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 jadrového a fyzikálneho inžinierstva

Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025
Evidenčné číslo: FEI-XXXXX-YYYYYY

ZADANIE BAKALÁRSKEJ PRÁCE

Študent:	**Meno Priezvisko**
ID študenta:	012345
Študijný program:	jadrové a fyzikálne inžinierstvo
Študijný odbor:	elektrotechnika
Vedúci práce:	tituly, Meno Priezvisko, tituly
Vedúci pracoviska:	tituly Meno Priezvisko, tituly
Miesto vypracovania:	Ústav jadrového a fyzikálneho inžinierstva

Názov práce: **Rozšírená šablóna záverečnej práce na FEI STU v Bratislave v systéme LaTeX**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

1. Vytvorte šablónu záverečnej práce pre študentov bakalárskeho a inžinierskeho štúdia na FEI STU v Bratislave.
2. Napíšte návod na písanie záverečnej práce s použitím vytvorenej šablóny.
3. Vytvorte prehľad najpoužívanejších typov citovaných zdrojov a zdokumentujte ich tak, aby boli v súlade s normou ISO 690.

Zoznam odbornej literatúry:

1. Zákon č. 131/2002 Z. z. z 21. februára 2002 o vysokých školách a o zmene a doplnení niektorých zákonov [online]. [cit. 2024-09-01]. Dostupné na https://www.slov-lex.sk/static/pdf/2002/131/ZZ_2002_131_20240901.pdf
2. Vyhláška č. 233/2011 Z. z. Ministerstva školstva, vedy, výskumu a športu Slovenskej republiky z 1. júla 2011, ktorou sa vykonávajú niektoré ustanovenia zákona č. 131/2002 Z. z. o vysokých školách a o zmene a doplnení niektorých zákonov v znení neskorších predpisov [online]. [cit. 2024-09-01]. Dostupné na https://www.slov-lex.sk/static/pdf/2011/233/ZZ_2011_233_20201015.pdf

3. Metodické usmernenie Ministerstva školstva, vedy, výskumu a športu SR č. 56/2011 o náležitostiach záverečných prác, ich bibliografickej registrácii, kontrole originality, uchovávaní a sprístupňovaní č. 2011-11513/31015:4-071 [online]. [cit. 2024-09-24]. Dostupné na http://www.minedu.sk/metodicke-usmernenie-c-562011-o-nalezitostiach-zaverecnych-prac-ich-bibliografickej-registracii-uchovavani-a-spristupnovani/.
4. STN ISO 214: 1998. Dokumentácia. Abstrakty (referáty) pre publikácie a dokumentáciu. Bratislava: Slovenský ústav technickej normalizácie. Bratislava. Úrad pre metrológiu a skúšobníctvo Slovenskej republiky.
5. STN ISO 690: 2022. Dokumentácia – Bibliografické odkazy – Obsah, forma a štruktúra. Bratislava. Úrad pre metrológiu a skúšobníctvo Slovenskej republiky.
6. STN 01 6910: 2023. Pravidlá písania a úpravy písomností. Bratislava. Úrad pre metrológiu a skúšobníctvo Slovenskej republiky.
7. STN ISO 2145: 1997. Dokumentácia. Číslovanie oddielov a pododdielov písaných dokumentov. Bratislava. Úrad pre metrológiu a skúšobníctvo Slovenskej republiky.
8. STN ISO EN 80 000-1: 2022. Veličiny a jednotky. Bratislava. Úrad pre metrológiu a skúšobníctvo Slovenskej republiky.
9. LICHNEROVÁ, L. a B. BELLÉROVÁ. Nové pravidlá citovania podľa ISO 690 z roku 2021. *ITlib. Informačné technológie a knižnice,* 3–4/2023, 10 – 22. [cit. 2024-09-06]. Dostupné na http://doi.org/10.52036/1335793X.2023.3-4.10-22.
10. HOFTICH, M. ISO 690 biblatex style [online]. [cit. 2024-09-30]. Dostupné na https://mirrors.ibiblio.org/CTAN/macros/latex/contrib/biblatex-contrib/biblatex--iso690/biblatex-iso690.pdf.

Termín odovzdania bakalárskej práce:	31. 05. 2025
Dátum schválenia zadania bakalárskej práce:	28. 02. 2025
Zadanie bakalárskej práce schválil:	tituly Meno Priezvisko, tituly – 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

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.

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 [4]. 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 [5].

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 [6].

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-[7].

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.

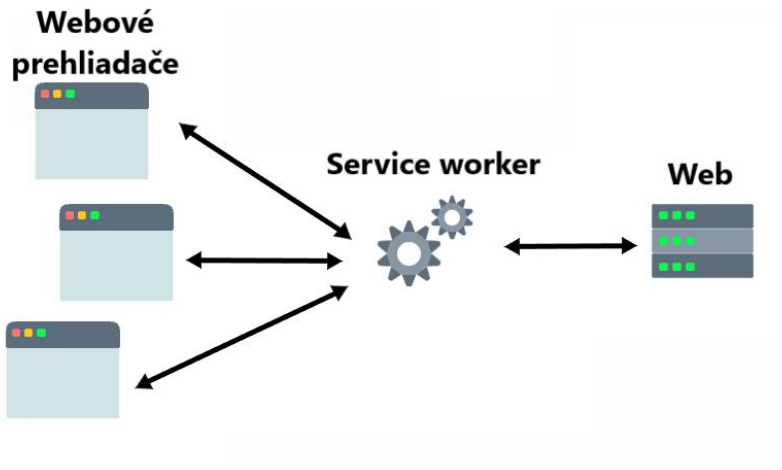
Medzi najbežnejšie využitia JavaScriptu na serveri patrí obsluha API požiadaviek, spracovanie dát, práca s databázami a správa autentifikácie. Jednoduchý a výkonný framework ako Express.js výrazne urýchľuje tvorbu webových serverov. Týmto spôsobom sa z JavaScriptu stal univerzálny jazyk schopný pokrývať celý vývojový cyklus webovej aplikácie [8].

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 [9]

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 öffline first-[9].

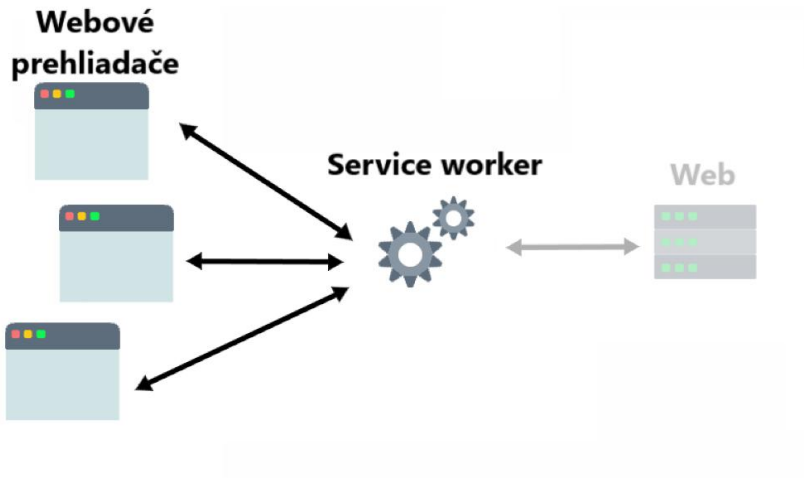
Notifikácie - Progresívne webové aplikácie možu posielať používateľom takzvané "Push notificationsäj niekoľko dní po návšteve stránky. Tieto notifikácie sú skvelým nástrojom ako si môže stránka udržať viditeľnosť a svojich navštevovateľov. [9].

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 [9]

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 [10]. 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 ne-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 [11].
- 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 [11].
- 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 [12].
- 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 [11].

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) [13]. Zároveň v porovnaní s inými jazykmi ponúka PHP menej pokročilé mechanizmy pre vykresľovanie a spracovanie výnimiek, toto všetko komplikuje odhalovanie a ladenie chýb [14]. PHP kód vyžaduje prítomnosť webového servera ako napríklad Apache alebo Nginx s nainštalovaným interpretrerom, čo ho robí menej flexibilným pre skriptovanie na príkazovom riadku [13].

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 [15].

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 [16].

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 [17].

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 [18]. 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 5 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 [17].

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 [19]. V práci spomínam funkcie mnou považované za kľúčové pre túto prácu, samotná knižnica disponuje nikoľkonásobne viac funkciami, pre viac informácií je dostupná dokumentácia knižnice [20].

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.

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, 2024-09. Dostupné tiež z: https://www.squadspot.com.au/articles/understanding-target-shooting-scoring-a-comprehensive-guide.
4. GEEKSFORGEEKS. *History of JavaScript*. 2024. Dostupné tiež z: https://www.geeksforgeeks.org/history-of-javascript/.
5. 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.
6. GEEKSFORGEEKS. *Introduction to JavaScript*. 2024. Dostupné tiež z: https://www.geeksforgeeks.org/introduction-to-javascript/.
7. 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/.
8. GEEKSFORGEEKS. *Introduction to Node.js*. 2023. Dostupné tiež z: https://www.geeksforgeeks.org/node-js-introduction/.
9. 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.
10. 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.
11. THE PHP GROUP. *PHP: Hypertext Preprocessor Manual*. The PHP Group, 2025. Dostupné tiež z: https://www.php.net/manual/en/.

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

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

14. GEEKSFORGEEKS. *Difference between PHP and .NET*. Sanchhaya Education Private Limited, 2021-01. Dostupné tiež z: https://www.geeksforgeeks.org/php/difference-between-php-and-net/.

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

16. 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/.

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

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

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

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

Použitie nástrojov umelej inteligencie

OpenAI (2025), ChatGPT 4o, časť **??**, generovanie vzorca záznamu použitia AI.

OpenAI (2024), ChatGPT 3, Záver, generovanie textu.

OpenAI (2025), ChatGPT 4o, dodatok B, generovanie kódu programu.

OpenAI (2024), ChatGPT 3, dodatok C, generovanie zoznamu.

Dodatok A: Algoritmus

Dodatok B: Výpis dlhého kódu

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 5: Pripojenie k databáze pomocou PDO

Dodatok C: Slovníček pojmov