitelný a neudržitelný.

Odpovědí jsou tak architektury a metody, které nabízí různá řešení, jak se s tímto problémem vypořádat. Jedním z nich je balíček `provider`, jehož koncepty jsem využil ve své výsledné aplikaci. Řešení spočívá v pozvednutí stavu do větších logických celků, které jsou nad rámcem celé aplikace. Tyto části jsou dále označovány jako `Provider`, jež lze následně zaregistrovat v kořeni stromu widgetů, čímž se stávají dostupnými pro všechny ostatní podstromy. Jednotlivé uzly pak mohou přistupovat k datům a provolávat metody těchto poskytovatelů služeb. Důležitou vlastností této architektury je možnost podstromů přihlásit se k odběru změn, které u poskytovatelů nastávají. Všechny takto přihlášené uzly jsou v případě změn informovány a je vynuceno jejich překreslení na obrazovce.

V samotné aplikaci je dostupných několik takových poskytovatelů, přičemž všichni se nacházejí v adresáři projektu `providers`. Řízení autentizace a lokalizace pak například obstarává třída `AppStateProvider`, časovače spravuje `TimerProvider`, data z databází poskytuje `DataProvider` apod.

Kapitola 5

Uživatelské testování a výsledky

Nezbytnou součástí vývoje jakékoliv uživatelské aplikace je její řádné otestování na skupině testovacích uživatelů. Cílem testování bylo především získat pocity a dojmy z aplikace od jiných lidí a odhalit tak různé nedokonalosti v návrhu uživatelského rozhraní, případně funkcí aplikace. Vzhledem k probíhající pandemii onemocnění COVID-19 byly možnosti fyzického testování omezené, podařilo se mi však oslovit 15 uživatelů, kteří se mnou aplikaci otestovali. Samotné testování probíhalo přímo na zařízeních samotných uživatelů, měl jsem tak možnost pozorovat, jakým způsobem se chová GUI na zařízeních různé velikosti. Pro jednoduchou instalaci aplikace jsem nahrál testovací alpha verzi na službu Google Play, odkud ji bylo možné stáhnout na zařízení s operačním systémem Android. Aby bylo možné data z průběhu testování lépe analyzovat, vytvořil jsem rovněž stručný dotazník, který lze nalézt v příloze B.

5.1 Průběh testování

Testování jsem rozdělil na dílčí části, kterými každý z uživatelů musel postupně projít. Konkrétně jsem se zaměřil na následující funkce:

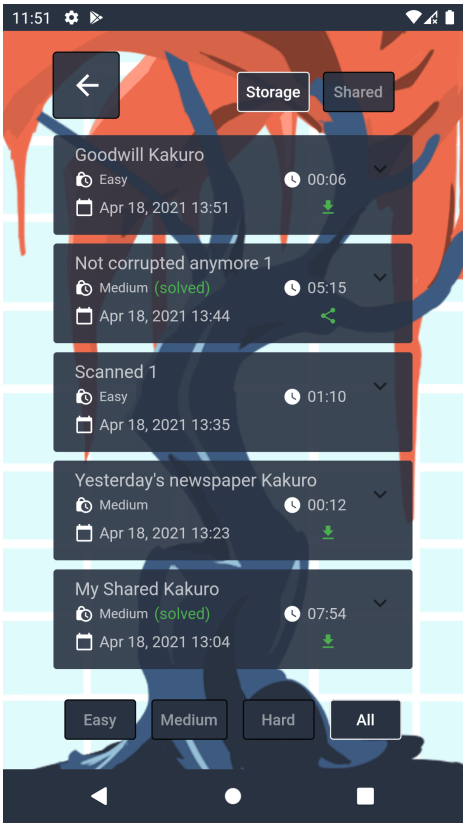
- Registrace a přihlášení
- Navigace v hlavní nabídce
- Výběr a stahování sdílených her
- Ovládání hrací plochy
- Proces naskenování hry
- Využití editačního módu k opravě naskenované hry

Po dokončení testování aplikace jsem uživatele požádal o vyplnění dotazníku, ve kterém mohli jednotlivé části ohodnotit a případně zapsat své nápady a připomínky.

5.2 Zpětná vazba uživatelů a úpravy aplikace

Zpětná vazba od uživatelů byla velkým přínosem, neboť se podařilo odhalit velké množství chyb a nedodělků, které narušovaly požitek při používání aplikace. Chyby byly z většiny spojené s nevhodným škálováním widgetů vůči velikosti obrazovky, což způsobovalo například rozpad textu nebo kolizi prvků. Všechny nalezené nedostatky jsem následně odstranil.

S kritikou se setkala především funkce skenování. Nutnost provedení výřezu hry v obraze se ukázala jako velmi nekomfortní a nespolehlivá. V důsledku nedokonalých ořezů systém často neodpovídal a nebyl schopen hry spolehlivě naskenovat. Pro některé uživatele tak bylo skenování her velmi obtížné. Na základě těchto okolností jsem se rozhodl přepracovat zpracování obrázků na serveru takovým způsobem, aby systém automaticky nalezl hrací plochu v obraze a provedl tak co nejvhodnější výřez, který následně zpracuje. Tato úprava značně navýšila úspěšnost skenování a úroveň uživatelského komfortu.



Obrázek 5.1: GUI výběru her po úpravách

Často opakujícím se návrhem byla indikace již vyřešených her a řazení her podle data posledního spuštění, případně stažení. Tyto návrhy jsem se následně rozhodl implementovat. K jednotlivým položkám, které byly již kompletně a správně vyřešeny, přibyl indikátor se značkou „`(solved)`“. Jednotlivé položky lokálně uložených her se pak ve výběru zobrazují sestupně podle data posledního spuštění. Uživatel tak vždy vidí na prvních příčkách naposledy odehrané, nebo právě stažené hry. Provedené úpravy lze vidět na obrázku 5.1.

Obecně však byla aplikace hodnocena převážně kladně. Nezaznamenal jsem žádné výhrady vůči barevné paletě a designu prvků. Ovládání aplikace včetně herní části působilo na uživatele velmi intuitivně a plynule.

Kapitola 6

Závěr

Výsledkem této práce je mobilní a serverová aplikace. Mobilní aplikace je vyvinuta a plně funkční na platformách Android a iOS, serverová část je pak určena pro běh na operačním systému Linux a uvnitř Docker kontejnerů.

Aplikace umožňuje uživatelům naskenovat hru Kakuro například z novin a poskytnout asistenci s jejím dohráním. Aplikaci jsem rovněž rozšířil o možnost sdílet hry s ostatními uživateli, kteří si je následně mohou stáhnout a vyřešit. Uživatelé tak nebudou mít nouzi o nové hry, i když zrovna nemají žádné hry k naskenování. Aplikace je závislá na internetovém připojení kvůli registraci uživatelů, skenování a sdílení her, lze ji však využít rovněž v režimu offline, ve kterém jsou k dispozici všechny lokálně uložené hry. Aplikace je dostupná na službě Google Play[1].

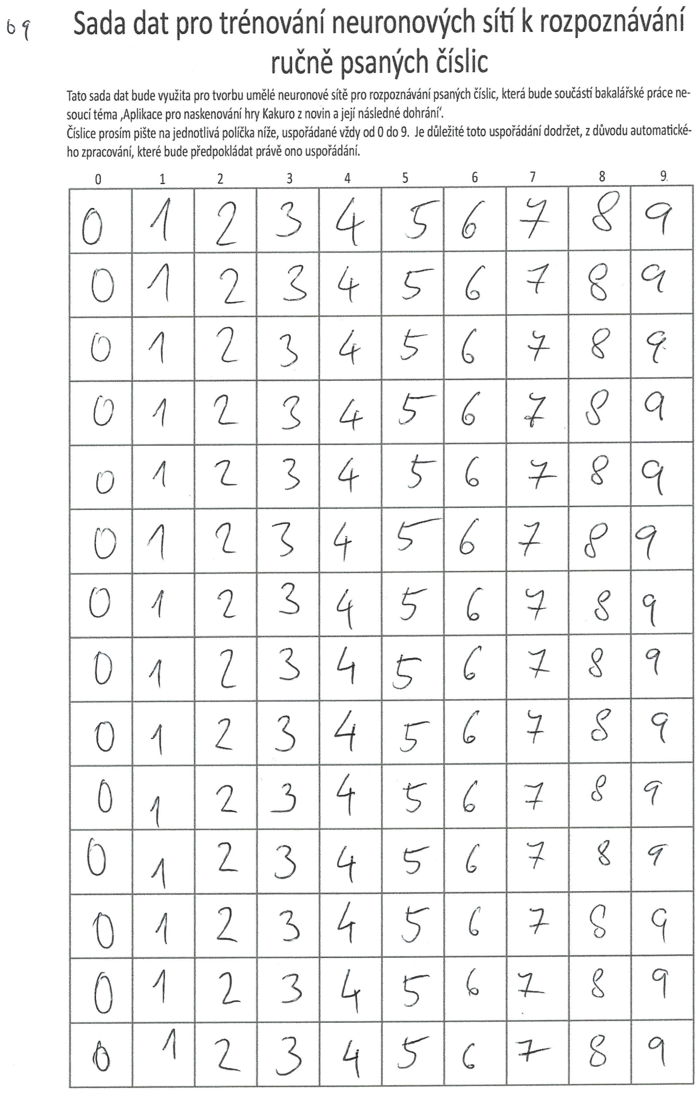
Práce mi přinesla mnoho zkušeností, neboť jsem se seznámil s problematikou neuronových sítí, tvorbou webových API, procesem vývoje aplikací a prošel všemi etapami od průzkumu existujících řešení, návrhu, implementace a uživatelského testování, až po samotné nasazení aplikací na cloudové služby. Prohloubil jsem si rovněž své znalosti v oblastech jazyka Python, Dart, frameworků Flask, Flutter, verzovacího systému Git a kontejnerech služby Docker.

V budoucnu bych chtěl aplikaci dále zdokonalit a rozšířit o nové funkce. Hlavním cílem je zpřístupnit funkci skenování i v offline režimu, tedy implementovat proces zpracování snímků pro obě platformy zvlášť nativně. V závislosti na úspěchu aplikace na Google Play lze rovněž aplikaci v budoucnu zpřístupnit i na službě App Store pro zařízení s operačním systémem iOS.

[1] https://play.google.com/store/apps/details?id=com.coderpp.kakuro_solver

Příloha A

Formulář pro extrakci ručně psaných číslic



69

Sada dat pro trénování neuronových sítí k rozpoznávání ručně psaných číslic

Tato sada dat bude využita pro tvorbu umělé neuronové sítě pro rozpoznávání psaných číslic, která bude součástí bakalářské práce nesoucí téma ‚Aplikace pro naskenování hry Kakuro z novin a její následné dohrání'.
Číslice prosím pište na jednotlivá políčka níže, uspořádané vždy od 0 do 9. Je důležité toto uspořádání dodržet, z důvodu automatického zpracování, které bude předpokládat právě ono uspořádání.

0	1	2	3	4	5	6	7	8	9.
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

Příloha B

Dotazník k testování aplikace

Dotazník k uživatelskému testování mobilní aplikace Kakuro Solver

*Povinné pole

1. Setkali jste se někdy se hrou Kakuro? *

   Označte jen jednu elipsu.

   - Ano, tuto hru hraji ve svém volném čase.
   - Ano, ale nikdy jsem ji aktivně nehrál/a.
   - Ne, o této hře slyším poprvé.

2. Celkový vzhled a barevná paleta aplikace *

   Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
nelíbivá												líbivá

3. Navigace v rámci menu aplikace k jednotlivým funkcím *

   Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
velmi obtížná												zcela intuitivní

4. Odezva aplikace ve formě upozornění a oznámení na různé netypické a chybové stavy *

   Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
špatná												výborná

5. Ovládání a funkce menu pro výběr her jsem *

   Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
zcela nepochopil/a												pochopil/a bez problému

6. Jednotlivé funkce považuji za: *

Označte jen jednu elipsu na každém řádku.

	Užitečné	Zbytečné	Překážející
Sdílení her	◯	◯	◯
Filtrování podle obtížnosti	◯	◯	◯
Pojmenování her	◯	◯	◯

7. Které z možností filtrování a řazení her podle Vás v aplikaci u sdílených her chybí? *

Zaškrtněte všechny platné možnosti.

- ☐ Žádné další možnosti bych nepřidával/a
- ☐ Řazení podle počtu stažení
- ☐ Vyhledávání podle uživatelského jména
- Jiné: ☐ _____

8. Které z možností filtrování a řazení her podle Vás v aplikaci u uložených her chybí? *

Zaškrtněte všechny platné možnosti.

- ☐ Žádné další možnosti bych nepřidával/a
- ☐ Řazení podle celkového času stráveného řešením
- ☐ Řazení podle data od posledního hraní
- Jiné: ☐ _____

9. Ovládání a funkce hrací obrazovky jsem *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
zcela nepochopil/a	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	pochopil/a bez problému

10. Rozložení ovládacích prvků na herní obrazovce je *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
zcela nepřehledné	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	přehledné a jasné

11. Jednotlivé funkce herní části považuji za: *

Označte jen jednu elipsu na každém řádku.

	Užitečné	Zbytečné	Překážející
Časovač			
Návrat o krok zpět			
Postup o krok vpřed			
Vyřešit hru			
Mód pera/tužky			
Nápověda s kandidáty			
Zvýrazňování vybraných polí			

12. Ovládací prvky při skenování her jsem *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
zcela nepochopil/a												pochopil/a bez problému

13. Přehlednost a funkčnost editačního módu hrací plochy při neúspěšné detekci některých prvků je *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
špatná												výborná

14. Jak hodnotíte celkovou úspěšnost detekce a klasifikace naskenovaných her? *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
nefunkční												funkční

15. Jak byste ohodnotili aplikaci jako celek? *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
špatné												výborné

16. Zde můžete napsat jakékoliv připomínky *

Literatura

[1] Boukhary, S. a Colmenares, E. A Clean Approach to Flutter Development through the Flutter Clean Architecture Package. In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2019, s. 1115–1120. DOI: 10.1109/CSCI49370.2019.00211.

[2] Caterini, A. L. *Deep Neural Networks in a Mathematical Framework*. 2018. SpringerBriefs in Computer Science Ser. ISBN 9783319753041.

[3] Davies, R. P., Roach, P. A. a Perkins, S. Automation of the Solution of Kakuro Puzzles. In: *Research and Development in Intelligent Systems XXV: Proceedings of AI-2008, the Twenty-eighth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*. London: Springer London, 2009, s. 219–232. ISBN 9781848821705.

[4] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. ISBN 0262035618. Dostupné z: http://www.deeplearningbook.org.

[5] Gupta, S. *How to solve Kakuro* [online]. [vid. 2020-11-15]. Dostupné z: https://theory.tifr.res.in/~sgupta/kakuro/algo.html.

[6] Janovič, T. *Detekce očí v obrazech obličeje pomocí Houghovy transformace*. Brno, CZ, 2012. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=54849.

[7] Jeřábek, F. *Webová a mobilní aplikace pro zadávání a potvrzování úkolů*. Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/23112/.

[8] Kiryati, N., Eldar, Y. a Bruckstein, A. A probabilistic Hough transform. *Pattern Recognition*. 1991, sv. 24, č. 4, s. 303–316. DOI: 10.1016/0031-3203(91)90073-E. ISSN 0031-3203. Dostupné z: https://www.sciencedirect.com/science/article/pii/003132039190073E.

[9] Laganiére, R. *OpenCV 2 computer vision application programming cookbook : over 50 recepies to master this library of programming functions for real-time computer vision*. 1st ed. Brimmingham: Packt Publishing, 2011. Quick Answers to Common Problems. ISBN 978-1-84951-324-1.

[10] Moravec, Z. *Kontrola zobrazení textu ve formulářích*. Brno, CZ, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/19700/.

[11] OpenCV. *Canny Edge Detection* [online]. [vid. 2020-10-24]. Dostupné z: https://docs.opencv.org/master/da/d22/tutorial_py_canny.html.

[12] OpenCV. *Color conversions* [online]. [vid. 2020-10-22]. Dostupné z: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html.

[13] Reinfelder, L., Benenson, Z. a Gassmann, F. Differences between Android and iPhone Users in Their Security and Privacy Awareness. In:. Září 2014, s. 156–167. DOI: 10.1007/978-3-319-09770-1. ISBN 978-3-319-09769-5.

[14] Ruepp, O. a Holzer, M. The Computational Complexity of the Kakuro Puzzle, Revisited. In: *Fun with Algorithms: 5th International Conference, FUN 2010, Ischia, Italy, June 2-4, 2010. Proceedings*. Springer Berlin Heidelberg, 2010, sv. 6099, s. 319–330. Lecture Notes in Computer Science. ISBN 9783642131219.

[15] Španěl, M. a Beran, V. *Obrazové segmentační techniky*. Brno, CZ, 2006. Přehled existujících metod. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: http://www.fit.vutbr.cz/~spanel/segmentace/.