SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta elektrotechniky a informatiky

Evidenčné číslo: FEI-16605-120403

Program pre vyhodnotenie zásahov pri športovej streľbe

Bakalárska práca

2024 **Nicolas Droppa**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta elektrotechniky a informatiky

Evidenčné číslo: FEI-16605-120403

Program pre vyhodnotenie zásahov pri športovej streľbe

Bakalárska práca

Študijný program: Aplikovaná informatika
Študijný odbor: Informatika
Školiace pracovisko: Ústav elektroniky a fotoniky
Školiteľ: doc. Ing. Miroslav Hagara, PhD.

2024 **Nicolas Droppa**

Slovenská technická univerzita v Bratislave
Ústav elektroniky a fotoniky

Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025
Evidenčné číslo: FEI-16605-120403

ZADANIE BAKALÁRSKEJ PRÁCE

Študent:	**Nicolas Droppa**
ID študenta:	120403
Študijný program:	aplikovaná informatika
Študijný odbor:	informatika
Vedúci práce:	doc. Ing. Miroslav Hagara, PhD.
Vedúci pracoviska:	doc. Ing. Anton Kuzma, PhD.
Miesto vypracovania:	Ústav elektroniky a fotoniky (FEI)

Názov práce: **Program pre vyhodnotenie zásahov pri športovej streľbe**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom bakalárskej práce je vytvoriť program, ktorý by umožnil interaktívnym spôsobom vyhodnotiť výsledky športovej streľby z fotografie terča.

Úlohy:
1. Naštudujte spôsoby vyhodnocovania športovej streľby zo vzduchovej pištole počas súťaže a počas tréningu.
2. Zvoľte hodnotiace parametre športovej streľby počas tréningu, ktoré bude vytvorený program vyhodnocovať z fotografie terča.
3. Určite podmienky, ktoré musí fotografia terča spĺňať, aby z nej bolo možné vyhodnotiť výsledky športovej streľby z fotografie terča.
4. Vytvorte program, ktorý umožní automaticky alebo manuálne označiť na fotografii miesto zásahu a pre označené miesta zásahov vypočíta zvolené hodnotiace parametre športovej streľby.
5. Vytvorený program odskúšajte s rôznymi fotografiami terčov.

Zoznam odbornej literatúry:

1. GONZALES, Rafael C.; WOODS, Richard E. *Digital Image Processing.* New York : Pearson Education Limited, 2018. 1019 s. ISBN 978-1-292-22304-9.
2. PRATT, William K. *Digital Image Processing. 4th ed.* Hoboken, New Jersey: John Wiley & Sons, 2007. 808 s. ISBN 978-0-471-76777-0.
3. VALKOVIČ, Ladislav; WINDISCHBERGER, Christian. Method for geometric distortion correction in fMRI based on three echo planar phase images. *Measurement Science Review [elektronický zdroj],* 10. s. 116–119.
4. HREBIČÍK, Viliam; GARAN, Martin. *Návrh spracovania obrazu pre strojové videnie robotického systému.* Diplomová práca. 2017.

Termín odovzdania bakalárskej práce:	06. 06. 2025
Dátum schválenia zadania bakalárskej práce:	17. 04. 2025
Zadanie bakalárskej práce schválil:	prof. Dr. rer. nat. Martin Drozda – garant študijného programu

Poďakovanie

Touto cestou by som sa chcel poďakovať vedúcemu mojej bakalárskej práce doc. Ing. Miroslavovi Hagarovi, PhD. za výpomoc, cenné rady a konštruktívnu kritiku, ktoré mi pomohly pri vypracovaní tejto práce a posunutí sa v odbornej sfére.

Abstrakt

Cieľom bakalárskej práce je... Práca pozostáva z X kapitol. Prvá kapitola sa zameriava na... Druhá kapitola sa venuje... Posledná kapitola obsahuje...

Kľúčové slová

Detekcia, Perspektíva, Streľba

Abstract

The manual for students of the Faculty of Electrical Engineering and Information Technology at Slovak University of Technology in Bratislava provides advice and guidelines on how to approach the formal aspects of writing a final thesis for university studies. The document can also serve as a template for the thesis in the LaTeX typesetting system. It covers in detail the rules for typesetting mathematical equations, numbering floating objects, and referencing them. It also thoroughly addresses the method of citing external literary sources.

Keywords

Detection, Perspective, Shooting

Obsah

Zoznam značiek a skratiek

PWA – Progresívna Webová Aplikácia, z angl. *Progresive Web App*

2D – Dvojrozmerné, z angl. *two-dimensional*

3D – Trojrozmerné, z angl. *three-dimensional*

HTML – Hypertext Markup Language, z angl. *HyperText Markup Language*

CSS – Kaskádové štýly, z angl. *Cascading Style Sheets*

DOM – Dokumentový objektový model, z angl. *Document Object Model*

API – Aplikačné programové rozhranie, z angl. *Application Programming Interface*

PHP – Predspracovateľ hypertextu, z angl. *Hypertext Preprocessor* kedysi aj *Personal Home Page Tools*

REST – Architektonický štýl webových služieb, z angl. *Representational State Transfer*

NoSQL – Nerelačné databázy (nie len SQL), z angl. *Not Only SQL*

XSS – Skriptovanie medzi stránkami, z angl. *Cross-Site Scripting*

CSRF – Falšovanie požiadavky medzi stránkami, z angl. *Cross-Site Request Forgery*

JSON – Notácia objektov JavaScriptu, z angl. *JavaScript Object Notation*

HTTP – Hypertextový prenosový protokol, z angl. *HyperText Transfer Protocol*

JWT – JSON Webový token, z angl. *JSON Web Token*

PDO – PHP dátové objekty, z angl. *PHP Data Objects*

SQL – Štruktúrovaný dotazovací jazyk, z angl. *Structured Query Language*

DSN – Názov dátového zdroja, z angl. *Data Source Name*

napr. – Napríklad

OpenCV – Knižnica počítačového videnia, z angl. *Open Source Computer Vision Library*

Zoznam algoritmov

Zoznam výpisov kódov

Úvod

Športová streľba na terč je disciplína, ktorá vyžaduje vysokú presnosť, sústredenie a stabilnú techniku. S rozvojom moderných technológií sa nie len vzyšujú potreby zefektívniť proces vyhodnocovania samotných striel na terč, ale aj spätnej väzby pre strelcov aby mohli na základe štatistiky posunúť svoje výkony ešte ďalej.

Táto bakalarska práca sa zaoberá vývojom programu na vyhodnocovanie zásahov v športovej streľbe, ktorý následne analyzuje vyznačené zásahy. Cieľom je vytvoriť systém, ktorý by umožnil používateľom pomocou jednoduchého a prehľadného používateľského rozhrania vyznačovať jednotlivé zásahy a následne efektívne poskytovať dáta pre zlepšovanie výkonu športovcov. Program by mal umožniť trénerom a športovcom lepšie porozumieť ich výkonu a identifikovať slabé miesta.

Na základe týchto cieľov bola v práci použitá matóda spracovania obrazu. ...

1 Športová streľba na terč

Streľba na terč má bohatú históriu, ktorá sa vyvinula z životne dôležitej schopnosti prežitia až po moderný súťažný šport. Tento vývoj bol ovplyvnený technologickým pokrokom, zmenami v spoločenských postojoch k strelným zbraniam a lukostreľbe a vývojom rôznych typov terčov, ako sú oceľové, papierové, skákacie a siluetové terče [1].

Streľba na terč bola zaradená do programu olympijských hier už v Paríži v roku 1896, kde sa prvýkrát objavila ako súťažný šport. V tomto období sa objavili rôzne formy streleckých súťaží, vrátane streľby na živé holuby. Po tomto ročníku bolo zavedené strieľanie na terče z umelej hmoty [2].

1.1 Moderná streľba na terč

Existujúce systémy na vyhodnocovanie zásahov v športovej streľbe kombinujú rôzne technológie na meranie a analýzu zásahov. Tieto riešenia zvyčajne zahŕňajú použitie digitálnych terčov, kamier, senzorov alebo iných optických zariadení, ktoré umožňujú automatické zistenie polohy zásahov na terči. Mnohé komerčné systémy používajú kamery na analýzu pohybu a zásahov v reálnom čase. Tieto systémy sú schopné poskytnúť okamžité výsledky, no sú často veľmi nákladné a vyžadujú zložité zariadenie, čo môže byť obmedzujúce pre amatérske alebo menšie strelecké kluby.

Niektoré z týchto riešení využívajú pokročilé metódy spracovania obrazu, ako je detekcia hrán alebo identifikácia štruktúr na terči, aby presne vyhodnotili zásahy. Napríklad, pri analýze obrazu sa často využíva technika, ktorá sleduje zmeny v pixeloch obrazu medzi jednotlivými snímkami, aby sa určilo miesto zásahu. V niektorých prípadoch sa využívajú algoritmy strojového učenia, ktoré sa učia rozpoznať vzory na terči a určiť bod zásahu s vysokou presnosťou.

Existujú aj systémy, ktoré sú zamerané na analýzu štatistík strelca počas viacerých kôl, čím umožňujú trénerom a športovcom lepšie pochopiť výkon a identifikovať oblasti na zlepšenie.

1.2 Prehľad disciplín a ich skórovacích systémov

V športovej streľbe sa v súčasnosti používajú tri základné typy skórovacích systémov.

1. Elektronické, kde sa využívajú moderné senzory spolu so softvérom pre okamžité a presné vyhodnotenie strielania. Jedná sa o veľmi presnú a spoľahlivú metódu využívanú v dnešnej modernej kompetitívnej streľbe.

2. Manuálne skórovanie, ide o streľbu na papierové terče, kde rozhodcovia zaznamenávajú výsledky, však táto metóda vyžaduje pozornosť a férovosť vyhodnocujúceho.

3. Optické skórovanie. Tu sa kombinujú vysokorýchlostné kamery alebo snímače s pokročilými algoritmami na detekciu zásahov. Po počiatočnom automatickom rozpoznaní zásahu nasleduje manuálna verifikácia pre zaistenie maximálnej presnosti vyhodnotenia.

V dnešnom svete existuje množstvo strelných zbraní, preto bolo v športovej streľbe potrebné zaviesť rozličné kategórie a ich príslušné skórovacie systémy. Spomenieme si iba pár pre načrtnutie problematiky a nepôjdeme do hĺbky vysvetlovania, iba ich charakteristiky aby čitateľ pochopil v čom sú odlišné.

- **Olympijská puška**

  Disciplíny: 10 m vzduchová puška, 50 m trojičné polohy

  Skórovanie: Desatinné bodovanie, každá rana až do hodnoty 10,9 bodu.

- **Pištoľ**

  Disciplíny: 10 m vzduchová pištoľ, 25 m rýchlopalná pištoľ, 50 m pištoľ

  Skórovanie: Kombinácia celých a desatinných bodov podľa charakteru súťaže.

- **Broková streľba**

  Disciplíny: trap, skeet, športové hlinenky, double trap

  Skórovanie: Systém zásah/prechyb, strelec získava bod za každý úspešný zásah.

- **Silueta**

  Disciplíny: pušková silueta, pištoľová silueta

  Skórovanie: Body za zhadzovanie kovových siluetových terčov v tvare zvierat na rôznych vzdialenostiach.

- **Praktická streľba (IPSC/USPSA)**

  Disciplíny: pištoľ, puška, brokovnica

  Skórovanie: Kombinácia rýchlosti a presnosti, terče sú rozdelené na zóny s rôznou bodovou hodnotou a časový faktor ovplyvňuje konečný výsledok.

Práve tieto disciplíny boli zvolené pre poučenie čitateľa na základe toho, že majú odlišné terče a skórovania [3]. V tejto práci sa budeme venovať druhej spomenutej disiplíne - Pištoľ a konkrétne, streľbe zo vzduchovej pištole a následne vyhodnoteniu výsledkov.

1.3 Vyhodnocované metriky

V tejto podkapitole predstavíme základné metriky, ktoré sa bežne používajú na kvantifikáciu kvality streleckého výkonu.

- **Presnosť (Accuracy)** Vyjadruje priemerný skóre na každý výstrel. Čím vyššia presnosť, tým bližšie zásahy k stredu terča. [4]
- **Stredný polomer (Mean Radius)** Priemerná vzdialenosť všetkých zásahov od stredu terča. Čím menší stredný polomer, tým tesnejšie sú skupiny. [5]
- **Variancia rozptylu (Dispersion Variance)** Štatistická variancia vzdialeností od stredu - ukazuje rozptyl zásahov v rámci série. [6]
- **Skupinovanie (Grouping)** Počet zásahov v rámci pevného kružnicového priemeru (napr. 10 cm). Používa sa na porovnanie tesnosti rôznych streleckých sérií. [7]
- **Windage (Horizontálny odklon)** Priemerné horizontálne posunutie skupiny vzhľadom na stred - pomáha identifikovať systémový odklon doľava alebo doprava. [7]
- **Elevation (Vertikálny odklon)** Priemerné vertikálne posunutie skupiny vzhľadom na stred - ukazuje, či strelec systematicky strieľa vyššie alebo nižšie. [7]
- **Štandardná odchýlka polomeru (Radius Standard Deviation)** Meria jednotnosť rozptylu zásahov okolo stredného bodu, doplňuje metriky variancie a mean radius. [4]
- **Index konzistencie (Consistency Index)** Pomerný ukazovateľ, definovaný ako podiel medzi priemerným skóre a štandardnou odchýlkou skóre, hodnotí stabilitu výkonu v čase. [4]
- **Čas medzi výstrelmi (Shot Interval)** Priemerné časové rozostupy medzi jednotlivými výstrelmi - dôležité pre analýzu uvoľnenia spúšte a rytmu streľby. [4]

2 Programovací jazyk JavaScript

JavaScript je objektovo orientovaný, vysokoúrovňový programovací jazyk. Narozdiel od kompilovaných jazykov kde sa najprv kód preloží do strojového kódu a následne vytvorí spustiteľný súbor ako napríklad jazyk C, C++ alebo Rust, JavaScript je jazyk interpretovaný - teda kód sa vykonáva riadok po riadku pomocou interpreteru. O vznik tohoto programovacieho jazyka sa zaslúžil Brendan Eich. JavaScript bol vydaný v roku 1995 v máji a to iba za 10 dní. Jeho prvým názvom bolo Mocha [8]. Tento programovací jazyk umožňuje implementovať komplexné funkcie na webových stránkach. Zakaždým, keď webová stránka robí viac, než len sedí a zobrazuje statické dáta - zobrazuje včasné aktualizácie obsahu, interaktívne mapy, animovanú 2D/3D grafiku, rolovacie video jukeboxy alebo iné interaktívne prvky, ide pravdepodobne o zakomponovanie JavaScriptu [9].

JavaScript patrí medzi najpoužívanejšie programovacie jazyky najmä vďaka svojej jednoduchej a flexibilnej syntaxe, ktorá umožňuje rýchly štart a intuitívnu prácu s webovými technológiami.

Okrem svojej úlohy pri základnej interaktivite sa JavaScript v modernej webovej architektúre presunul z doplnkového jazyka fungujúceho popri html a css na samotne plnohodnotný základ pre vývoj komplexných aplikácií. Vďaka vývoju frameworkov ako React, Vue či Angular sa JavaScript stal nástrojom nielen pre manipuláciu s DOM - Document Object Model - štruktúrovaná reprezentácia HTML, ale aj pre budovanie komponentovo orientovaných systémov, spracovanie stavov aplikácií, routing či prácu s API na strane klienta. Tento posun umožnil načítavanie nových dát a informácií bez opätovného načítania celej stránky. Výsledkom je rýchlejšia, plynulejšia a používateľsky prívetivejšia skúsenosť, porovnateľná s natívnymi aplikáciami.

2.1 Výhody jazyka JavaScript

Medzi veľkú výhodu jazyka patrí "Client-Side Scripting", Teda JavaScript beží v používateľovom prehliadači, to znamená, že má kratší čas odpovedí, pretože nie je nutné komunikovať so serverom. Veľmi kladnou záležitosťou tohoto jazyka je určite jeho všestrannosť, JavaScript je možné použiť pre rôznu škálu úloh, od jednoduchých výpočtov až po zložité aplikácie na strane servera. Dôvodom, prečo je jazyk tak populárny pri vývoji webových aplikácií je, že umožňuje reagovať na klikanie, stlačenie kláves, zmeny okna a rozhrania, atď. v reálnom čase. Javascript funguje asynchrónne - teda zvláda úlohy ako načítanie dát zo servera bez toho aby zamrzlo používateľské rozhranie. Vďaka obľúbenosti tohoto jazyka medzi programátormi po celom svete, tento jazyk disponuje množstvom dokumentácie a tutoriálov [10].

2.2 Nevýhody jazyka JavaScript

JavaScript je interpretovaný jazyk, to znamená, že je pomalší ako kompilované jazyky ako sú C, C++ alebo Pascal. Keďže JavaScript beží na strane klienta, je náchylný na útoky ako Cross-Site Scripting (XSS) a Cross-Site Request Forgery (CSRF). JavaScript je dynamicky typovaný jazyk, čo znamená, že typy premenných nie sú striktne definované. To môže viesť k chybám, ktoré sa objavia až pri spustení programu, a sťažuje to údržbu väčších kódových báz. JavaScript sa vo veľkej miere spolieha na asynchrónne programovanie, čo môže byť pre vývojárov náročné na pochopenie a správne implementovanie. Aj keď moderné konštrukcie ako Promises a async/await pomáhajú, stále existuje riziko vzniku tzv. "callback hell-[11].

2.3 JavaScript na serveri

Hoci bol JavaScript pôvodne vytvorený ako skriptovací jazyk pre webové prehliadače, vďaka Node.js sa rozšíril tento jazyk aj na stranu serveru - teda back-end. Node.js umožňuje vývojárom písať serverový kód v rovnakom jazyku, aký používajú aj pre klientskú časť, čo zjednodušuje vývoj tzv. fullstack aplikácií.

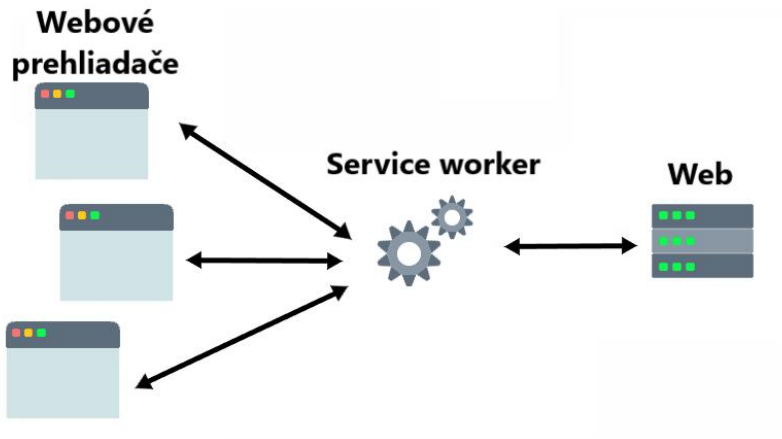
Medzi najbežnejšie využitia JavaScriptu na serveri patrí obsluha API požiadaviek, spracovanie dát, práca s databázami a správa autentifikácie. Jednoduchý a výkonný framework ako Express.js výrazne urýchľuje tvorbu webových serverov. Týmto spôsobom sa z JavaScriptu stal univerzálny jazyk schopný pokrývať celý vývojový cyklus webovej aplikácie [12].

3 PWA

PWA je skratkou pre progressive web app - teda progresívna webová aplikácia vytvorená pomocou štandardných webových technológií ako HTML, CSS a JavaScript, ktorá využíva moderné webové schopnosti na poskytnutie používateľského zážitku podobného natívnym aplikáciám. Tieto aplikácie môžu byť inštalované na zariadenia a fungovať offline, čo zvyšuje ich dostupnosť a použiteľnosť.

3.1 Silné stránky PWA

Dostupnosť bez ohľadu na internetové pripojenie - Progresívne webové aplikácie nie sú závislími od pripojenia používateľa k internetu tak ako to býva u zvyčajných webstránok. Pri používateľovej návšteve PWA bude zaregistrovaný service worker, ktorý deteguje a reaguje na zmeny používateľovho pripojenia. Práve pre túto prácu teda bude kľúčovou funkcionalitou to, že dokáže poskytnúť plnohodnotnú funkcionalitu aj pre takých používateľov ktorý nie sú v daný moment propojený k internetu. Podrobnosti budú v podkapitole 3.2.



Obr. 1: Názov obrázka [13]

Krátke doby načítavania - Pri používaní service workerov je možné vytvárať stránky, ktoré budú zobrazené hneď po otvorení bez ohľadu na to ako silné a dobré pripojenie používateľ má. Stŕanky sa dokážu načítať v priebehu milisekúnd za použitia metódy “offline first” [13].

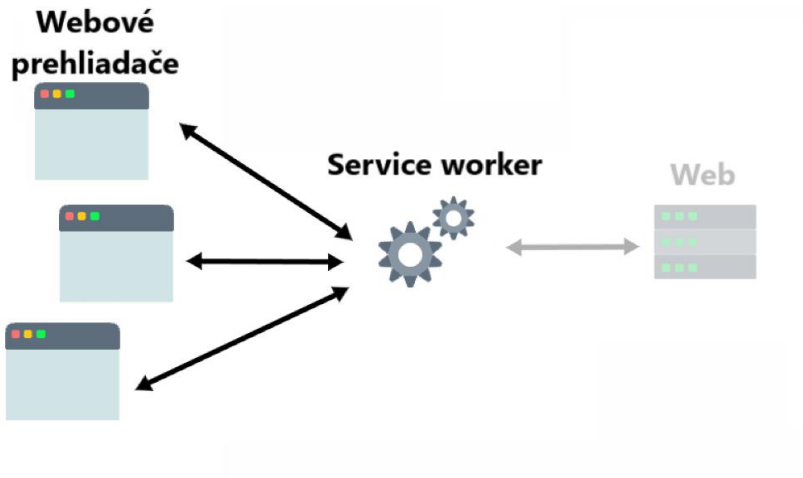
Notifikácie - Progresívne webové aplikácie možu posielať používateľom takzvané "Push notificationsäj niekoľko dní po návšteve stránky. Tieto notifikácie sú skvelým nástrojom ako si môže stránka udržať viditeľnosť a svojich navštevovateľov. [13].

Natívny výzor - PWA môžu mať kompletne nerozoznatelný vzhľad od klasických desktopových aplikácií. Je možné aby sa zapínali bez prehliadačového user interface-u, mali animáciu alebo nejaké intro pri spúštaní, dokonca sa môžu aj uzamknúť na predvolenú orientáciu, na šírku alebo výšku.

3.2 Neviditeľná vrstva

Tak ako sme už načrtli, základom každej progresívnej webovej aplikácie je service worker. Pred tým ako sa vôbec service workeri objavili, kód bežal buď na strane serveru alebo na strane klienta, spolu so service workermi bola zavedená nová vrstva. Service worker je umiestnený za webovými prehlidačmi, tak ako je zobrazené na obrázku 1. Service worker dokáže takto obsluhovať rôzne podnety zo všetkých stránok pod jeho kontrolou.

Práve vďaka tejto vrstve dokážeme odpovedať na podnety bez ohľadu na pripojenie používateľa - teda aj keď je offline, práve to čo je vhodné pre našu aplikáciu. Táto vrstva taktiež dokáže detegovať pomalé odozvy od servera a posielať obsah v cache.



Obr. 2: Názov obrázka [13]

Analogicky touto logikou je možná aj funckia spomínaných push notifikácií po tom ako používateľ opustil webovú stránku. Teda service worker ostáva stále pracovať a môže s webom naďalej komunikovať.

4 Programovací jazyk PHP

PHP je dlho a široko používaný skriptovací jazyk, ktorý je zvlášť vhodný na vývoj webových aplikácií, teda priam vytvorený na tento účel. PHP môže byť vložený priamo do HTML za pomoci špeciálnych php tagov. PHP, ktoré pôvodne vytvoril Rasmus Lerdorf v roku 1995, sa vyvinulo prostredníctvom viacerých verzií do vyspelej a flexibilnej platformy na vytváranie dynamických, interaktívnych webových stránok [14]. Napriek tomu, že jazyk PHP pôsobí na prvý pohľad veľmi jednoducho, vďaka svojej flexibilite a bohatému ekosystému knižníc je schopný uspokojiť aj náročné požiadavky moderných webových aplikácií. PHP umožňuje rýchly vývoj vďaka dynamickému typovaniu, jednoduchej a prehľadnej syntaxi i rozsiahlej štandardnej knižnici, pričom stále podporuje pokročilé paradigmy, ako je objektovo-orientované programovanie či funkcionálna štylizácia kódu. Okrem priamočiareho generovania HTML dokáže PHP spracovávať REST API, vykonávať dávkové úlohy cez CLI, pracovať so súborovým systémom či komunikovať so širokým spektrom databázových aj no-SQL úložísk. Vďaka tomu sa PHP stalo jedným z pilierov back-endového vývoja - od jednoduchých skriptov až po komplexné, enterprise-úrovňové riešenia s dôrazom na bezpečnosť, škálovateľnosť a udržiavateľnosť.

4.1 Výhody jazyka PHP

- Jednoduchý štart a nízka krivka učenia - PHP má prehľadnú a ľahko pochopiteľnú syntax, taktiež vďaka svojej popularite a viac než dvadsať ročnej histórii je dostupných nespočetne veľa video návodov, článkov a diskusií [15].
- Široká dostupnosť na hostingových platformách - Vďaka popularite PHP je interpreter predinštalovaný prakticky na všetkých formách webhostingu vrátane lacných “shared” hostingov, čím odpadá potreba špeciálnych nastavení [15].
- Výkonnosť a škálovateľnosť - Moderné verzie PHP 7.x a 8.x priniesli výrazné zrýchlenie vykonávania skriptov a nižšiu spotrebu pamäte. Vďaka OPcache sa PHP aplikácie dajú škálovať aj pri vysokých nárokoch na výkon [16].
- Jednoduchá implementácia PHP do HTML a integrácia webových štandardov - PHP kód možno jednoducho vkladať priamo do HTML dokumentov a prirodzene pracuje so štandardmi HTTP, cookies alebo session, čo značne zjednodušuje tvorbu dynamického webu [15].

4.2 Nevýhody jazyka PHP

Ako open-source jazyk je PHP k dispozícii v podobe čitateľného zdrojového kódu, to znamená, že každý môže študovať, upravovať a vylepšovať jeho vnútornú logiku, no zároveň to útočníkom uľahčuje objaviť a zneužiť prípadné bezpečnostné chyby. Ak sa nedodržujú osvedčené bezpečnostné postupy môže vzniknúť riziko ako (napr.: SQL injection, XSS a CSRF útoky) [17]. Zároveň v porovnaní s inými jazykmi ponúka PHP menej pokročilé mechanizmy pre vykresľovanie a spracovanie výnimiek, toto všetko komplikuje odhalovanie a ladenie chýb [18]. PHP kód vyžaduje prítomnosť webového servera ako napríklad Apache alebo Nginx s nainštalovaným interpretrerom, čo ho robí menej flexibilným pre skriptovanie na príkazovom riadku [17].

4.3 REST API v PHP

REST (Representational State Transfer) je architektonický štýl pre navrhovanie webových služieb, ktorý kladie dôraz na jednoduchú komunikáciu cez štandardné HTTP metódy bez ukladania stavu na strane servera. Základné princípy ako bezstavovosť požiadaviek, možnosť cacheovania serverových odpovedí a jednotné rozhranie zdrojov umožňujú vytvárať vysoko dostupné a škálovateľné služby [19].

V prostredí PHP možno REST API implementovať už bez akýchkoľvek externých knižníc, čím zostáva kód ľahký a prehľadný. Typický tok spracovania požiadavku zahŕňa načítanie HTTP metódy a URI zo superglobálnych premenných, čítanie JSON tela požiadavky a odoslanie odpovede formou JSON cez funkciu `json_encode()` s hlavičkou `Content-Type: application/json`. Praktická implementácia REST API v čistom PHP zahŕňa:

- identifikáciu metódy (`GET`, `POST`, `PUT`, `DELETE`) a cesty (`REQUEST_URI`) pre smerovanie požiadaviek,
- spracovanie vstupných dát z JSON a ich validáciu pred ďalším spracovaním,
- tvorbu odpovede s vhodnými HTTP stavovými kódmi (200, 201, 400, 404, 500) a kontrolu cacheovacích hlavičiek (`Cache-Control`),
- ochranu proti bežným útokom (SQL injection, XSS, CSRF) pomocou prepared statements a sanitizácie vstupov.

Frameworky ako Slim, Lumen, Laravel či Symfony prinášajú do vývoja REST API v PHP predpripravené nástroje pre spravovanie routovania, spracovanie požiadaviek či odpovedí a validáciu vstupov. Vďaka nim vývojár už nemusí písať opakujúcu sa boilerplate logiku, ale môže sa sústrediť na implementáciu dôležitej logiky. Tieto riešenia okrem

využitia jednotných konvencií podporujú aj jednoduchú integráciu bezpečnostných mechanizmov (napr. overovanie prístupu cez JWT), ponúkajú vlastné knižnice pre logovanie a testovanie API a poskytujú detailnú dokumentáciu a komunitnú podporu. Výsledkom je rýchlejší postup vývoja, vyššia spoľahlivosť kódu a ľahšia údržba celej aplikácie [20].

4.4 Práca s databázami v PHP

Práca s relačnými databázami je kľúčovou súčasťou mnohých PHP aplikácií. PHP poskytuje dve hlavné rozhrania pre komunikáciu s databázou MySQL a MariaDB, v práci budeme používať rozhranie MySQLi s univerzálnou knižnicou PDO (PHP Data Objects).

Používanie PDO sa odporúča predovšetkým kvôli jeho flexibilite - PDO podporuje viacero databázových systémov (napr. MySQL, PostgreSQL, SQLite) a umožňuje bezpečné vykonávanie SQL dotazov pomocou prepared statements, čím významne znižuje riziko SQL injection [21].

Pri vytváraní spojenia s databázou cez PDO sa najprv inicializuje nový objekt triedy PDO s odkazom na DSN (Data Source Name) kde je následne definované používateľské meno s heslom tak ako to je možné vidieť v úkážke kódu 1.

```
$dsn = 'mysql:host=localhost;dbname=strelnica;charset=utf8';
$username = 'uzivatel';
$password = 'heslo';
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
];
$pdo = new PDO($dsn, $username, $password, $options);
```

Výpis kódu 1: Vytvorenie spojenia s databázou pomocou PDO

V parametri PDO::ATTR_ERRMODE sa nastavuje režim hlásenia chýb, čo uľahčuje hľadanie a následné ladenie týchto chýb pri komunikácii s databázou.

Prepared statements predstavujú základný mechanizmus na prevenciu SQL injection útokov. Dotaz sa v nich najprv pripraví s názvami parametrov tak ako môžeme vidieť v ukážke kódu 2.

```
$stmt = $pdo->prepare('SELECT * FROM users WHERE email = :email');
$stmt->execute(['email' => $emailAdresa]);
$user = $stmt->fetch();
```

Výpis kódu 2: Prepared statements

Namiesto priameho vkladania používateľského vstupu do reťazca SQL sa hodnota parametra :email viaže až pri volaní metódy execute(). Všetky špeciálne znaky sa pritom správne escapujú a to vedie k zamedzeniu vloženia škodlivého kódu do dotazu [22]. Podobným spôsobom sa pripravujú aj príkazy INSERT, UPDATE alebo DELETE.

Transakcie umožňujú vykonať viacero súvisiacich operácií ako jeden logický celok. V PHP sa transakcia otvorí volaním $pdo->beginTransaction(), následne sa vykonajú jednotlivé príkazy, a až po úspešnom dokončení všetkých príkazov sa transakcia potvrdí pomocou $pdo->commit(). Táto praktika bola zavedená, aby v prípade chyby bolo možné transakciu zrušiť volaním $pdo->rollBack(). Tento prístup sa využíva napríklad pri registrácii nového používateľa, ktorá vyžaduje vloženie údajov do viacerých tabuliek súčasne - teda ak by niektorý z príkazov zlyhal, transakcia sa zruší a databáza zostane v nezmenenom stave - nebude zapísaný žiaden údaj.

Bežné operácie s databázou zahŕňajú získavanie údajov a ich následné spracovanie v PHP. Na výber dát z tabuľky je možné použiť metódu fetchAll() na získanie všetkých riadkov alebo fetch() na postupné načítanie z databázy riadok po riadku ako môžeme pozorovať v ukážke 3.

```
$stmt = $pdo->query('SELECT id, meno, skore FROM vysledky ORDER BY skore DESC');
$vysledky = $stmt->fetchAll();
foreach ($vysledky as $riadok) {
    // ďalšie príkazy...
}
```

Výpis kódu 3: Výber pomocou fetchAll()

Z organizačného hľadiska je vhodné databázové prístupy oddeliť do vlastnej triedy alebo samostatného súboru (napr. config.php a Object.class.php). V týchto súboroch sa často definuje metóda na vytvorenie PDO spojenia a potom následne pre daný "objekt" metódy pre vykonanie dotazov ako môžeme pozorovať na následovných ukážkach 9 a 4.

```
class Object {
  // deklarácia db a konštruktor

  public function show($id) { // získanie jedného záznamu
      $stmt = $this->db->prepare("SELECT * FROM objects WHERE id = :id");
      $stmt->bindParam(':id', $id, PDO::PARAM_INT);
      $stmt->execute();
      return $stmt->fetch(PDO::FETCH_ASSOC);
  }
  // ďalšie funkcie
}
```

Výpis kódu 4: Ukážka súboru Object.class.php

Týmto spôsobom sa minimalizuje opakovanie kódu - teda pri každom dotaze sa iba zavolá Database::query() s príslušným SQL a parametrami, zatiaľ čo samotné pripojenie sa spravuje len raz.

Optimalizácia výkonu databázových dotazov sa zvyčajne rieši správnym indexovaním tabuliek. Primárne kľúče, jedinečné indexy a indexy na často filtrovaných stĺpcoch (napr. email, timestamp) zaručujú rýchle vyhľadávanie.

Použitie PDO namiesto MySQLi prináša aj ďalšiu výhodu, ak by sa v budúcnosti rozhodlo migrovať na iný databázový systém - väčšina kódu zostane bežať bez zmien, stačí aktualizovať DSN a prípadne niektoré špecifické nastavenia. Celkovo spracovanie prístupu k databáze v PHP vyžaduje kombináciu bezpečnostných postupov (prepared statements, validácia vstupov), správu chýb cez výnimky a optimalizáciu dotazov, čím sa zabezpečí spoľahlivosť a škálovateľnosť celého riešenia [21].

5 OpenCV

OpenCV je skratka pre Open Source Computer Vision a je to knižnica funkcií, ktoré sú užitočné pri programovaní aplikácií počítačového videnia v reálnom čase. Termín počítačové videnie sa používa pre subjekt, ktorý vykonáva analýzu digitálnych obrázkov a videí pomocou počítačového programu. Počítačové videnie je dôležitou súčasťou moderných disciplín, akými sú umelá inteligencia a strojové učenie [23]. V práci spomínam funkcie mnou považované za kľúčové pre túto prácu, samotná knižnica disponuje nikoľkonásobne viac funkciami, pre viac informácií je dostupná dokumentácia knižnice [24].

5.1 Čítanie obrázkov

Medzi jednu z najdôležitejších funckcií knižnice patrí funkcia na čítanie obrázkov. Knižnica OpenCV poskytuje funkciu s názvom imread(), ktorá slúži na načítanie obrázku do pamäte zo zvoleného súboru pomocou prvého argumentu funkcie.

Funkcia prečíta obrázok a uloží ho vo formáte matice, kde tvar matice zodpovedá rozmerom obrázka. Druhým argumentom funkcie je parameter menom flags. Tento parameter rozhoduje o tom ako je obrázok načítaný, teda môže byť načítaný ako farebný (predvolené), šedý (jeden kanál) alebo nezmenený, teda čítaný taký aký je, ale zachovávajúci všetky kanály transparentnosti.

5.2 Vlastnosti obrázka

Medzi funkcie pracujúce s vlastnosťami obrázkov radíme funkcie shape() a resize()

Funkcia shape() sa používa na získanie rozmerov obrázka alebo matrice. Táto funkcia vráti veľkosť obrázka alebo matice ako n-ticu, čo môže byť užitočné na pochopenie štruktúry údajov. Často sa používa na kontrolu vlastností obrazu, ako sú šírka, výška a farebné kanály.

Funkcia resize() sa používa na zmenu veľkosti obrázka na zadanú veľkosť alebo podľa danej mierky. Táto funkcia je kľúčová pri mnohých úlohách spracovania obrazu, kde je potrebné upraviť veľkosť obrázkov pred vykonaním operácií, ako je detekcia alebo klasifikácia objektov.

5.3 Vyobrazenie geometrických útvarov

Kreslenie tvarov a textu na obrázky je užitočné na vizualizáciu údajov alebo anotácií v aplikáciách počítačového videnia. OpenCV poskytuje rôzne funkcie na kreslenie

základných tvarov ako sú čiary(cv2.line), obdĺžniky(cv2.rectangle) a kruhy(cv2.circle), taktiež umožňuje pridávanie textu(cv2.puText) do obrázkov.

6 Analýza existujúcich riešení

V práci sa zameriame na podrobnú analýzu existujúcich riešení na vyhodnocovanie zásahov pri športovej streľbe. Najprv objasníme, prečo je dôležité ponúknuť cenovo dostupné a efektívne nástroje pre kluby či jednotlivcov, ktorí nemajú možnosť používať drahé elektronické terče či iné pokročilé zariadenia. Potom predstavíme konkrétne príklady aplikácií a softvérov, ktoré dnes umožňujú digitalizovať a vyhodnocovať papierové terče pomocou bežného smartfónu, tabletu či skenera.

6.1 Motivácia

Nie každý strelecký klub alebo individuálny strelec má finančnú možnosť investovať do elektronických terčov či pokročilých video-optických systémov. Pokiaľ sa zaznamenávanie zásahov vedie manuálne (papierové prehľadové terče), existujú softvérové nástroje, ktoré pomocou telefónu alebo skenera dokážu tieto stredné body zásahov digitalizovať.

6.2 Príklady existujúcich aplikácií

Cieľom tejto analýzy je nie len zhrnúť hlavné funkcie a vlastnosti existujúcich riešení, ale aj identifikovať ich silné a slabé stránky pre následné zlepšenie. Zároveň bude čitateľ oboznámený s tým, aké riešenia sú dnes na trhu dostupné a kde sa nachádzajú potenciálne medzery.

6.2.1 TargetScan - Pistol & Rifle

Aplikácia je určená výhradne pre mobilné zariadenia (iOS aj Android) a nie je dostupná na počítači ani pre spracovanie skenovaných fotografií terča z galérie. Aplikácia vyhodnocuje pomocou algoritmu automaticky vrátanie INNER hodnôt a desatinného skórovania. Aplikácia disponuje automatickou detekciou dierok v terči s možnosťou posúvania jednotlivých zásahov v prípade chybnej detekcie [25].

Aplikácia taktiež disponuje výpočtom stredného bodu nárazu, vertikalnou a horizontálnou elevációu, priemerný polomer skupiny a maximálna vzdialenosť medzi dvoma najvzdialenejšími bodmi [25].

- Výhody aplikácie:

  Zobrazenie “Potential Score” - ukazovateľ, o koľko bodov by sa výsledok zlepšil, ak by strelecká skupina bola presunutá presne do stredu terča. “Heat map” zobrazujúca vzory odchýlok v dlhodobom horizonte. Graf vývoja skóre a rozptylu naprieč časom. Porovnanie viacerých sedení s cieľom sledovať progres - napríklad skóre za posledné mesiace [25].

- Nevýhody aplikácie:

  Iba mobilná verzia - nie je možné vyhodnocovať skeny terčov v počítači. Cena aplikácie je 20 €, čo môže byť pre niektorých používateľov vyššia investícia. Zastaralý dizajn, ktorý neodpovedá moderným štandardom používateľského rozhrania. Obmedzovanie na základné disciplýni - v aplikácii sú možnosti ďalších poplatkov, ktoré odomknú pokročilé disciplíny. Aj keď funkcia "Eagle Eye" umožňuje posun zásahu o drobné hodnoty, pri veľmi sústredenej skupine môže byť ťažké rozhodnúť, ktorú dierku presne posunúť, ak sú na fotografii takmer zlepené [25].

6.2.2 Blackhole-App ("Scan Targets")

Aplikácia je určená výhradne pre mobilné zariadenia (iOS aj Android) a nie je dostupná na počítači ani pre spracovanie skenovaných fotografií terča z galérie. Pomocou zabudovaného optického vodováhy pomáha nasnímať terč kolmo a stabilne. Po nasnímaní aplikácia automaticky vyhodnotí zásahy a zobrazí výsledné hodnoty spolu s grafickou vizualizáciou zásahov.

Používateľ môže po vyhodnotení manuálne pridávať, upravovať alebo mazať jednotlivé zásahy. Výsledky je možné prezerať a porovnávať s predchádzajúcimi sedeniami, taktiež je možné exportovať výsledky do PDF [26].

- Výhody aplikácie:

  Automatické vyhodnotenie zásahov na desatinu prstenca** v priebehu niekoľkých sekúnd po nasnímaní terča. Možnosť manuálnej korekcie, po automatickom spracovaní je možné pridať, upraviť alebo odstrániť zásah v prípade nepresnej detekcie. Záznamník s podporou grafickej analýzy (rozptyl, stredný bod impaktu) a porovnávaním viacerých streleckých sedení. Export do PDF [26].

- Nevýhody aplikácie:

  Iba mobilná verzia - nie je možné vyhodnocovať skeny terčov v počítači. V aplikácii sú možnosti ďalších poplatkov, ktoré odomykajú mnohé zo spomenutých funkcionalít. Občasné pády či zdržanie aplikácie pri spracovaní veľkého počtu zásahov, čo potvrdzujú aj recenzie používateľov. Neintuitívne rozhranie [26].

6.2.3 AccuShoot

Aplikácia je dostupná pre mobilné zariadenia (iOS aj Android) a poskytuje skórovanie terčov tak v reálnom čase, ako aj z fotografií terčov. Vďaka funkcii *Real-time Target Scoring* môže strelec sledovať výsledky každého zásahu okamžite bez potreby vracať sa k terču, čo výrazne zvyšuje komfort a dynamiku tréningu. Pre pokročilú analýzu sú k

dispozícii detailné metriky rozptylu, stredného bodu impaktu, priemerného polomeru skupiny a maximalnej vzdialenosti medzi najvzdialenejšími zásahmi. Aplikácia ponúka aj prehľadnú históriu všetkých streleckých sedení vrátane interaktívnych grafov vývoja skóre a skupinových analýz, ktoré umožňujú jednoduché sledovanie osobného progresu. Pomocou desktopovej aplikácie *AccuShoot Desktop App* je navyše možné vykonať predikciu budúceho výkonu na základe historických dát a exportovať podrobné reporty vo formáte PDF alebo CSV. Sociálne rebríčky a funkcia *community challenges* umožňujú porovnávať sa s ostatnými používateľmi, zdieľať výsledky a motivovať sa prostredníctvom súťaživého prostredia. Funkcia *picture-based sessions* navyše podporuje digitalizáciu starších terčov – stačí nahrať fotografiu, a aplikácia automaticky deteguje a vyhodnotí zásahy vrátane možnosti manuálnej korekcie [27].

- Výhody aplikácie:

  Real-time Target Scoring umožňuje okamžité vyhodnotenie zásahov bez nutnosti chodiť k terču. Pokročilá analýza skupín - automatické výpočty pre hodnotenie rozptylu, stredného bodu impaktu a ďalších metrík. Prostredníctvom bezplatnej desktopovej aplikácie AccuShoot Desktop App je možné predikovať výkon na základe dát z minulosti. Moderný dizajn a intuitívne rozhranie. Sociálne rebríčky pre porovnanie výkonu s inými strelcami. Analýza starších streleckých terčov cez nahratie fotografie do desktopovej aplikácie [27].

- Nevýhody aplikácie:

  V aplikácii sú možnosti ďalších poplatkov, ktoré odomykajú mnohé zo spomenutých funkcionalít. Nutnosť internetového pripojenia, synchronizácia dát a prístup k sociálnym rebríčkom vyžaduje stabilné pripojenie, čo môže obmedziť použitie v odľahlých lokalitách. Obmedzená presnosť pri špeciálnych terčoch: niektoré neštandardné terče alebo podmienky nemusí aplikácia optimálne podporovať, výsledky môžu byť menej presné v porovnaní s desktopovou analýzou [27].

6.2.4 Zhrnutie funkcionality dostupných rieŠení

V nasledujúcej podkapitole predstavíme prehľad základných funkcií jednotlivých dostupných riešení formou tabuľky. Táto tabuľka umožní rýchlu orientáciu a porovnanie kľúčových vlastností bez potreby prechádzať rozsiahlymi opisnými textami. V ľavom stĺpci bude uvedený názov riešenia, v ďalších stĺpcoch sa čitateľ prehľadne dozvie o podpore danej funckie.

Funkcia / Vlastnosť	**TargetScan**	**Blackhole-App**	**AccuShoot**
Platforma	Áno	Áno	Áno
Živé snímanie terča	Áno	Áno	Áno
Fotky z galérie	Nie	Nie	Áno
Automatické skórovanie	Áno	Áno	Áno
Manuálna korekcia zásahov	Áno	Áno	Nie
Pokročilé analytické metriky	Áno	Áno	Áno
História výkonou sedení	Áno	Áno	Áno
Export / synchronizácia	Nie	Áno	Áno
Cena / model platby	jednorazovo	in-app poplatky	in-app poplatky
UI - moderné / intuitívne	Nie	Nie	Áno
Obmedzenia	len mobil	len mobil	potreba internetu

Tabuľka 1: Porovnanie hlavných funkcií aplikácií

7 Návrh riešenia

Témou práce je vytvoriť aplikáciu, ktorá z fotografie terča transformuje perspektívu a pripravý plátno pre používateľa, tak aby mohol potvrdiť zásahy v terči a následne mu boli vygenerované výsledky. V tejto kapitole sa čitateľ dozvie aké technológie budeme používať a ako bude navrhnutá celá aplikácia za pomoci týchto technológií.

Na základe prieskumu sme sa rozhodli použiť JavaScript, ako bolo ukázané v teoretickej časti, pretože podporuje knižnicu OpenCV (viď podkapitolu 5) a beží vo všetkých moderných prehliadačoch bez potreby inštalácie ďalšieho softvéru. Okrem toho má obrovskú komunitu, množstvo návodov, tutoriálov a praktických príkladov, vrátane ukážok použitia OpenCV.js. Na prístup k databáze a jej správu využijeme PHP a MySQL.

7.1 Voľba hodnotiacich parametrov pre aplikáciu

Na základe analýzy potrieb používateľov a dostupných metrik sme pre vyhodnotenie streleckého výkonu v aplikácii vybrali nasledujúce parametre.

Najlepší výstrel označuje zásah s najvyšším bodovým hodnotením v rámci série, čím sa identifikuje moment maximálnej presnosti

najhorší výstrel je zásah s najnižším skóre, ktorý poukazuje na najväčšiu odchýlku od ideálneho výstrelu

priemerné skóre sumarizuje celkovú výkonnosť série a slúži ako základná metrika konzistentnosti

mean radius (stredný polomer) meria priemernú vzdialenosť všetkých zásahov od stredu terča a charakterizuje tesnosť skupiny

radius variance (variancia polomeru) kvantifikuje rozptyl týchto vzdialeností a odhaľuje nerovnomernosť rozloženia zásahov

consistency (MAD) využíva medián absolútnych odchýlok pre odhad odolnosti voči extrémom a poskytuje robustný ukazovateľ stability výkonu

elevation a *windage* sledujú systémové vertikálne a horizontálne posunutie skupiny voči stredu, čím odhaľujú opakované odklony dohora, nadol, doľava či doprava

max spread (maximálny rozptyl medzi dvoma výstrelmi) identifikuje najväčší možný rozmer skupiny v danej sérii, čo je kľúčové pre hodnotenie extrémnych odchýlok. Tieto parametre spoločne pokrývajú aspekty presnosti, konzistencie i systematických chýb a umožňujú komplexné vyhodnotenie streleckého výkonu.

7.2 Podmienky pre fotografie terčov

V práci používame na detekciu a získavanie dát z fotografie knižnicu OpenCV (viď podkapitolu 5), preto je nevyhnutné zaviesť prísne pravidlá pre snímanie terča, aby algoritmy mohli spoľahlivo a presne identifikovať zásahy.

Pri fotení terčov je potrebné dodržiavať následné pravidlá. Terč musí byť na obrázku úplne viditeľný - nesmú chýbať žiadne rohy ani okraje. Papier, na ktorom je vytlačený terč, nesmie byť pokrčený, prehnutý ani inak zdeformovaný. Povrch musí byť rovný. Osvetlenie terča musí byť rovnomerné a dostatočné, pričom treba dbať na elimináciu tieňov. Užívateľ by mal fotografovať terč čo najpriamejšie, ideálne s telefónom drženým vodorovne aj zvisle, čím sa minimalizuje perspektívne skreslenie. Odporúča sa približný záber z primeranej vzdialenosti - ani príliš z diaľky (kde klesá ostrosť a rozlíšenie), ani príliš zblízka (kedy môže dochádzať k geometrickému skresleniu okrajov). Cieľom je mať terč vycentrovaný v strede snímky, aby OpenCV dokázalo spoľahlivo vykonať kalibráciu, detekciu stredu a rohov terča.

7.3 Špecifikácia požiadaviek

V tejto časti zhrnieme základné požiadavky na navrhovanú aplikáciu, ktoré zabezpečia správnu funkčnosť a použiteľnosť.

Funkčné požiadavky

- **Zachytenie terča** - nahratie obrázkov z galérie.
- **Automatické vyhodnocovanie terča** - detekcia rohov a stredu terča, s možnosťou manuálnej úpravy v prípade chybnej detekcie.

- **Manuálny výber bodov** - Manuálne zakliknutie výstrelov na terči.
- **Výpočet metrík** - najlepší a najhorší výstrel, priemerné skóre, mean radius, radius variance, consistency (MAD), windage, elevation, max spread a historické štatistiky pre sledovanie progresu.
- **Vizualizáciu výsledkov** - grafy vývoja skóre, heat mapy rozptylu, porovnanie s predchádzajúcimi sedeniami.
- **Ukladanie dát** - ukladanie v databáze.

Nefunkčné požiadavky

- **Platformová nezávislosť** - podpora webového rozhrania pre desktop a mobilné telefóny.
- **Výkonnosť** - spracovanie jednej snímky a zobrazenie výsledkov do 10 sekúnd.
- **Bezpečnosť a ochrana dát** - šifrovanie uložených záznamov, zabezpečené prihlasovanie.
- **Používateľské rozhranie (UX)** - intuitívne rozhranie.
- **Škálovateľnosť** - možnosť rozšírenia o nové metriky bez zásadných zmien architektúry.

7.4 Diagramy

V tejto kapitole budeme vytvárať diagramy pre popísanie funcionality našej aplikácie. Pre najzákladnejší opis toho, čo aplikácia dovoluje navrhneme diagram prípadov použitia. Následne budeme vytvárať entitno relačný diagram, ktorý bude opisovať to aké stĺpce a polia bude databáza obsahovať.

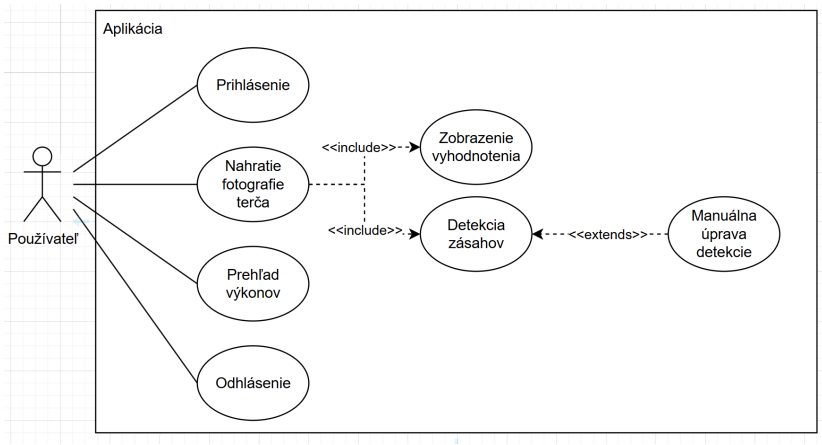
7.4.1 Diagram prípadov použitia

Na obr. 3 je znázornený diagram prípadov použitia, ktorý popisuje hlavné interakcie medzi používateľom (strelec) a systémom aplikácie. Primárnym aktérom je *Používateľ*, ktorý vykonáva nasledovné prípady použitia:

- **Nahratie fotografie terča** - vstupný krok, pri ktorom používateľ nahraje snímku terča z galérie. Tento prípad použitia *zahŕňa* (include) *Zobrazenie vyhodnotenia* a *Detekciu zásahov*.

- **Detekcia zásahov** - automatický výpočet polohy a skóre zásahov. Ak detekcia nevyhovuje, rozširuje sa (extend) o *Manuálnu úpravu detekcie* alebo *Vyklikanie výstrelov*, ktoré umožňujú používateľovi doladiť alebo označiť chýbajúce zásahy.
- **Zobrazenie vyhodnotenia** - okamžitá vizualizácia výsledkov a metrík po spracovaní snímky.
- **Prehľad výkonov** - samostatný prípad použitia umožňujúci pristupovať k histórii všetkých predchádzajúcich sedení a štatistík.

Diagram využíva spojky *include* na označenie povinných krokov, ktoré sa vždy vykonajú (napr. zobrazenie výsledkov po nahratí fotografie) a spojky *extend* pre voliteľné, situáciu-podmienené akcie (napr. manuálna korekcia detekcie zásahov). Takto rozdelené logické bloky uľahčujú pochopenie, ktoré funkcie sú nevyhnutné pre základný tok, a ktoré sa aktivujú iba v prípade potreby doplnenia či opravy automatizovaných procesov.



Obr. 3: Diagram prípadov použitia pre aplikáciu

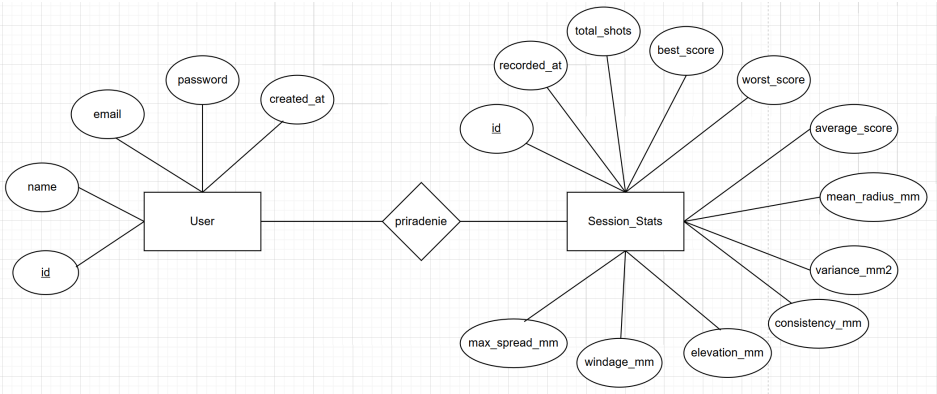
7.4.2 ER diagram

Na obr. 4 je znázornený ER diagram, ktorý obsahuje entity, ich atribúty a vzťahy v navrhovanej aplikácii na hodnotenie strelby na terč. Diagram pozostáva z dvoch hlavných entít *User* a *Session_Stats*, medzi ktorými je definovaný many-to-many vzťah.

- **User** (entita) - predstavuje registrovaného používateľa aplikácie. Atribúty:
  - `id (PK)` - primárny kľúč
  - `name` - meno používateľa
  - `email` - e-mailová adresa (index)
  - `password` - heslo (hash)

  - `created_at` - dátum registrácie

- **Session_Stats** (entita) – predstavuje jednu streleckú sériu so súhrnnými metrikami. Atribúty:
  - `id (PK)` - primárny kľúč
  - `recorded_at` - dátum a čas záznamu
  - `total_shots`, `best_score`, `worst_score`, `average_score`
  - `mean_radius_mm`, `variance_mm2`, `consistency_mm`
  - `elevation_mm`, `windage_mm`, `max_spread_mm`



Obr. 4: ER diagram - entity *User*, *Session_Stats*

8 Implementácia riešenia

Po navrhnutí riešenia celej práce a vyčlenení pravidiel a podmienok pre správne fungovanie sme začali s implementáciou. Táto kapitola bude rozdelená do niekoľkých častí, ako prvú veľkú časť sme zvolili nahratie obrázku terča pre vyhodnotenie. Pre následné umožnenie používateľovi vyklikať výsledky pre vyhodnotenie sme najprv museli zabezpečiť to, že obrázku nahratému do aplikácie bude transformovaná perspektíva.

8.1 Nahratie obrázku a transformácia perspektívy

Prvým krokom našej práce bolo umožniť používateľovi vybrať fotografiu terča z miestneho úložiska. V používateľskom rozhraní je to realizované pomocou HTML prvku `<input type="file">`, na ktorý je napojený "change" event listener. Po výbere súboru sa v JavaScripte vytvorí objekt `Image`, pričom sa jeho `src` nastaví na lokálnu URL súboru.

Po nahratí obrázku budeme tento obrázok vykreslovať do `<canvas>` elementu s identifikátorom `warpCanvas`. Je však potrebné zabezpečiť aby sa pri rôznych veľkostiach obrázku "nerozbilo" rozloženie našej webovej aplikácie. Následne v aplikácii vyčistíme predošlý obsah a uložíme pixelové data do 2D kontextu (`CanvasRenderingContext2D`).

8.1.1 Detekcia rohov papiera terča – podrobný popis

Proces od načítania obrázku až po spoľahlivé nájdenie štyroch rohov papierového terča je realizovaný v niekoľkých úzko previazaných krokoch. Tento reťazec transformácií sme optimalizovali preto, aby sme predišli chybám pri náročných vstupoch (neostré okraje, šum, nerovnomerné osvetlenie) a zabezpečili čo najvyššiu spoľahlivosť.

1. Načítanie a prvotné zobrazenie obrazu Prvým krokom je načítanie vybranej fotografie do OpenCV matice pomocou `cv.imread()`. Bez tohto volania by sme nemohli použiť žiadne ďalšie spracovanie:

```
const imgElement = new Image();
reader.onload = e => imgElement.src = e.target.result;
reader.readAsDataURL(file);
imgElement.onload = () => {
  let srcMat = cv.imread(imgElement);
  % srcMat je teraz vstup pre ďalšie funkcie
};
```

Výpis kódu 5: Načítanie obrázku do OpenCV matice

2. Konverzia do medzipodôb Aby sme mohli presne vyseparovať papier od pozadia, transformujeme farbu do viacerých reprezentácií – odtieň šedej pre binárne prahovanie a HSV pre farebné maskovanie. Obe konverzie používame pre robustnejšiu detekciu:

```
let gray = convertToGrayScale(srcMat);    // cv.COLOR_RGBA2GRAY
let hsv  = convertToHSV(srcMat);           // cv.COLOR_BGR2HSV
let mask = createMask(hsv, LOW, HIGH);     // inRange + inRange
```

Výpis kódu 6: Konverzia na šedú a HSV pred maskou

3. Filterovanie a reťazenie rozostrení Priame prahovanie často trpí šumom alebo drobnými škvrnami na papieri. Preto na masku aplikujeme v postupnosti:

- `applyMedianBlur()` – odstraňuje drobné salt-and-pepper artefakty,
- `applyGaussianBlur()` – uhladzuje hrany kontúr,
- `applyDefaultBlur()` – zjemňuje prechody a pripravuje na spoľahlivejšiu detekciu kontúr.

```
let mBlur = applyMedianBlur(mask);
let gBlur = applyGaussianBlur(mBlur); mBlur.delete();
let dBlur = applyDefaultBlur(gBlur); gBlur.delete();
// výsledná matica dBlur je pripravená na findContours()
```

Výpis kódu 7: Reťazenie blur operácií pre elimináciu šumu

4. Vyhľadanie a výber najväčšej kontúry Z rozostrenej masky zavoláme `findContours()` a vyberieme tú, ktorá má najväčšiu plochu (`getLargestContour()`). Táto kontúra najpravdepodobnejšie zodpovedá obrysu papierovej plochy:

```
let contours = findContours(dBlur);
let largest = getLargestContour(contours);
contours.delete();  // uvoľnenie pamäte
```

Výpis kódu 8: Detekcia a výber najväčšej kontúry

Na ďalšie spracovanie potrebujeme OpenCV maticu pixelov, ktorú získame volaním

```
const srcMat = cv.imread(warpCanvas);
```

Funkcia `cv.imread` načíta obsah plátna do objektu typu `cv.Mat`, s ktorým už dokážeme pracovať ako s maticou v OpenCV.js.

Pre odstránenie perspektívneho skreslenia implementujeme v module `utils/imageProcessing.js` funkciu

`warpPerspective(srcMat, corners)`

kde `corners` je pole štyroch bodov (získaných buď automaticky alebo manuálne kliknutím), definujúcich rohy terča. Funkcia vypočíta transformačnú maticu (homografiu) a aplikuje ju na `srcMat`, čím vznikne nový `cv.Mat` narovnaného výrezu terča. Výsledok vykreslíme späť na plátno pomocou

```
cv.imshow(warpCanvas, warpedMat);
```

a odstránime dočasné matice volaniami `srcMat.delete()` a `warpedMat.delete()`, aby sme predišli úniku pamäte.

Takto pripravený a narovnaný výrez terča je vstupom pre nasledujúce kroky - detekciu elipsy a vyklikanie zásahov.

Záver

Literatúra

1. LEGION TARGETS. *The Evolution of Target Shooting: From Practice to Competition*. 2024. Dostupné tiež z: https://www.legiontargets.com/blogs/legion/the-evolution-of-target-shooting-from-practice-to-competition.
2. THE SCOTSMAN. *Final curtain: The last live pigeon shooting event at the Olympic Games, 1900*. 2008. Dostupné tiež z: https://www.scotsman.com/sport/final-curtain-the-last-live-pigeon-shooting-event-at-the-olympic-games-1900-2474241.
3. ROBERTS, JUSTIN. *Understanding Target Shooting Scoring: A Comprehensive Guide*. Squadspot, 2024. Dostupné tiež z: https://www.squadspot.com.au/articles/understanding-target-shooting-scoring-a-comprehensive-guide.
4. ISSF. *Technical Rules for Target Shooting - Measurement of Performance Metrics*. 2024. Dostupné tiež z: https://www.issf-sports.org/theissf/rules.ashx.
5. ISSF. *10 m Air Pistol Target Plot*. [N.d.] Dostupné tiež z: https://targetshootingapp.com/images/slider/ISSF_10m_air_pistol_target_plot.png.
6. REDDIT USER U/WRIPJA2CAYQ81. *Shot dispersion plot (preview.redd.it image)*. [N.d.] Dostupné tiež z: https://preview.redd.it/wripja2cayq81.png?width=257&format=png&auto=webp&s=6377eb9119547ed636e0fc5aa44b2e20ba40f2a3.
7. UNKNOWN. *Target grouping visualization (encrypted-tbn0.gstatic.com image)*. [N.d.] Dostupné tiež z: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRTZl44GBjMSxRagKq1ncElCfObki9Y75wMOA&s.
8. GEEKSFORGEEKS. *History of JavaScript*. 2024. Dostupné tiež z: https://www.geeksforgeeks.org/history-of-javascript/.
9. MDN WEB DOCS. *What is JavaScript?* 2024. Dostupné tiež z: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/What_is_JavaScript.
10. GEEKSFORGEEKS. *Introduction to JavaScript*. 2024. Dostupné tiež z: https://www.geeksforgeeks.org/introduction-to-javascript/.
11. HOOGENRAAD, W. *6 moderných programovacích jazykov a ich nevýhody*. 2023. Dostupné tiež z: https://www.itpedia.nl/sk/2023/07/13/6-moderne-programmeertalen-en-hun-downsides/.
12. GEEKSFORGEEKS. *Introduction to Node.js*. 2023. Dostupné tiež z: https://www.geeksforgeeks.org/node-js-introduction/.

13. ATER, T. *Building Progressive Web Apps: Bringing the Power of Native to the Browser*. O'Reilly Media, 2017. ISBN 9781491961650. Dostupné tiež z: https://github.com/febycloud/PROG8110/blob/master/Tal-Ater-Building-Progressive-Web-Apps_-Bringing-the-Power-of-Native-to-the-Browser-OReilly-Media-2017.pdf.

14. SKLAR, D. *Learning PHP 5*. 1st. Sebastopol, CA: O'Reilly, 2004. ISBN 0596005601, ISBN 9780596005603. Dostupné tiež z: https://archive.org/details/learningphp50000skla. Accessed: 2025-05-29.

15. THE PHP GROUP. *PHP: Hypertext Preprocessor Manual*. The PHP Group, 2025. Dostupné tiež z: https://www.php.net/manual/en/.

16. EBOOKFOUNDATION. *free-programming-books*. 2025. Dostupné tiež z: https://github.com/EbookFoundation/free-programming-books.

17. TUTORIALSPOINT. *PHP Tutorial*. Tutorialspoint, 2025. Dostupné tiež z: https://www.tutorialspoint.com/php/index.htm.

18. GEEKSFORGEEKS. *Difference between PHP and .NET*. Sanchhaya Education Private Limited, 2021. Dostupné tiež z: https://www.geeksforgeeks.org/php/difference-between-php-and-net/.

19. RED HAT. *What is a REST API?* 2020. Dostupné tiež z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

20. WPWEB INFOTECH. *How to Build a Simple REST API in PHP*. 2025. Dostupné tiež z: https://wpwebinfotech.com/blog/how-to-build-simple-rest-api-in-php/.

21. THE PHP GROUP. *PHP: PDO - PHP Data Objects*. The PHP Group, 2025. Dostupné tiež z: https://www.php.net/manual/en/book.pdo.php.

22. OWASP FOUNDATION. *SQL Injection Prevention Cheat Sheet*. 2024. Dostupné tiež z: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html.

23. TUTORIALSPOINT. *OpenCV Python Tutorial*. [N.d.] Dostupné tiež z: https://www.tutorialspoint.com/opencv_python/index.htm.

24. OPENCV. *OpenCV Documentation*. [N.d.] Dostupné tiež z: https://docs.opencv.org/4.x/index.html.

25. TARGET SCAN, INC. *TargetScan - Pistol & Rifle*. 2023. Ver. 3.1. Dostupné tiež z: https://targetshootingapp.com/. Mobilná aplikácia pre vyhodnocovanie papierových terčov.

26. 9OCLOCK SOFTWARE GMBH. *Blackhole - Scan Targets*. 2024. Ver. 1.8.2. Dostupné tiež z: `https://www.blackhole-app.com/scan-targets-app/`. Mobilná aplikácia pre vyhodnocovanie papierových terčov.

27. ACCUSHOOT, INC. *AccuShoot*. 2024. Ver. 1.2.9. Dostupné tiež z: `https://apps.apple.com/us/app/accushoot/id6447296391`. Mobilná aplikácia pre digitálne skórovanie papierových terčov.

Použitie nástrojov umelej inteligencie

OpenAI (2025), ChatGPT o4-mini, refaktorizácia kódu.

OpenAI (2025), ChatGPT o4-mini, preformulovanie textov.

Dodatok A: Algoritmus

Dodatok B: Jednorazové nadviazanie spojenia s databázou pomocou rozhrania PDO

Nasledujúci skript zabezpečuje jednorazové nadviazanie spojenia s databázou pomocou rozhrania PHP Data Objects (PDO), pričom je ošetrený aj prípadný výskyt chýb. V prípade úspešného pripojenia sa vráti inštancia objektu PDO, ktorú možno ďalej použiť na vykonávanie databázových operácií. Táto funkcionalita je umiestnená v súbore config.php, čím sa zabezpečí, že pripojenie k databáze sa vytvára len raz a následne je možné pracovať s databázou prostredníctvom metódy Database::query() ako môžeme vidieť v podkapitole 4.4 bez nutnosti opätovného pripájania sa.

```php
<?php

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

$hostname = 'localhost';
$database = 'myAimBuddyDB';
$username = 'username';
$password = 'password';

function connectDatabase($hostname, $database, $username, $password) {
    try {
        $conn = new PDO("mysql:host=$hostname;dbname=$database", $username,
    $password);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        //echo "Connected successfully";

        return $conn;
    } catch(PDOException $e) {
        echo "Connection failed: " . $e->getMessage();

        return null;
    }
}

$db = connectDatabase($hostname, $database, $username, $password);

?>
```

Výpis kódu 9: Pripojenie k databáze pomocou PDO

Dodatok C: Slovníček pojmov